Warsaw University of Technology

DISCIPLINE OF SCIENCE INFORMATION AND COMMUNICATION TECHNOLOGY

FIELD OF SCIENCE ENGINEERING AND TECHNOLOGY

Ph.D. Thesis

Jan Dubinski

Reliable and Safe Generative Models

Supervisor

Prof. Przemysław Rokita

WARSAW 2023

Acknowledgements

I would like to begin by expressing my deepest gratitude to my supervisor, Przemysław Rokita, for his steady guidance and invaluable support throughout my scientific journey. I am especially thankful for the trust he placed in me during my doctoral studies.

I am also grateful to the remarkable researchers with whom I had the privilege to collaborate. I owe special thanks to Tomasz Trzciński for his mentorship, encouragement, and guidance in helping me grow as a researcher. I am deeply indebted to Adam Dziedzic and Franziska Boenisch for their support in my academic development and for our joint work on the safety of artificial intelligence, which shaped my understanding of the field.

My appreciation also goes to my colleagues and coauthors from Warsaw, Saarbrücken, and Rome for their dedication to our shared projects and conducting research together, among them: Kamil Deja, Antoni Kowalczuk, Stanisław Pawlak, Filip Szatkowski, Daniel Marczak, Michał Bortkiewicz, Patryk Będkowski, Tomasz Michalak, Stanisław Dziembowski, Simone Scardapane, and Stefano Giagu.

Finally, I am profoundly grateful to my family for their unwavering support and for providing the foundation that enabled me to pursue a scientific career. Above all, I wish to thank my wife and friend, Joanna Kaleta, with whom I share not only my life but also the academic journey.

Reliable and Safe Generative Models

This thesis presents a series of solutions advancing the development of reliable and safe generative models. We propose novel approaches to generative modeling that address real-world applications, with particular emphasis on high-energy physics, while also introducing methods to safeguard the intellectual value of both models and their training data. Through these contributions, we strengthen the integrity and reliability of generative machine learning systems for model developers, users, and data owners.

In the first part, we focus on the design of reliable generative models for scientific use cases. We address the challenge of simulating high-energy physics experiments at CERN European Organization for Nuclear Research, by developing machine learning alternatives to conventional detector simulations at the Large Hadron Collider. Our contributions include a formulation of generative adversarial networks that aligns the diversity of generated samples with real data, as well as a generative mixture-of-experts framework that captures the multimodal nature of experimental distributions.

In the second part, we turn to the protection of intellectual value in deep learning models. As machine learning models become increasingly capable of solving practical tasks, their utility and intrinsic value grow, making protection a natural extension of their development. In this part, we introduce the first defense against unauthorized replication of encoder models, which detects extensive probing of the model's response space and prevents extraction attempts.

In the final part, we extend this focus from models to the data that underpins them, recognizing that safeguarding training data is equally critical to ensuring trustworthy machine learning. We demonstrate the limitations of existing data identification techniques and propose a reliable method for detecting copyrighted datasets used in the training of diffusion models. Building on this, we analyze privacy risks in emerging image autoregressive models and successfully adapt our method to this new, previously unexplored class of architectures.

Overall, this thesis contributes to the development of generative models that reliably address real-world scientific problems while also providing mechanisms to safeguard both models and data, supporting the creation of a dependable, trustworthy, and accountable environment for generative machine learning.

Keywords: Generative Models, Generative Adversarial Networks, Diffusion Models, Image Autoregressive Models, High-Energy Physics, Machine Learning Safety

Niezawodne i Bezpieczne Modele Generatywne

Niniejsza praca przedstawia zestaw rozwiązań wspierających rozwój wiarygodnych i bezpiecznych modeli generatywnych. Proponujemy nowe podejścia do modelowania generatywnego, ukierunkowane na praktyczne zastosowania, ze szczególnym uwzględnieniem fizyki wysokich energii. Jednocześnie wprowadzamy metody chroniące własność intelektualną zawartą w modelach i danych treningowych. Dzięki temu wzmacniamy bezpieczeństwo i niezawodność generatywnego uczenia maszynowego, wspierając zarówno twórców i użytkowników modeli, jak i właścicieli danych.

W pierwszej części pracy koncentrujemy się na tworzeniu wiarygodnych modeli generatywnych na potrzeby zastosowań naukowych. Szczególny nacisk kładziemy na symulację eksperymentów z zakresu fizyki wysokich energii prowadzonych w Europejskiej Organizacji Badań Jądrowych (CERN). Proponujemy rozwiązania oparte na uczeniu maszynowym jako alternatywę dla tradycyjnych metod symulacji stosowanych w Wielkim Zderzaczu Hadronów. Osiągnięte rezultaty obejmują opracowanie generatywnych sieci adwersarialnych wiernie odwzorowujących różnorodność danych treningowych, a także stworzenie modelu opartego na mieszance ekspertów, umożliwiającego uchwycenie wielomodalnej natury generowanych danych.

W drugiej części pracy skupiamy się na ochronie wartości intelektualnej w modelach uczenia głębokiego. Wraz z rosnącą zdolnością modeli do rozwiązywaniu konkretnych problemów wzrasta również ich wartość. Naturalnie tworzy to potrzebę opracowania metod jej ochrony. W odpowiedzi proponujemy pierwszą metodę obrony przed nieautoryzowanym odtwarzaniem modeli typu enkoder, pozwalającą wykrywać próby eksploracji przestrzeni odpowiedzi modelu przez atakującego.

Ostatnia część pracy rozszerza perspektywę ochrony wartości intelektualnej z modeli na dane. Pokazujemy, że istniejące podejścia do identyfikacji danych używanych do trenowania modeli dyfuzyjnych mają istotne ograniczenia i proponujemy efektywną metodę wykrywania wykorzystania zbiorów chronionych prawem autorskim. Jako pierwsi analizujemy również zagrożenia dla prywatności w autoregresyjnych modelach wizyjnych i skutecznie adaptujemy naszą metodę do tego typu architektur.

Podsumowując, niniejsza praca wnosi wkład w rozwój modeli generatywnych zdolnych w sposób niezawodny odpowiadać na rzeczywiste wyzwania naukowe, a jednocześnie dostarcza mechanizmy ochrony zarówno modeli, jak i danych. Tym samym wspiera tworzenie godnego zaufania, wiarygodnego i odpowiedzialnego ekosystemu uczenia maszynowego opartego na metodach generatywnych.

Słowa kluczowe: Modele Generatywne, Sieci Generatywne Kontradyktoryjne, Modele Dyfuzyjne, Modele Autoregresyjne Obrazów, Fizyka Wysokich Energii, Bezpieczeństwo Uczenia Maszynowego

Contents

1. Introduction

Generative models are a cornerstone of modern machine learning. Architectures such as Generative Adversarial Networks (GANs) [105], Diffusion Models [214], and Autoregressive Models [223] have demonstrated a remarkable ability to synthesize data that closely mirrors real-world distributions. Their applications span diverse domains, including realistic image generation [24, 223], video creation [138, 235], speech synthesis [53, 128], and music generation [62, 92]. At the same time, these models are increasingly used in scientific contexts, where they accelerate simulations in physics [162, 171, 245], support drug discovery [143, 250], advance computational medicine [34, 37], and enable progress in material science [56, 262].

A key feature of generative models is their ability to approximate complex data distributions and to generate new samples that follow the learned structure. This capability has driven advances in both creative and scientific domains, but it also reveals important limitations and risks. On the one hand, designing models that move beyond academic benchmarks and reliably capture complex real-world distributions remains a central challenge. On the other hand, the increasing dependence on large-scale generative systems raises concerns about protecting the growing intellectual value embedded in both models and their training data.

This dissertation addresses both aspects. The first part focuses on developing reliable generative models for real-world scientific applications. Specifically, we examine high-energy physics experiments at CERN, the European Organization for Nuclear Research, and propose generative methods for simulating detector responses at the Large Hadron Collider. Our contributions begin with a method for controllably increasing diversity in GAN-generated samples. Building on this, we introduce a generative mixture-of-experts framework that delivers higher fidelity and more accurately models the multimodal distributions characteristic of particle collision data.

As generative models mature and prove their value in solving practical scientific tasks, the results of the first part also highlight a new dimension: their growing utility makes them increasingly valuable intellectual assets. A model that can faithfully simulate complex physical processes, for example, embodies not only scientific insight but also significant investment in data, design, and computation. Developing reliable models is therefore only one side of the challenge. Once such systems

demonstrate real-world utility, it becomes equally important to ensure that they are safeguarded against misuse and unauthorized replication, so that the benefits of their development remain with their creators and intended users.

The second part of this dissertation turns to the problem of safeguarding intellectual value in machine learning, both in models and in data. To protect the value of models against unauthorized replication, we develop the first active defense for encoder models, based on detecting extensive probing of the model's response space and preventing extraction.

Safeguarding models alone is not sufficient, since the value of generative systems also depends on the data that trains them.

We then shift focus to the protection of training data. In this context, we demonstrate the shortcomings of existing data identification techniques for large diffusion models and show that these methods are limited as audit tools for protecting the rights of data owners. To address this, we propose a method for reliably detecting copyrighted datasets. Finally, we examine the privacy risks introduced by emerging image autoregressive models and adapt our training data identification method to this new class of generative architectures.

Overall, this work contributes to the development of trustworthy generative models that not only support demanding scientific use cases such as high-energy physics simulations but also ensure the protection of models and data, providing safeguards for their creators and owners, contributing to the creation of a dependable, trustworthy, and accountable environment for generative machine learning.

1.1. Research Goals

In this work, we focus on two complementary aspects of generative modelling. The first is the design of reliable models that can be applied to real scientific challenges, with a particular emphasis on high-energy physics simulations at CERN. The second is the development of methods that safeguard the intellectual value contained in both models and training data, addressing the growing importance of ownership and protection in the era of large-scale generative systems. From this perspective, the central research goals of this dissertation are as follows:

1. *Design reliable generative models that can solve specific real-world problems, such as high-energy physics simulations.*
2. *Develop safeguards that protect the intellectual value embedded in deep learning models against unauthorized extraction or replication.*
3. *Protect the value of training data and support the rights of data owners, particularly in the era of large-scale generative models.*

Each of these research goals is addressed in a dedicated part of the dissertation. Chapters 4 and 5 explore the first goal by introducing methods for building reliable generative models in high-energy physics. In particular, we propose a modification of conditional GANs that selectively increases the diversity of generated samples to better match the properties of real collision data, and we develop ExpertSim, a Mixture-of-Generative-Experts framework that achieves high fidelity in simulating responses of the Zero Degree Calorimeter at the ALICE experiment while maintaining computational efficiency.

Chapter 6 addresses the second goal by turning to model protection. Here, we introduce *Bucks for Buckets (B4B)*, the first active defense against stealing encoder models. This method detects extensive probing of the embedding space, introduces cost functions that penalize extraction attempts, and employs per-user transformations to mitigate Sybil-style attacks, ultimately stopping unauthorized replication.

The final goal is explored in Chapters 7–9, where we focus on the protection of training data and the rights of data owners in large-scale generative models. We begin by evaluating the limitations of membership inference attacks against diffusion models and propose a more realistic assessment framework. Building on this, we introduce *Copyrighted Data Identification (CDI)*, a method that reliably detects whether specific datasets have been used to train large diffusion models, providing a practical tool for copyright protection. Finally, we extend the analysis to newly proposed image autoregressive models, where we show that these architectures are particularly prone to privacy leakage and extend training data identification techniques to this paradigm.

1.2. Thesis Content

The dissertation begins with Chapter 1, which introduces the motivation, research problems, and objectives of this work. Chapter 2 then provides the theoretical background on generative modelling, covering probabilistic foundations, generative adversarial networks, diffusion models, and autoregressive architectures. Chapter 3 surveys related work, situating the contributions of this dissertation within the broader research landscape in machine learning, privacy, and intellectual property protection.

The first part addresses the problem of developing reliable generative models for scientific applications, with a focus on high-energy physics. In this domain, conventional Monte Carlo simulations provide highly accurate detector modelling but at prohibitive computational cost, creating a demand for efficient machine

learning alternatives. Standard conditional GANs, while promising, often collapse to a limited set of modes and fail to reproduce the full variability of collision data. Chapter 4 investigates this limitation and proposes *Selective Diversity Increase GAN (SDI-GAN)*, a modification of the training objective that improves sample diversity in a controlled manner without sacrificing fidelity. Nevertheless, a single generative model remains insufficient for capturing the intrinsically multimodal distributions of calorimeter responses. Chapter 5 addresses this challenge with *ExpertSim*, a mixture-of-generative-experts framework in which specialized models are coordinated by a routing network, enabling faithful and computationally efficient simulation of detector responses at CERN.

The second part examines the protection of machine learning models as intellectual property. Modern encoder architectures require significant computational resources to train and constitute valuable assets, yet they are increasingly threatened by model stealing attacks. In such attacks, adversaries interact with a deployed model through an API to reconstruct a high-fidelity substitute, undermining both the economic and scientific value of the original system. Chapter 6 introduces *Bucks for Buckets (B4B)*, the first active defense specifically designed to counter encoder stealing. The method quantifies the coverage of queries in the embedding space, detects probing indicative of extraction, and injects adaptive perturbations to degrade adversarial reconstructions while maintaining utility for legitimate users.

The third part turns to the protection of training data, which underpins modern generative models but is often assembled from large-scale, uncurated, and potentially copyrighted sources. Ensuring that data owners can verify whether their datasets were used in model training is an open problem, as existing membership inference attacks are unreliable for large diffusion models trained on billions of samples. Chapter 7 provides a critical evaluation of these techniques, demonstrating their limitations under realistic conditions and introducing a more rigorous framework for assessing privacy leakage. Building on this analysis, Chapter 8 presents *Copyrighted Data Identification (CDI)*, a dataset-level auditing method that aggregates multiple membership signals and applies statistical testing, enabling reliable detection of copyrighted datasets in diffusion models. Finally, Chapter 9 extends the study to image autoregressive models, a new class of generative architectures. We show that these models exhibit stronger memorization tendencies than diffusion models, making them particularly vulnerable to privacy and copyright leakage, and we adapt dataset identification methods to effectively audit this setting.

The dissertation concludes in Chapter 10, which synthesizes the findings of all parts and outlines future research directions toward generative models that are efficient, scientifically reliable, and robust against both model and data misuse.

2. Background

In this chapter, we establish the theoretical and methodological foundations that support the remainder of this thesis. We review the main families of modern deep generative models, outline the requirements of fast simulation in high-energy physics, and introduce the security considerations that motivate later analyses.

2.1. Generative Modeling

This section provides a concise overview of the generative modeling landscape relevant to our work. While not exhaustive, it introduces the principles and motivations behind the architectures discussed later.

2.1.1. Generative Autoencoders

Autoencoders are one of the earliest neural architectures proposed for unsupervised representation learning. They consist of two networks trained jointly: an encoder q_ϕ that maps the input $\mathbf{x}$ into a latent code $\mathbf{z}$, and a decoder p_θ that reconstructs the original data point from this code. The model is trained by minimising the reconstruction loss, typically the squared error between the input and the reconstruction:

$$L_r = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2, \tag{2.1}$$

where $\hat{\mathbf{x}} = p_\theta(q_\phi(\mathbf{x}))$. By constraining the latent space to have lower dimensionality than the input, autoencoders learn compressed representations that capture salient features of the data. These latent codes can be used in downstream tasks such as dimensionality reduction [241], anomaly detection [194], image retrieval [217], or compression [222].

Despite these advantages, the vanilla autoencoder is not a generative model in the strict sense. Because the encoder-decoder mapping is deterministic, the model cannot describe a full data distribution, only point reconstructions. Furthermore, latent codes of training examples tend to form disjoint regions of the latent space without smooth transitions. As a result, sampling $\mathbf{z}$ from a simple prior distribution

such as $\mathcal{N}(0, I)$ yields invalid outputs, making the model unsuitable for generating new examples.

To address this limitation, extensions of the autoencoder introduce probabilistic components and regularisation strategies that enforce structure in the latent space. This leads to the family of generative autoencoders, where the encoded representations are no longer arbitrary but follow a distribution that supports meaningful sampling. The most influential formulation in this direction is the Variational Autoencoder.

Variational Autoencoder

The Variational Autoencoder (VAE) [135] reformulates the autoencoder in probabilistic terms. Instead of mapping each input deterministically to a single latent point, the encoder defines a distribution over latent variables conditioned on the input:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_\phi(\mathbf{x}), \sigma^2_\phi(\mathbf{x})I), \tag{2.2}$$

where the encoder outputs the mean and variance of a Gaussian distribution. The decoder is likewise defined as a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$, turning the reconstruction process into probabilistic inference.

The training objective of the VAE is derived from the marginal likelihood of the data under the generative model $p_\theta(\mathbf{x})$. Since direct maximisation is intractable, [135] introduce a variational lower bound (ELBO):

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})], \tag{2.3}$$

where the first term corresponds to the reconstruction likelihood, and the second regularises the approximate posterior towards a prior distribution $p(\mathbf{z})$, typically a standard normal $\mathcal{N}(0, I)$.

Optimising this objective encourages the encoder to map training examples into overlapping regions of the latent space structured by the prior. This ensures that new latent points sampled from $p(\mathbf{z})$ decode into realistic examples, turning the autoencoder into a generative model.

A technical challenge arises because the encoder must sample $\mathbf{z}$ from the distribution it defines, which breaks gradient flow. To solve this, the reparametrisation trick is used: instead of sampling $\mathbf{z}$ directly, one samples auxiliary noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and computes

$$\mathbf{z} = \mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x}) \cdot \boldsymbol{\epsilon}. \tag{2.4}$$

This formulation allows gradients to propagate through μ_ϕ and σ_ϕ, enabling end-to-end training with backpropagation.

The VAE provides a principled way to impose continuity in the latent space and to balance reconstruction accuracy with generative quality. However, it often suffers from overly smooth outputs due to the Gaussian assumption and the averaging effect of the reconstruction loss, which motivated the development of adversarial approaches discussed in the next section.

2.1.2. Generative Adversarial Networks

Generative Adversarial Networks (GANs) [105] represent a different philosophy of generative modelling compared to autoencoders. Instead of compressing and reconstructing examples, GANs aim to learn the data distribution directly by mapping noise variables $\mathbf{z} \sim \mathcal{N}(0, I)$ into realistic outputs $\hat{\mathbf{x}} = G_\theta(\mathbf{z})$.

The training procedure involves two neural networks: a generator G_θ and a discriminator D_ϕ. The generator is tasked with producing synthetic examples, while the discriminator evaluates whether a given sample comes from the true distribution p_{data} or from the generator. Both networks are optimised jointly in a minimax game, where the discriminator maximises its classification accuracy and the generator minimises it. This objective can be written as

$$\min_\theta \max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \left[\log D_\phi(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} \left[\log(1 - D_\phi(G_\theta(\mathbf{z})))\right]. \tag{2.5}$$

Because the generator is trained through the gradients of the discriminator, improvements of one network push the other to adapt, creating an adversarial dynamic. With sufficient capacity and stable optimisation, this process yields samples that are indistinguishable from real ones.

Adversarial training has important advantages. Most notably, it avoids the blurring effects common in reconstruction-based models, and GANs are therefore known for producing visually sharp and detailed outputs [13, 178]. This property has made them one of the most widely adopted generative methods across computer vision and beyond.

Nevertheless, there are two critical drawbacks. First, the minimax game is unstable in practice and often converges to degenerate solutions. A well-known failure mode is mode collapse, in which the generator produces only a narrow set of outputs, ignoring the diversity of the training distribution [197]. Second, GANs do not include an encoder, which means they lack the ability to map data points back into a latent space. As a result, they are less suitable for representation learning compared to autoencoder-based approaches.

2.1.3. Diffusion-Based Deep Generative Models

A more recent class of generative models are Diffusion-Based Deep Generative Models (DDGMs) [119, 214]. These methods define the generative process not as a single transformation from latent noise to data, but as the reversal of a gradual noising procedure. The key idea is to corrupt training examples step by step with Gaussian noise until they become indistinguishable from pure noise, and then train a neural network to invert this process. By learning how to denoise progressively, the model acquires the ability to sample realistic data starting from random noise.

Formally, let $\mathbf{x}_0$ denote a clean data sample. The forward diffusion process defines a Markov chain of noisy variables $\mathbf{x}_1, \ldots, \mathbf{x}_T$, where noise is added at each step according to a variance schedule $\beta_1, \ldots, \beta_T$:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\big(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}\big). \tag{2.6}$$

Because this process is Gaussian and linear, it has a closed-form expression that relates $\mathbf{x}_t$ directly to the original data $\mathbf{x}_0$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\big(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\,\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I}\big), \tag{2.7}$$

where $\alpha_t = 1-\beta_t$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. After a sufficient number of steps T, the distribution of $\mathbf{x}_T$ approaches an isotropic Gaussian.

The generative model is defined as the reverse Markov chain, parameterised by a neural network $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ that attempts to invert the noising procedure:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\big(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)\big). \tag{2.8}$$

Sampling from the model consists of starting with $\mathbf{x}_T \sim \mathcal{N}(0, I)$ and repeatedly applying these learned reverse transitions until $\mathbf{x}_0$ is obtained.

Training is carried out by minimising a variational bound on the negative log-likelihood of the data [214]. In practice, [119] showed that this objective can be simplified to predicting the Gaussian noise $\boldsymbol{\epsilon}$ that was added at a given step t. For a randomly chosen time index, the training loss becomes

$$L_{t,\text{simple}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\Big[\, \| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t) \|^2 \Big], \tag{2.9}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and $\boldsymbol{\epsilon}_\theta$ is a neural network (often a U-Net [192]) trained to predict the injected noise. This simplified objective is now the standard way of training diffusion models.

Compared to GANs, diffusion models are easier to optimise and rarely suffer from collapse. They naturally capture diverse modes of the training distribution and

can generate outputs of very high quality. Their main drawback is computational: both training and sampling require running the process over hundreds or thousands of steps, although recent work explores acceleration strategies [24, 196].

Latent Diffusion Models

A major improvement in the efficiency of diffusion models was introduced with Latent Diffusion Models (LDMs) [24]. Instead of applying the forward and reverse processes directly in pixel space, the data is first compressed into a latent representation using a generative autoencoder, as discussed in Section 2.1.1. Concretely, an encoder E_ϕ maps the input $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ to a lower-dimensional latent $\mathbf{z} = E_\phi(\mathbf{x})$, and a decoder D_ψ reconstructs the image from this representation, $\hat{\mathbf{x}} = D_\psi(\mathbf{z})$. The autoencoder is trained with a combination of reconstruction and perceptual losses to ensure that $\mathbf{z}$ captures the semantic structure of the data while remaining computationally compact.

Once this autoencoder is trained, the diffusion process is applied in the latent space $\mathbf{z}$ rather than in the original pixel space. Formally, the forward process becomes

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}\big(\mathbf{z}_t; \sqrt{1-\beta_t}\,\mathbf{z}_{t-1}, \beta_t\mathbf{I}\big), \tag{2.10}$$

and the reverse process is parameterised by a neural network $p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t)$. Sampling begins from Gaussian noise in latent space and produces a sequence $\mathbf{z}_T, \ldots, \mathbf{z}_0$, which is then decoded into an image using D_ψ.

This formulation dramatically reduces the dimensionality of the diffusion process, as the latent space is typically several times smaller than pixel space. As a result, training and sampling become significantly more efficient, while the autoencoder ensures that perceptually relevant features are preserved. Beyond efficiency, the separation of the autoencoder from the diffusion process also allows conditioning and architectural modifications to be applied flexibly at the latent level.

Because LDMs build directly on autoencoder architectures, they connect the advantages of representation learning with the stability of diffusion. In this thesis, they play a key role in the analysis of privacy and copyright risks, since most contemporary large-scale generative models, including Stable Diffusion, are trained in this latent formulation.

2.1.4. Image Autoregressive Models

Autoregressive (AR) models define the probability of an image as a sequential product of conditional distributions. Given a sequence of elements $\mathbf{t} = (t_1, \ldots, t_N)$ the model assigns

$$p(\mathbf{t}) = \prod_{n=1}^{N} p(t_n \mid t_{<n}), \tag{2.11}$$

where N is the length of the sequence and t_n is the n-th element. Training proceeds by maximising the log-likelihood, or equivalently, minimising the negative log-likelihood objective

$$\mathcal{L}_{\text{AR}} = \mathbb{E}_{\mathbf{t}\sim\mathcal{D}_{\text{train}}}\Big[-\log p(\mathbf{t})\Big]. \tag{2.12}$$

Early work treated the image $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ itself as the sequence, scanning pixels in raster order (row by row, left to right) and modelling $t_n = x_n$ as the value of the n-th pixel [52]. While this approach guarantees a valid likelihood, it is computationally demanding: the number of steps equals the number of pixels, and long-range dependencies are hard to capture.

To reduce sequence length and increase modelling power, modern image AR models operate in a token space rather than directly on pixels. This is achieved with a vector-quantised autoencoder such as VQ-VAE [182]. As described in Section 2.1.1, these models first map the image into a lower-dimensional latent representation $\mathbf{z} \in \mathbb{R}^{h\times w\times d}$, which is then quantised using a learnable codebook $\mathcal{C} = \{e_k\}_{k=1}^{K}$ of discrete embeddings. Each latent vector z_i is replaced with its nearest neighbour in the codebook,

$$q(z_i) = e_k \quad \text{where} \quad k = \arg\min_j \|z_i - e_j\|_2. \tag{2.13}$$

The quantised latents form a grid of indices $\mathbf{t} \in \{1,\dots,K\}^{h\times w}$, which is then flattened into a one-dimensional sequence of tokens. In this formulation, Equation 2.11 is applied not to pixels but to tokens, and the generation task becomes directly analogous to language modelling. The reduction from $N = H \times W$ pixel values to $N = h \times w$ tokens enables training at higher resolutions and with better efficiency.

Autoregressive image models thus employ transformer decoders [230], where causal masking ensures that each prediction depends only on past tokens. The analogy with natural language is direct: just as language AR models generate text one token at a time [177], image AR models generate visual tokens sequentially until the full sequence is complete. The generated tokens are then projected back into the image space by the decoder of the VQ-VAE.

While raster-order prediction of tokens is the simplest option, images do not possess a natural left-to-right structure. Several recent works therefore propose alternative factorisations. Randomised Autoregressive Visual Generation (RAR) [256] trains with randomised token orders, which in expectation provide each token with

both left and right context while still allowing deterministic generation at inference. Visual Autoregressive Modeling (VAR) [223] instead generates images in a coarse-to-fine fashion, predicting token maps at progressively higher resolutions so that global structure is established before fine details are added. Finally, Autoregressive Image Generation without Vector Quantization (MAR) [145] removes the discrete codebook entirely, treating tokens as continuous vectors and predicting them with parametric densities rather than categorical distributions.

Despite these differences, all models share the same underlying principle: to represent images as autoregressive sequences. In recent years, these methods have matured into a parallel line of research to diffusion models, achieving comparable or even superior results in high-resolution image generation. Its distinct properties motivate the final part of this thesis, where we study their privacy risks and evaluate how they differ from diffusion models in terms of data leakage.

2.2. Fast Machine Learning Simulations at CERN

Having introduced the main families of generative models, we now turn to a setting where they can be applied in practice. This section describes the main challenges of leveraging generative models in high-energy physics experiments at CERN and shows how they motivate the methods developed in this thesis.

High-energy physics experiments at CERN rely on large-scale simulations to interpret detector signals and to compare observed collision events with theoretical models. The standard approach is based on Monte Carlo methods, which stochastically model the propagation of particles through detectors by computing successive interactions with matter. While highly accurate, these simulations are computationally intensive. The ALICE experiment alone required more than half a million CPU cores in 2023 to meet its simulation demands [50]. This creates a bottleneck that slows the experimental cycle.

The recent introduction of generative machine learning models offers a promising alternative. Unlike Monte Carlo methods, which repeatedly solve the underlying physics equations, neural models can learn the mapping between particle properties and detector responses directly from data. Once trained, they generate new samples by forward passes through a network, which is orders of magnitude faster and easily parallelised on GPUs. However, these approaches face domain-specific challenges. Detector responses often follow heterogeneous distributions that differ substantially between simulation conditions.

Therefore, the main objective of fast simulation research at CERN is to develop methods that combine simulation fidelity with computational scalability. Generative

models must reproduce subtle detector effects while maintaining simulation speed. This dual requirement motivates architectures such as the SDI-GAN or ExpertSim presented later in this thesis.

2.2.1. The Zero Degree Calorimeter at the ALICE Experiment

ALICE (A Large Ion Collider Experiment) is one of the four major detectors at the Large Hadron Collider. Its primary goal is to study the quark–gluon plasma, a state of matter believed to have existed in the first microseconds after the Big Bang. To probe this regime, ALICE analyses the products of heavy-ion collisions at TeV energies [70].

Among its subdetectors, the Zero Degree Calorimeter (ZDC) plays a crucial role in centrality determination. The ZDC consists of two quartz-fibre calorimeters placed at zero degrees with respect to the beam line: the Proton Calorimeter (ZP) and the Neutron Calorimeter (ZN). Their design leverages Cherenkov light emission: charged particles from collision remnants produce showers in silica fibres, and the emitted photons are collected by photomultiplier tubes. The ZP and ZN capture complementary signals from non-interacting protons and neutrons, which allows reconstruction of the collision geometry.

Structurally, each calorimeter is organised as a grid of fibres coupled to distinct readout channels. This configuration enables spatially resolved measurements of energy deposition, which can be represented as two-dimensional images with intensities corresponding to photon counts. In simulation tasks, each ZDC response is therefore treated as a conditional image, associated with a set of collision variables such as energy, momentum, and impact parameter. The diversity of these responses is particularly challenging for generative modelling, as the distributions split into qualitatively distinct groups ranging from low-intensity diffuse patterns to high-intensity focused showers. In the following chapters, we propose and evaluate machine learning architectures that address these challenges, providing both fidelity and efficiency in ZDC simulations.

Figure 2.2.1. Fast simulation of the Zero Degree Calorimeter

2.3. Protecting Value in Models and Data

Machine learning systems primarily concentrate value in two assets: the training data $\mathcal{D}_{\text{train}}$, which is expensive to curate and often subject to licensing or privacy constraints, and the trained model M_θ, which distills this data into parameters that can be deployed at scale. Both are vulnerable to adversarial exploitation. Attacks can either expose properties of the data or replicate the functionality of the model, thereby undermining the investment in collecting $\mathcal{D}_{\text{train}}$ and training M_θ. Importantly, many of the same techniques that threaten confidentiality also enable auditing: a membership inference attack can reveal whether an individual record was present in $\mathcal{D}_{\text{train}}$, but the same mechanism can be used to test whether a model was trained on copyrighted content. Similarly, dataset inference and model extraction serve simultaneously as attack vectors and as tools for asserting provenance.

In this section, we review three families of methods: *model stealing*, which targets the parameters or functionality of M_θ; *membership inference*, which asks whether a specific record was used in training; and *dataset inference*, which extends this reasoning to groups of samples. Together, they form the basis for the empirical studies in later chapters of this thesis.

2.3.1. Model Stealing Attacks

Model stealing (or model extraction) occurs when an adversary reproduces a target model's functionality without direct access to its parameters or training set [12]. The attacker interacts with the model through queries $q_i \in \mathcal{Q}$ and records outputs $M_\theta(q_i)$. With these pairs, a surrogate model $M_{\theta'}$ is trained to minimise the discrepancy

$$L(\theta') = \frac{1}{n} \sum_{i=1}^{n} \ell\big(M_{\theta'}(q_i), M_\theta(q_i)\big), \tag{2.14}$$

where ℓ is an appropriate task loss and n is the number of queries. This basic strategy has been applied to classification, regression, and generative models alike, with effectiveness depending primarily on the richness of the query set. The end result is a surrogate model $M_{\theta'}$ that closely reproduces the outputs and behaviour of the original model M_θ, effectively granting the adversary a functional copy of the target system.

The efficiency of this process can be improved through adaptive querying. Rather than issuing queries at random, adversaries can strategically select inputs that reveal the model's behaviour in high information regions, for example by probing uncertain decision boundaries or sampling diverse modes of the output space [125, 169]. These ideas extend beyond classifiers. For self supervised encoders, recent

work shows that matching representations using contrastive objectives or MSE is highly effective [85, 205]. In particular, stealing SSL encoders can be made much more query efficient by exploiting augmentation invariances, where a single victim returned representation can supervise many locally generated augmentations [153].

2.3.2. Membership Inference Attacks

Membership inference attacks (MIAs) aim to determine whether a specific record $\mathbf{x}$ belongs to the training set $\mathcal{D}_{\text{train}}$ of a model M_θ [211]. Formally, the attacker seeks a decision rule

$$A(\mathbf{x}) = \mathbb{1}[\mathbf{x} \in \mathcal{D}_{\text{train}}], \tag{2.15}$$

based only on the observable behaviour of M_θ.

The simplest attack compares losses on $\mathbf{x}$ against a threshold:

$$\hat{y} = \mathbb{1}[\ell(M_\theta(\mathbf{x}), y) \leq \tau], \tag{2.16}$$

where τ is chosen such that points with unusually low loss are predicted as members. More advanced strategies use shadow models trained on reference datasets to simulate the behaviour of M_θ, and then train a binary classifier to distinguish member from non-member samples based on the loss signal.

MIAs exploit the gap between training and generalisation behaviour. Overfitting increases the success of such attacks, but even well-regularised models can leak membership information through subtle distributional cues. While this presents a privacy risk, the same mechanism can be repurposed to audit generative models: if $\mathbf{x}$ is a copyrighted image, a successful membership inference indicates that $\mathbf{x}$ was included in the training set of M_θ.

2.3.3. Dataset Inference Attacks

Dataset Inference (DI) [157] aims to determine whether a specific dataset was included in a model's training set. Unlike membership inference attacks (MIAs), which operate on individual samples, DI aggregates weak membership signals across many points to assess dataset-level usage.

Given a candidate set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the attacker seeks to determine whether $S \subseteq \mathcal{D}_{\text{train}}$. Even if each $\mathbf{x}_i$ provides only weak evidence on its own, their aggregated behavior can produce a strong dataset-level signal. The simplest decision rule is

$$A(S) = \mathbb{1}\left[\frac{1}{n} \sum_{i=1}^{n} \ell(M_\theta(\mathbf{x}_i), y_i) \leq \tau \right],$$

where τ is a threshold on the average loss.

DI is particularly relevant in copyright and data provenance contexts. Demonstrating that an entire collection, such as a licensed or curated dataset, was used during training may be more feasible and actionable than proving membership of any single sample. At the same time, DI can be misused to identify proprietary or confidential training datasets, highlighting its dual-use nature.

3. Related Works

In this chapter, we review works connected to the main contributions of this dissertation. We begin with applications of generative models for fast detector simulation in high-energy physics (HEP). We then discuss defenses against model stealing, and finally, membership and dataset inference for modern generative models.

3.1. Generative Models for Fast Simulation in HEP

Machine learning based generative models have recently become an important direction for accelerating detector simulations across CERN experiments. Early work concentrated on adversarial training, with Paganini et al. [171] introducing CaloGAN, a conditional convolutional GAN capable of producing calorimeter showers at orders of magnitude lower computational cost than Monte Carlo-based alternatives. Subsequent studies extended this idea to different calorimeter geometries, particle types, and jet-based observables [74, 134]. These efforts demonstrated that GANs can approximate complex shower patterns while offering a substantial speed-up. At the same time, they revealed characteristic limitations of adversarial training, including instability and the tendency of single generator models to collapse toward a restricted subset of the target distribution, limiting their ability to reflect the full diversity of calorimeter responses and to reproduce rare but physically meaningful modes.

In parallel, several alternative generative paradigms have been explored for fast HEP simulation. Autoencoder models [68, 120] exploit compact latent representations to reconstruct calorimeter data, offering stable training and efficient inference but typically yielding reduced sample diversity due to structural constraints in the latent space. Diffusion-based methods [63] have shown strong fidelity and robustness, though at the expense of iterative sampling procedures that remain computationally demanding for large-scale usage. Normalizing flow models [247] provide exact likelihood estimation and expressive invertible mappings, yet their computational cost increases rapidly with dimensionality and architectural depth. These works demonstrate a recurring trade-off between fidelity, diversity, and computational

efficiency, and illustrate the difficulty of achieving all three objectives simultaneously within a single generative architecture.

Fast Simulation of ZDC

These challenges are also clearly present in the context of the Zero Degree Calorimeter (ZDC). In our earlier preliminary work Dubiński et al. [82], we investigated the use of VAEs and GANs for ZDC simulation. That study showed that relatively simple generative models can produce plausible ZDC responses and significantly reduce computational cost by generating calorimeter observables directly rather than relying on detailed particle transport. In related work, Deja et al. [68] applied an end-to-end sinkhorn autoencoder to ZDC data. Their approach introduced a separate noise to latent mapping trained with the sinkhorn algorithm to align samples from a known noise distribution with the latent codes of real samples. By avoiding an explicit prior on the latent space, the model represented ZDC responses more naturally, without forcing them into a predefined latent geometry. This resulted in a better alignment between generated and real latent distributions, allowing the method to reproduce a wider range of ZDC shower patterns while retaining efficient inference. However, these approaches still struggled to reproduce the full multimodal structure of ZDC responses, leaving important variations in shower patterns insufficiently represented. Particle shower parameters, spatial patterns, and energy deposition vary across several physical regimes, and single models tend to focus on the most frequent behaviours, underrepresenting rare but physically important modes.

These observations provide the motivation for the contributions developed in this dissertation. Standard conditional GANs often fail to reproduce the natural variability of calorimeter data and tend to collapse to only a subset of possible outcomes. To address this, Chapter 4 introduces a method that enhances the diversity of generated samples in a way that remains consistent with the variability observed in the training data, improving the reliability of adversarial models for scientific use. As the strongly multimodal nature of calorimeter responses remains difficult to capture within a single generative model, Chapter 5 builds on the improvements introduced in Chapter 4 and presents a generative mixture-of-experts model. This approach distributes modelling capacity across several specialised generators, enabling a more faithful representation of distinct data regimes.

3.2. Defenses against Model Stealing

Model extraction attacks tra [12] pose a practical risk for deployed machine learning systems, as an adversary can recreate a high-fidelity surrogate by adaptively querying a target model. This threat has been extensively studied in the context of classifiers, language models, and, more recently, encoders. Existing defenses span several families, each addressing the problem from a different angle while exposing characteristic limitations.

A first line of work focuses on limiting or obfuscating model outputs. Techniques such as rounding confidence values, removing probability vectors, or returning only top-k predictions reduce the information leaked per query. While these approaches can modestly degrade the fidelity of stolen models, they also reduce the usefulness of the service for legitimate users Dziedzic et al. [85], Liu et al. [153]. Randomisation-based defenses inject noise into returned outputs Orekondy et al. [170], aiming to poison the attacker's training signal. Although this can slow naive attacks, adversaries frequently average repeated queries or adjust sampling strategies to cancel out the perturbations, and stronger noise levels tend to undermine the model's accuracy for benign use.

Provenance-oriented strategies, including watermarking Adi et al. [38], Jia et al. [126], Uchida et al. [228] and dataset inference–based fingerprinting Dziedzic et al. [86], Maini et al. [157], approach the problem differently: they allow the model owner to verify after the fact whether a released model is an extracted copy. These methods provide valuable evidence of theft but do not prevent extraction itself, and their robustness depends on how well the watermark or fingerprint transfers through the attacker's training process.

A complementary direction investigates behavioural defenses based on monitoring query patterns. Methods such as PRADA Orekondy et al. [169] assess whether incoming queries deviate from normal usage, for instance by probing atypical regions of the input space or by distributing queries across sybil identities to evade rate limits. Such methods can detect suspicious activity, but they do not directly interfere with the attacker's learning unless paired with an intervention mechanism. Moreover, adaptive attackers can reduce detection risk by shaping their queries to resemble benign workloads.

Defenses against Encoder Model Stealing

These challenges become more pronounced for encoder models, particularly in self-supervised learning. Unlike classifiers, encoders return high-dimensional representations that reveal substantially more information per query Dziedzic et al.

[85]. As a result, extraction attacks against encoders tend to be more effective and require fewer queries than attacks against supervised models. At the same time, many traditional defenses transfer poorly. Output perturbations, even when small, can significantly degrade the downstream utility of embeddings Liu et al. [153], and monitoring strategies lack clear indicators of suspicious behaviour when outputs are continuous vectors rather than discrete labels. Embedding-level watermarking and dataset-based signatures provide a partial remedy Dziedzic et al. [86], Maini et al. [157], but these mechanisms primarily enable post-hoc verification and do not impede the extraction process itself.

These limitations motivate the development of active defenses that intervene during the attack without compromising legitimate model utility. Chapter 6 introduces such an approach, which monitors coverage in the embedding space and selectively distorts outputs when query behaviour indicates systematic extraction. This mechanism preserves representation quality for honest users while degrading the attacker's ability to train a high-fidelity surrogate, addressing a gap not filled by existing passive or provenance-based defenses.

3.3. Training Data Identification in Generative Models

Membership Inference across Generative Models

Membership inference aims to determine whether a particular example was used to train a given model. Initially studied in the context of classifiers, Shokri et al. [211] demonstrated that models often assign higher confidence to training points than to unseen data, enabling adversaries to infer sample inclusion. Extending this idea to generative models, however, introduces several challenges. Unlike classifiers, generative models produce structured outputs such as images or sequences, and their loss functions, stochastic sampling, and lack of explicit decision boundaries complicate the detection of memorization. Moreover, well-trained generative models are designed to generalize across the data distribution, meaning that even when trained on a sample, they may not reproduce it exactly or assign it a clearly distinguishable score.

Despite these obstacles, membership inference attacks (MIAs) have been explored across several generative paradigms. In the context of GANs, Hayes et al. [111] proposed the LOGAN method, which leverages overfitting in the discriminator to detect whether particular examples influenced the generator. Subsequent work has

shown that under certain training regimes, GANs may indeed leak membership signals, particularly when they overfit or are trained on small datasets. For language models, Carlini et al. [47] demonstrated that membership can be inferred using perplexity gaps: models tend to assign lower perplexity to training sequences than to similar held-out ones. However, these signals are often weak and difficult to separate from natural variations in model output.

MIA Limitations

The effectiveness of MIAs tends to degrade as model scale increases or training regularization improves. Diffusion models present an especially challenging case for membership inference. These models generate outputs through iterative denoising, and their high capacity and diversity make direct memorization rare. Carlini et al. [48] and others have evaluated MIAs on diffusion models, finding that standard approaches often fail to achieve meaningful performance, especially when evaluated on web-scale models like Stable Diffusion. Even when some signal is present, identifying it with confidence at low false-positive rates remains difficult, limiting the practical utility of these methods. More recent attacks have improved performance under controlled conditions or on restricted datasets, but these do not translate well to real-world, large-scale settings. For example, evaluations on public diffusion checkpoints reveal that memorization is rare, and even successful extractions typically rely on atypical prompts or specially constructed input distributions.

Dataset Inference as an alterative to MIA

These findings highlight the fundamental limitations of single-sample MIAs in generative models. While in principle it is possible to detect training inclusion under narrow circumstances, these methods lack robustness and reliability in high-diversity settings. Models trained on massive datasets inherently dilute the influence of any individual sample, and the stochastic nature of generation further masks potential membership signals. Our own evaluations, discussed in Chapter 7, confirm that even under optimized attack conditions, MIAs fail to offer consistent or trustworthy identification of training data exposure.

In response to these challenges, recent work has shifted focus toward dataset-level inference, which asks whether an entire collection of samples, rather than a single point, was part of the training set. Pioneered by Maini et al. [157], this approach aggregates evidence across many related samples, enabling statistical assessments that are more reliable and interpretable. By analyzing consistent biases or performance differences over groups of inputs, dataset inference can detect subtle forms of training data influence that evade single-example analysis. In this dissertation, we

adopt and extend this line of work, proposing new methods for dataset inference in the context of diffusion models and image autoregressive models. These techniques, presented in Chapters 8 and 9, provide a more robust foundation for identifying whether a generative model was trained on protected or proprietary data, and represent a viable path forward for enforcing data provenance in high-capacity generative systems.

4. Selectively increasing the diversity of GAN-generated samples

Title	Selectively increasing the diversity of GAN-generated samples
Authors	Jan Dubiński, Kamil Deja, Sandro Wenzel, Przemysław Rokita, Tomasz Trzciński
Conference	International Conference on Neural Information Processing (ICONIP)
Year	2022

Preface

In the first part of this dissertation, we look at how generative models can be adapted for scientific use cases, focusing on simulations required in high-energy physics. In experiments at the Large Hadron Collider, researchers rely on detector simulations that are both computationally expensive and essential for interpreting results. Machine learning methods, and in particular Generative Adversarial Networks, have been proposed as an alternative, but their reliability remains a central challenge.

One of the most critical issues arises when applying conditional GANs, which are used to generate detector responses conditioned on the properties of incoming particles. While conditioning provides the necessary control over generated outputs, it also amplifies the mode collapse phenomenon. The generator tends to ignore the noise input and focuses only on the conditioning variables, producing a limited range of outcomes. Such behaviour prevents the model from capturing the full variability of collision data, reducing its usefulness for simulation tasks.

In this work, we propose a way to overcome this limitation. We first analyse how existing diversity-enhancing methods address mode collapse, noting that they often impose uniform diversity across all conditioning values. This assumption is rarely valid in real-world scenarios, where the variability of data strongly depends on experimental parameters. Building on this observation, we introduce Selective Diversity GAN (SDI-GAN), a method that regularises the generator in a way that enforces diversity only when it is supported by the training data. The approach scales the effect of diversity according to the variance observed for each conditioning input, allowing the model to reproduce realistic multimodal behaviour without sacrificing fidelity.

We demonstrate the advantages of this method in a synthetic benchmark, where the variability of conditional inputs can be explicitly controlled, and in a realistic use case of simulating the Zero Degree Calorimeter at the ALICE experiment. In both settings, the proposed approach increases the diversity of generated samples compared to standard conditional GANs, while maintaining their alignment with real measurements.

The following chapter builds directly on these results. Having shown that diversity can be introduced selectively into conditional GANs, we next investigate how combining multiple specialised generators can further improve the fidelity of simulations and better capture the multimodal distributions present in high-energy physics data.

Abstract

Generative Adversarial Networks (GANs) are powerful models able to synthesize data samples closely resembling the distribution of real data, yet the diversity of those generated samples is limited due to the so-called mode collapse phenomenon observed in GANs. Especially prone to mode collapse are conditional GANs, which tend to ignore the input noise vector and focus on the conditional information. Recent methods proposed to mitigate this limitation increase the diversity of generated samples, yet they reduce the performance of the models when similarity of samples is required. To address this shortcoming, we propose a novel method to selectively increase the diversity of GAN-generated samples. By adding a simple, yet effective regularization to the training loss function we encourage the generator to discover new data modes for inputs related to diverse outputs while generating consistent samples for the remaining ones. More precisely, we maximise the ratio of distances between generated images and input latent vectors scaling the effect according to the diversity of samples for a given conditional input. We show the superiority of our method in a synthetic benchmark as well as a real-life scenario of simulating data from the Zero Degree Calorimeter of ALICE experiment in LHC, CERN.

4.1. Introduction

Generative Adversarial Networks (GANs) [105] constitute a gold standard for synthesizing complex data distributions and they are, therefore, widely used across various applications, including data augmentation [164], image completion [17] or representation learning [15]. They are also employed in high energy physics experiments at the Large Hadron Collider (LHC) at CERN, where they allow to speed up the process of simulating particle collisions [68, 131, 172]. In this context, the generative models are used to generate samples of possible detectors' responses resulting from a collision of particles described with a series of physical parameters. For more controllable simulations, we can condition the generative models with additional parameters of the collision, using conditional GANs (cGANs) [163].

Although by conditioning GANs we obtain more context-dependent generations that are closer to the values observed in real experiments, these models are considered more vulnerable to the so-called mode collapse phenomenon [198] , observed as a tendency to generate a limited number of different outputs per each conditional prior. This, in turn, significantly reduces the effectiveness of employing GANs for particle collision simulations, as alignment of generated samples with the real data

distribution is fundamental for drawing correct conclusions from the performed experiments.

To address the above-mentioned limitations of cGANs, recent methods [152, 159, 252] attempt to increase the diversity of generated samples by modifying the associated cost function. However, they do not consider conditioning the diversity on the input conditioning values, assuming a uniform distribution of diversity across all of them. This assumption is rarely observed in practical applications, for instance in particle collision simulations at CERN diversity of generated samples highly depends on the set of conditioning variables.

In this work, we identify this shortcoming of existing models and propose a simple, yet effective method to selectively increase the diversity of GAN-generated samples, based on conditioning values. In principle, we introduce a regularization method that enforces GANs to follow diversity observed in the original dataset for a given conditional value. More exactly, we maximise the ratio of distances between latent vectors of generated images and inputs, scaling the effect accordingly to the diversity of samples corresponding to a given conditional input. Our approach, dubbed SDI-GAN, is readily applicable for conditional image synthesis models and does not require any modification of the baseline GAN architecture.

We evaluate our method on a challenging task of simulating data from the Zero Degree Calorimeter of the ALICE experiment in LHC, CERN. To better demonstrate the performance of our method we also include a synthetic dataset for 2D point generation. We compare our approach with competing methods and achieve superior results across all benchmarks.

The main contribution of this paper is a novel method for increasing the diversity of GAN-generated results for a selected subset of conditional input data while keeping the consistency of the results for the remaining conditional inputs.

4.2. Related work

4.2.1. Generative simulations:

The need for simulating complex processes exists across many scientific domains. In recent years, solutions based on generative machine learning models have been proposed as an alternative to existing methods in cosmology [191] and genetics [187]. However, one of the most profound applications for generative simulations is in the field of High Energy Physics, where machine learning models can be used as a resource-efficient alternative to classic Monte Carlo-based[124] approaches.

Recent attempts [90, 131, 172] leverage solutions based on Generative Adversar-

ial Networks [105] or Variational Autoencoders [135]. Although those methods offer considerable speed-up of the simulation process, they also suffer from the limitations of existing generative models. Controlling the diversity of simulated results while maintaining the high fidelity of the simulation is one of the challenges of using generative models for such applications.

4.2.2. Mode collapse and sample diversity in cGAN:

The authors of MS-GAN [159] address the mode collapse problem in cGANs by introducing mode-seeking loss. This additional regularization term added to the generator training function aims to improve generation diversity by maximizing the dissimilarity between two images generated from two different latent codes. During training, the generator tries to minimize the regularization term added to the loss function equal to the inverse of measured diversity.

DS-GAN [252] tries to tackle the problem with a similar approach. The main difference between the two methods is the fact that DS-GAN explicitly maximizes the measured diversity which is subtracted from the training loss of the generator.

In DivCo [152] the authors use contrastive learning to achieve diverse conditional image synthesis. They introduce a latent-augmented contrastive loss which encourages images generated from distant latent codes to be dissimilar and those generated from close latent codes to be similar. The similarity of images is measured using their latent representations extracted from the discriminator network.

Our approach shares a similar method of calculating the diversity of images with [159] and [252]. However, contrary to those approaches we do not base our measure of diversity on pixels of generated images. Instead we operate on image representation, similarly to [152]. Moreover, in principle, all previously described approaches do not account for different levels of variance of samples corresponding to different conditional inputs and instead maximize the diversity of the results generated for all conditional inputs.

4.3. Methodology

In this work we propose a novel selective diversity regularization method that improves alignment between generations from cGAN and data conditioned on available conditional values. For a dataset of images $\mathcal{X}$ with conditioning values $\mathcal{C}$ we want to learn a generator G that is able to produce realistic samples from the domain of $\mathcal{X}$ conditioned on $c \in \mathcal{C}$. Moreover, we want the synthesised images to be diverse or

similar to each other depending on the variance of samples in $\mathcal{X}$ that correspond to a given c.

Traditional conditional GANs are trained using adversarial loss. Given a condition vector $c \in \mathcal{C}$ and a k-dimensional latent code $z \sim \mathcal{N}_k(0,1)$ the generator G takes both c and z as input and produces an output image $\hat{x} = G(z,c)$. The image $\hat{x}$ should be indistinguishable from real data by a discriminator D. During training generator and discriminator play a min-max game, in which D learns to distinguish real data from samples synthesised by G while G tries to generate samples that are considered by D as real.

$$\mathcal{L}_{\text{adv}}(G,D) = \mathbb{E}_{x\sim\mathcal{X},c\sim\mathcal{C}}[\log D(x,c)] + \mathbb{E}_{c\sim\mathcal{C},z\sim\mathcal{N}(0,1)}[\log(1 - D(G(z,c),c)] \tag{4.1}$$

The adversarial loss function encourages the generator to produce realistic data, but as observed by [152] it does not directly promote the diversity of synthesised samples. To alleviate this problem Mao et al. [159] propose a regularization term that penalizes the low diversity of generated samples. More precisely, the introduced method maximizes the ratio of the distance between two images generated from two different latent codes z_1, z_2 and the same conditioning value c with respect to the distance between those latent codes. The proposed regularization term is added to the basic loss function from Eq. 4.1

$$\mathcal{L}_{\text{ms}} = \left(\frac{d_{\mathbf{I}}(G(c,\mathbf{z}_1), G(c,\mathbf{z}_2))}{d_{\mathbf{z}}(\mathbf{z}_1,\mathbf{z}_2)} \right)^{-1} \tag{4.2}$$

Although, this approach successfully forces the generator to produce dissimilar examples it does not account for different levels of sample diversity for different conditioning input c. To address this issue we propose a simple yet effective modification of the regularization term.

As a data preprocessing step, for each unique conditioning input $c \in \mathcal{C}$ we calculate the diversity of samples from the dataset $\mathcal{X}$ that correspond to c denoted as $\mathcal{X}_c$. We base our measure of diversity f_{div} on the variance of samples. As denoted in Eq. 4.3, for each set of images $\mathcal{X}_c$ we sum the standard deviation of pixel values with the same coordiantes for images in $\mathcal{X}_c$:

$$f_{div}(c) = \sum_{i,j} \sqrt{\frac{\sum_t (x_{ij}^t - \mu_{ij})^2}{|X_c|}} \tag{4.3}$$

where i and j are the pixel coordinates, t is the index of sample $x \in X_c$ and μ_{ij} is a mean value for a pixel ij from all samples from X_c. We normalize the obtained values of all sample diversity $f_{div}(c)$ to $< 0,1 >$.

To account for varying levels of sample diversity for each conditioning input c we multiply the regularization term introduced before by $f_{div}(c)$.

$$\mathcal{L}_{\text{div}} = f_{div}(c) * \left(\frac{d_{\mathbf{I}}(G(c, \mathbf{z}_1), G(c, \mathbf{z}_2))}{d_{\mathbf{z}}(\mathbf{z}_1, \mathbf{z}_2)} \right)^{-1} \tag{4.4}$$

This change forces the generator to better match the diversity observed in the original dataset with respect to conditional values. Intuitively, the generator produces more diverse images under conditions that allow for a higher variety of synthesised samples. At the same time, the generator is not forced to produce dissimilar results if the conditioning value corresponds to a set of similar images in the dataset. The overall objective of training SDI-GAN is the following:

$$\mathcal{L} = \mathcal{L}_{\text{adv}}(G, D) + \lambda_{\text{div}} \mathcal{L}_{\text{div}}(G) \tag{4.5}$$

where λ_{div} is a hyperparameter controlling the strength of the regularization.

Additionally, to better adapt our method to the task of applying cGANs as a fast simulation tool, we propose to base the distance d_g on the dissimilarity of latent representations of images rather than the dissimilarity of pixels. We measure the distance between two generated images by calculating the L_1 metric between their latent representations. We obtain those representations by treating the initial layers of the discriminator as an encoder E and extracting the latent features of the images from its penultimate layer during training.

$$e_{zc} = E(G(z, c)) \tag{4.6}$$

$$d_g(G(z_1, c), G(z_1, c)) = |e_{z_1 c}, e_{z_2 c}| \tag{4.7}$$

This change shifts the focus of the generator from the visual dissimilarity of images to the difference in their underlying characteristics extracted by the encoder.

4.4. Experiments

We evaluate our method on simulating data from the Zero Degree Calorimeter from the ALICE experiment in LHC, CERN. Additionally, we use a synthetic dataset as a benchmark. We compare our approach to a conditional DC-GAN [178], MS-GAN [159] and DivCo [152].

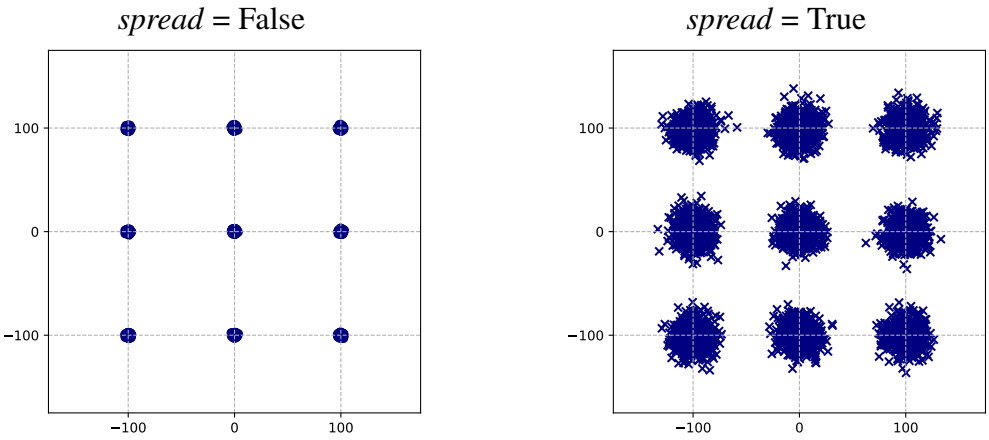


Figure 4.4.1. Synthetic dataset. For each class the generated points form a cluster. For point with *spread* = False the variance of each cluster is equal to 1 and for points with *spread* = True the variance of each cluster is equal to 100.

4.4.1. Synthetic dataset

To clearly demonstrate the effects of our method and its comparison to competing approaches we provide a synthetic dataset of 2D points generation. Each point is conditioned on a class label (from 1 to 9) and a binary variable *spread*. The position of the generated point depends mainly on its class label. However, points with *spread* = False are generated very close to each other, while points with *spread* = True are heavily dispersed. The dataset is presented in Fig. 4.4.1.

4.4.2. Zero Degree Calorimeter simulation

The task of simulating the response of the Zero Degree Calorimeter (ZDC) offers a challenging benchmark for generative models. The dataset consists of 295867 samples obtained from the GEANT4 [124] simulation tool. Each response is created by a single particle described with 9 attributes (mass, energy, charge, momenta, primary vertex).

During the simulation process, the particle is propagated through the detector for over 100 meters while simulation tools must account for all of its interactions with the detector's matter. The end result of the simulation is the energy deposited in the calorimeter's fibres, which are arranged in a grid with 44 × 44 size. We treat the calorimeter's response as a 1-channel image with 44 × 44 pixels, where pixel values are the number of photons deposited in a given fibre. To create the dataset the simulation was run multiple times for the same input particles. For that reason, multiple possible outcomes correspond to the same particle properties. We refer to this dataset as HEP.

Although the process that governs the propagation of the particles is non-deterministic by nature, the majority of particles create consistent ZDC responses. However, a

Table 4.4.1. Results comparison on the synthetic and HEP datasets. Our solution outperforms competing approaches by generating data with variance close to the real data for both types of conditioning input in the synthetic case. We achieve the lowest Wasserstein distance between test and generated coordinates for the diverse points without any significant trade-off for the consistent points. The performance improvement introduced by SDI-GAN is further evaluated on the real-life HEP dataset, where SDI-GAN achieves the lowest Wasserstein distance between channels calculated from original and generated data.

	Synthetic dataset				HEP
	Variance		Wassesrstein ↓		Wassesrstein ↓
Spread	0	1	0	1	-
Real	1	100	-	-	-
DC-GAN	**0.2**	2.3	**0.7**	7.4	7.6
MS-GAN	115.5	280,7	9.0	6.7	21.7
DivCo	163.3	3.7	1.1	7.0	14.3
SDI-GAN (ours)	3.3	**127.4**	1.2	**2.1**	**4.5**

subset of particles produces highly diverse results and allows for multiple possible calorimeter responses. In Fig. 4.4.2 we present sampled simulations for two different conditional values. In the first row, we depict the calorimeter response for a high-energy proton, while for the second the input particle is a neutron, which has no electric charge, therefore responses observed in the calorimeter are much more consistent.

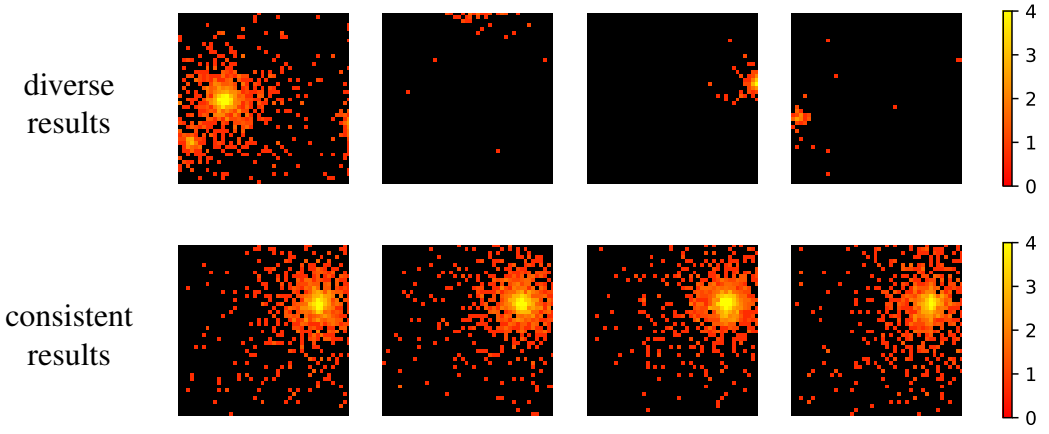


Figure 4.4.2. Examples of ZDC calorimeter responses. We show 4 possible outputs of the simulation generated for 2 distinct particles.

4.4.3. Results

We present the results of our experiments on the synthetic dataset using both qualitative and quantitative comparisons. As shown in Fig. 4.4.3 our method generates samples that are visually most similar to the real data. To confirm this observation we calculate the mean variance for all classes for points with *spread* = *0* and *spread* = *1*. The results in Tab. 4.4.1 show the superiority of our method in this scenario. DC-GAN is able to produce results with variance close to real data for conditional inputs with *spread* = 0, but fails to generate diverse results. MS-GAN properly reflect the diversity of possible results for conditional inputs with *spread* = 1, but does not generate consistent results. Although samples created by DivCo have different variance depending on the conditioning input, the data distribution generated under condition *spread* = 1 is distorted and does not match the real data, as presented in Fig. 4.4.3. Our approach is able to produce both diverse and similar results depending on the conditional information.

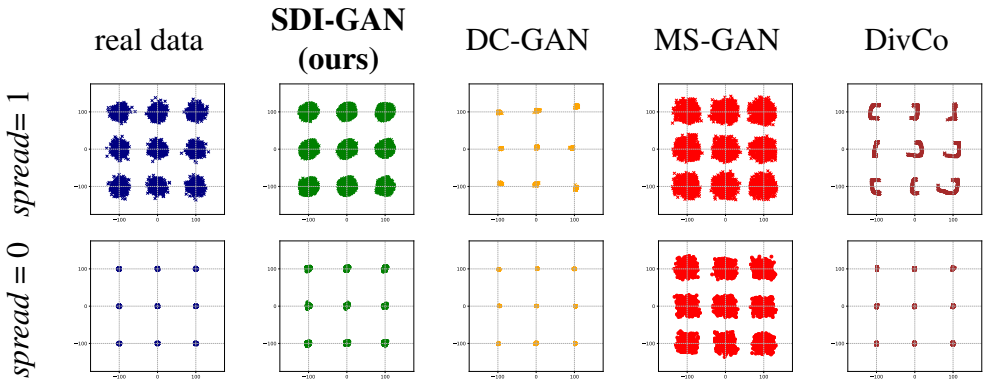


Figure 4.4.3. Comparison of generated results for the synthetic dataset. Contrary to competing methods, our approach is able to properly synthesise both diverse and similar samples depending on the conditioning variable *spread*.

The most common method for evaluating GANs on real datasets utilizes Frechet Inception Distance (FID) [115]. However, for the HEP dataset, we propose a domain-specific evaluation scheme that better measures the quality of the simulation. Following the calorimeter's specification [69] we base our evaluation procedure on 5 channels calculated from the pixels of generated images. These channels reflect the physical properties of simulated collision and are used for analysing the output of the calorimeter. To measure the quality of the simulation we compare the distribution of channels for the original and generated data using Wasserstein distance [224].

As presented in Tab. 4.4.1, our approach outperforms other solutions on the HEP datasets. In Fig. 4.4.4 we demonstrate that our method is able to generate diverse results for a specific subset of particles while keeping consistent responses

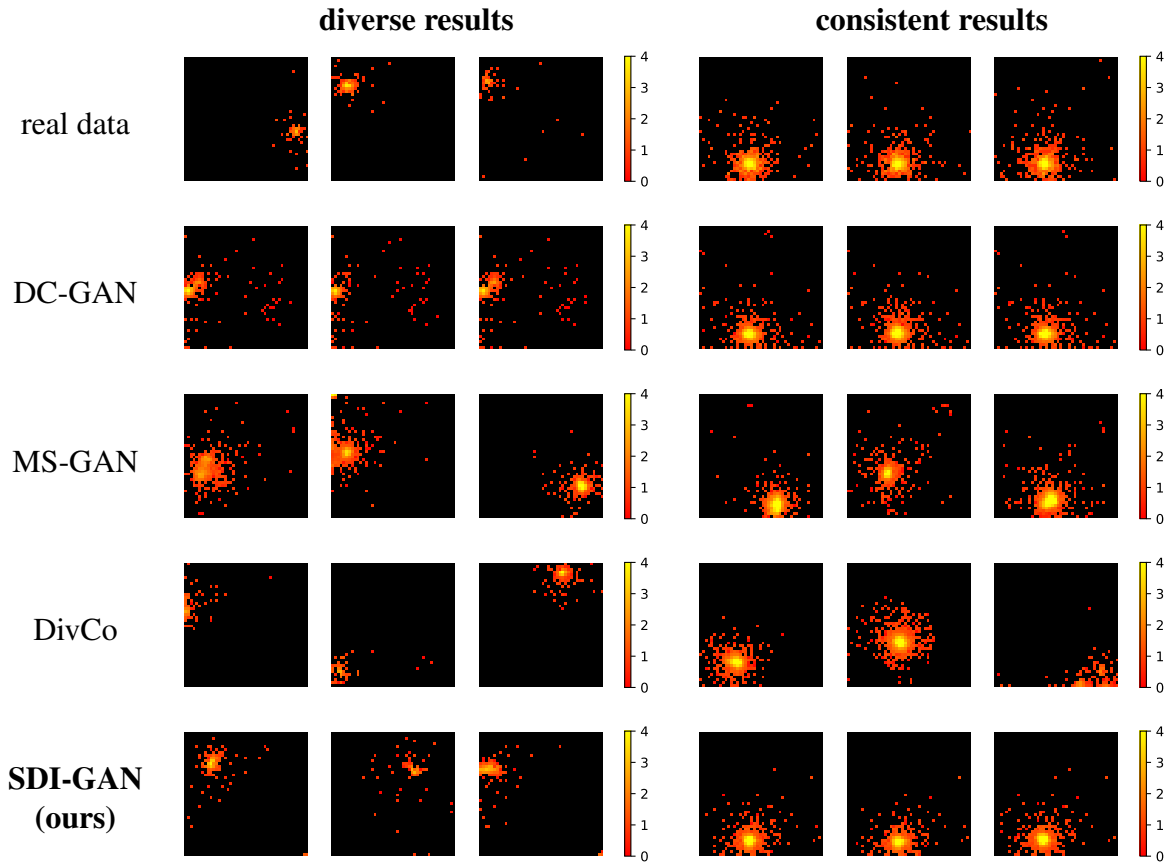


Figure 4.4.4. Examples of calorimeters response simulations with different methods. DC-GAN works well for particles with consistent responses but fails to generate diverse outcomes when needed. Although MS-GAN and DivCo successfully increase the diversity of generated samples those models do not distinguish between particles that should produce diverse or consistent showers. Our method is able to generate diverse results while producing consistent responses for appropriate particles.

for the remaining conditional inputs. The positive impact of this approach on the distribution of the generated samples is further confirmed by Fig. 4.4.5 where we compare channel distribution for SDI-GAN and competing approaches for 2 selected channels. Our method increases the fidelity of the simulation by smoothing the distribution of generated responses and covering the whole range of possible outputs.

The additional regularization term for training of SDI-GAN does not influence the inference speed of the model. In our initial experiments, we observe a speed-up of simulations of two orders of magnitude when compared to the standard Monte-Carlo approach. With SDI-GAN this computation boost is observed without degradation in simulation quality. We leave the detailed analysis of this performance gain and the influence of fast simulations on physical experiments for future work.

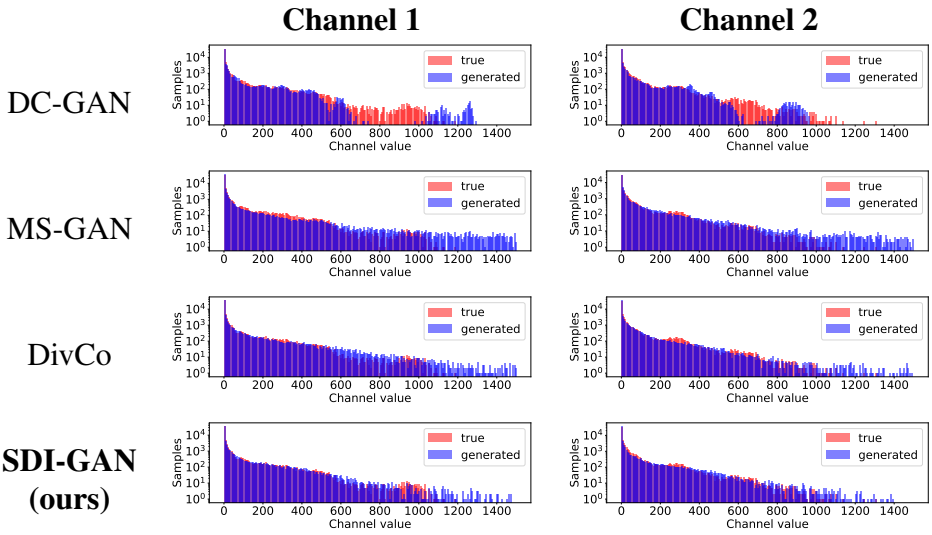


Figure 4.4.5. Comparison of channel values distribution for selected channels. Our method decreases the differences between the distribution of original and generated data and smooths the distribution of the synthesised results. In the case of SDI-GAN the increased diversity of generated samples does not harm the fidelity of the simulation, contrary to competing approaches.

4.5. Conclusions

In this work we introduce a simple, yet effective modification of the loss function for conditional generative adversarial networks. Our solution enforces increased sample diversity for a subset of conditional data without affecting samples that are characterised by conditional values associated with consistent responses.

We show that our solution outperforms other comparable approaches on the synthetic benchmark and the challenging practical dataset of calorimeter response simulations in the ALICE experiment at CERN.

5. ExpertSim: Fast Particle Detector Simulation Using Mixture-of-Generative-Experts

Title	ExpertSim: Fast Particle Detector Simulation Using Mixture-of-Generative-Experts
Authors	Patryk Będkowski, Jan Dubiński, Filip Szatkowski, Kamil Deja, Przemysław Rokita, Tomasz Trzciński
Conference	European Conference on Artificial Intelligence (ECAI)
Year	2025

Preface

In the previous chapter, we addressed the problem of limited diversity in conditional GANs and proposed a solution that selectively increases variability of generated samples while maintaining their fidelity. This allowed the model to reproduce more realistic responses of the Zero Degree Calorimeter at the ALICE experiment. However, despite these improvements, relying on a single generator still proved insufficient for capturing the full multimodal structure of high-energy physics data. The complex interactions of particles in the detector give rise to distributions that are difficult to approximate with one generative model, even when diversity is explicitly encouraged.

To overcome this limitation, we explore the idea of decomposing the simulation task into several specialised components. Instead of training a single generator to cover the entire space of particle responses, we design a mixture-of-experts framework where each expert focuses on a different part of the distribution. A trainable router is introduced to assign incoming conditions to the most suitable expert, enabling the overall system to better capture the heterogeneity of detector responses. In our implementation, each expert is based on the selectively diversified GAN introduced in the previous chapter, ensuring that both fidelity and diversity are preserved at the level of individual generators.

We evaluate this approach on the challenging task of simulating the Zero Degree Calorimeter. Through extensive experiments, we show that ExpertSim outperforms single-model baselines as well as other generative solutions. The system achieves higher accuracy in reproducing experimental distributions while also providing significant speed-up compared to traditional Monte Carlo simulations. Beyond accuracy, we analyse the behaviour of individual experts, observing that they naturally specialise in different modes of the data, which supports the intuition behind the mixture-of-experts design.

By moving from a single generative model to a modular architecture of multiple experts, this chapter demonstrates how fidelity and scalability of scientific simulations can be further improved. In the subsequent part of the dissertation, the focus shifts from building reliable generative models to safeguarding the intellectual value of machine learning systems, beginning with the protection of encoder models against unauthorized replication.

Abstract

Simulating detector responses is a crucial part of understanding the inner workings of particle collisions in the Large Hadron Collider at CERN. Such simulations are currently performed with statistical Monte Carlo methods, which are computationally expensive and put a significant strain on CERN's computational grid. Therefore, recent proposals advocate for generative machine learning methods to enable more efficient simulations. However, the distribution of the data varies significantly across the simulations, which is hard to capture with out-of-the-box methods. In this study, we present ExpertSim - a deep learning simulation approach tailored for the Zero Degree Calorimeter in the ALICE experiment. Our method utilizes a Mixture-of-Generative-Experts architecture, where each expert specializes in simulating a different subset of the data. This allows for a more precise and efficient generation process, as each expert focuses on a specific aspect of the calorimeter response. ExpertSim not only improves accuracy, but also provides a significant speedup compared to the traditional Monte-Carlo methods, offering a promising solution for high-efficiency detector simulations in particle physics experiments at CERN. We make the code available at `https://github.com/patrick-bedkowski/expertsim-mix-of-generative-experts`.

5.1. Introduction

ALICE (A Large Ion Collider Experiment) is one of the four major detectors located at the Large Hadron Collider (LHC) at CERN. One of its main goals is to replicate and study the intense conditions that existed in the early universe shortly after the Big Bang. To understand the physics behind this state, LHC collides particles accelerated almost to the speed of light, and gather the data describing the effect of those collisions. However, to validate hypotheses, collected data needs to be statistically compared with theoretical simulations. Such simulations are extremely computationally expensive, as existing approaches utilize statistical Monte-Carlo methods to model the physical interactions between particles. While these methods yield high-fidelity outcomes, they are also associated with high computational demands. In 2023, over 540 000 CPU devices [50] were engaged in the computations of ALICE experiments, marking a demand for developing more efficient simulation techniques.

The high computational cost of the simulations calls for alternative, lightweight solutions to accelerate the experimentation cycle. Throughout recent years, generative deep neural networks have emerged as a promising alternative, that does

not require repetitive calculations present in the Monte Carlo approach. Thanks to that, recent works show over tens or hundred times faster simulations [14, 68], with possible straightforward parallelization on high performance GPUs. However, applying standard machine learning algorithms to this domain requires significant modifications, as the data distributions in physical experiments do not resemble the traditional distributions encountered *e.g.* in computer vision. Therefore, maintaining the high fidelity and diversity of the simulations with generative models remains a challenging task due to the high variance in the data distribution.

This is especially true for the most computationally expensive simulation of the Zero Degree Calorimeter (ZDC). This device is designed to measure proton energy in heavy ion collisions and plays a crucial role in monitoring the centrality of particle collisions at the ALICE experiment. By design, most of the ZDC responses fall into one of the three general groups that exhibit significantly different properties. Recent techniques in this domain tackled the problem of simulating ZDC's responses [81, 82] aiming to increase the simulation quality without compromising the speed. However, modeling multiple distinct distributions with a single model requires high model capacity and complexity, which is contrary to the goal of improving the simulation speed. In this study, we propose to solve this problem through a Mixture-of-Experts (MoE) approach that allows us to maintain the fast generation speed without compromising the simulation quality.

MoE layers [207] are composed of multiple expert modules and a gating network. The gating network determines the information flow in the network by selecting which experts are utilized for a given input. MoE leverages the fact that different inputs to the neural networks might require different treatments, which allows for more efficient processing. Inspired by MoE methods, we propose to utilize a similar approach for the ZDC data simulation. To that end, we introduce a single generative MoE model that dynamically selects small, specialized expert for a given input to enable fast processing with high fidelity. In our approach, each expert is based on the deep convolutional generative adversarial network (GAN) [105]. To increase the diversity of generated samples within each expert, we use SDI-GAN model [81] architecture that adds a diversity regularization term to the standard GAN objective. Experts are assigned specific samples to process by an efficient routing network, which we train leveraging specific physical characteristics of the samples. Following [31], we also employ a regularization method focused on minimizing the difference in intensities between real and generated calorimeter responses, which improves the quality of the simulations. Finally, we introduce an auxiliary regressor which increases the model capabilities to learn accurate spatial features of the simulation data.

ExpertSim significantly outperforms all existing approaches and achieves the highest simulation fidelity while maintaining nearly the same inference time as single-model methods. We improve over the previous state-of-the-art by more than 15% and achieve a Wasserstein distance of 1.70 between the distributions of real and generated data. Our approach offers a promising avenue towards providing fast and reliable simulations in a challenging CERN environment.

We summarise the contributions of this work as follows:

- We propose ExpertSim, a novel, MoE-based generative model architecture simulating Zero Degree Calorimeter responses in the ALICE experiment at CERN.
- We introduce a router training scheme that leverages the physical properties of the data, which results in high expert specialization and precise routing decisions.
- We evaluate our simulation model in real-life scenarios and compare it with single model approaches, demonstrating over 15% improvement upon the previous state-of-the-art without compromising the generation speed.

5.2. Related Work

5.2.1. Generative Models for Fast Simulation in CERN

Throughout recent years, the use of Generative AI for various CERN simulations has shown the versatility of these methods. The majority of works focus on the usage of generative adversarial networks [74, 90, 134, 171]. Alternative approaches include generative autoencoders [68, 120, 188] or more recently diffusion models [63, 136] and normalizing flows [246–248].

For the particular case of ZDC, there are several approaches based on GANs [81, 82] or Sinkhorn autoencoder [68]. In particular, [82] employs generative machine learning algorithms for the task of simulating a Neutron ZDC. They propose a solution that utilizes variational autoencoders [135] and generative adversarial networks [105]. By expanding the GAN architecture with an additional regularization, the authors significantly increase the simulation speed by two orders of magnitude while maintaining the high simulation fidelity.

In this vein of research, to match the diversity of the simulation observed in the training data, in [81] authors propose a GAN model dubbed SDI-GAN. The method forces the generator to discover new data modes for inputs related to diverse outputs while generating consistent samples for the remaining ones.

5.2.2. Mixture-of-Experts

Many methods have been crafted to reduce the computational cost of training and inference of deep neural networks, including network architectures designed for efficiency [219], knowledge distillation [117], pruning and quantization [147]. Additionally, multiple works also try to reduce resource usage through the introduction of conditional computations, leveraging the fact that different data points might require different conditional complexity to save compute on the easy samples. Some works adapt the compute by selecting the subset of filters [114, 149], features [99, 232] or tokens [186] processed in each layer. Works such as [148, 167] introduce sparsity while training the model. Multiple methods [39, 67, 89, 106, 240] allow the model to adapt its depth to the input example via skipping layers [240], competing halting scores [39, 106], introducing recursive computations [67, 89] or attaching early exit classifiers [30, 160, 200]. However, perhaps the most widely spread dynamic network architecture is Mixture-of-Experts (MoE).

MoE was introduced as an efficient way to further increase the capacity of deep neural networks applied in NLP, initially in LSTM models [207], and later in Transformers [144]. Since then, they have also been adapted to computer vision [66, 186]. MoE layers have gained significant popularity primarily due to their excellent scaling properties [58, 76]. Nonetheless, training such models is challenging, primarily because gating decisions must be discrete to ensure sparse expert selection. Various methods of training were proposed, some of which include reinforcement learning [10], weighting the expert output by the probability to allow computation of the gradient of the router [207], or using the Sinkhorn algorithm [58]. Some of those approaches also suffer from the possibility of load imbalance and therefore require auxiliary losses or alternative expert selection methods [95, 266]. Interestingly, in many cases, fixed routing functions perform similarly to trainable routers [189], suggesting that current solutions are largely suboptimal. MoE models can also be derived from pre-trained dense models by splitting the model weights into experts and independently training the routers for each layer [263, 267], which avoids most of the problems present in end-to-end training.

Other works explore utilizing experts for different data subsets and introduce domain experts [108, 140], task-specialized [57] or finely-grained experts [64] designed to maximize the specialization. In the context of generative models, similar strategies utilizing experts to cover different data subsets were developed for GANs [173] and multi-modal autoencoders [210].

The key novelty of ExpertSim with respect to existing approaches lies in leveraging and extending MoE ideas to address a real-world problem of fast, high-fidelity simulations of particle collision. Unlike standard transformer-based MoEs, where

increasing parameter count is a primary goal, our motivation for using multiple experts is to enable the model to capture the diverse range of detector responses observed in practice, which cannot be effectively modeled by a single generative networks. Furthermore, the high-energy physics domain's need for fast simulations makes GANs the preferred modeling backbone for each expert. This is a significant distinction from conventional MoEs, where experts are typically implemented as multilayer perceptrons.

5.3. Zero Degree Calorimeter Simulation

The Zero Degree Calorimeter (ZDC) [8] includes a Proton (ZP) and a Neutron (ZN) device for recording energy from non-interacting nuclei in collisions. The role of this device is to provide information about the centrality of particle collision undergoing inside the ALICE experiment. The higher the energy of non-interacting nuclei measured by ZDC, the lower the centrality of the recorded collisions. Both ZP and ZN are quartz-fiber calorimeters, utilizing the principle of detecting Cherenkov light produced by charged particles of the shower in silica optical fibers [8]. Every alternate fiber is directed towards a photomultiplier (PMTc), with the rest of the fibers being grouped into bundles that lead to four distinct photomultipliers (PMT1 to PMT4). The design allows for precise measurement of particle energies in heavy ion collisions at the CERN LHC, with ZP and ZN capturing distinct particle types.

As the silica fibers are arranged in a n by n grid, we treat the response of the ZDC as a 1-channel $n \times n$ image with pixel values equal to the number of photons deposited in the respective optical fiber. Each simulation example is, therefore, an image coming from a ZP or ZN device, which is referred to as a response of the experiment associated with a vector of variables, referred to as conditional data. Conditional data compromises of 9 variables: energy, mass, charge, three spatial position coordinates, and three momentum coordinates. The resolution of ZP and ZN responses is 56×30 and 44×44 respectively.

5.4. Method

Simulating the responses of the ZDC at ALICE CERN requires capturing the complex relationships between particle properties and calorimeter outputs. These relationships are often too intricate for a single generative model. To address this challenge, we propose ExpertSim, a framework comprising a mixture of generative experts presented in Fig. 5.3.1.

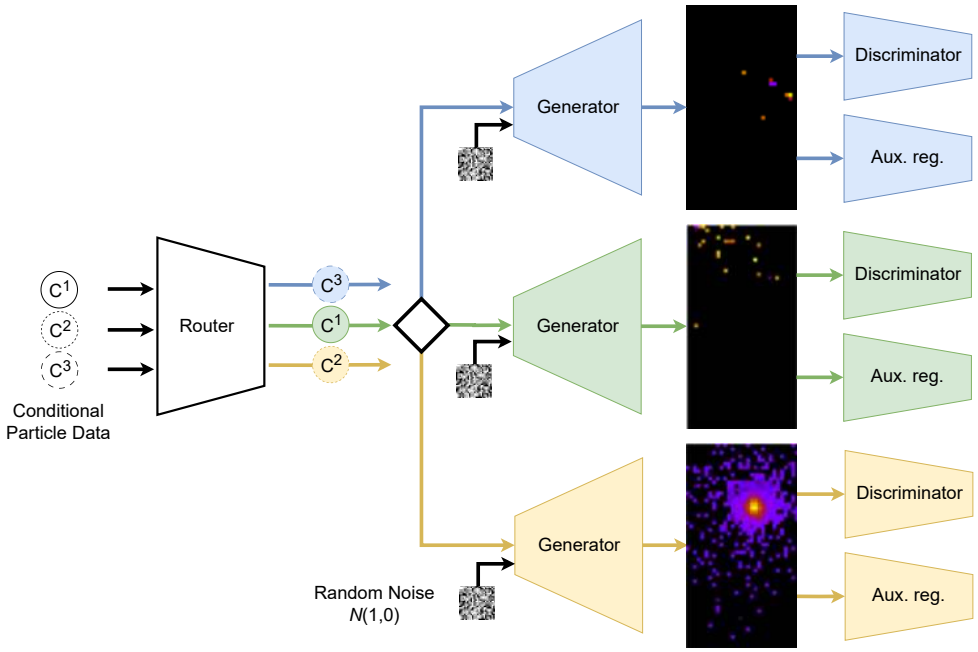


Figure 5.3.1. Overview of ExpertSim architecture. Our approach leverages the Mixture-of-Experts paradigm to dynamically route particle data to one of the three specialized generators, which enables us to use a small-sized generator while maintaining the generation quality.

Specifically, ExpertSim is composed of a router network and three generative expert models. The router is a fully connected neural network that takes conditional particle data as input. We train the router to optimize for specialization, guiding it to allocate particles that produce similar outputs to the same generative expert.

Each generative expert in our framework is implemented as a generative adversarial network (GAN) consisting of a Generator, an Auxiliary Regressor, and a Discriminator. The Generator takes random noise from a normal distribution, $N(0,1)$, conditioned on particle data to synthesize realistic images. The Auxiliary Regressor enhances spatial learning by estimating the coordinates of the image's high-intensity region, aiding in the geometric accuracy of generated outputs. The Discriminator, also conditioned on particle data, distinguishes between real and synthetic images, refining the Generator's outputs through adversarial training. This multi-expert setup enables ExpertSim to more effectively capture the nuanced response patterns of the ZDC, offering improved accuracy and specialization over traditional single-model generative approaches.

5.4.1. Expert Generative Adversarial Networks

As described above, we employ a Deep Convolutional generative adversarial network (DCGAN) [105] as a single generative expert, which consists of a generator,

$G(z,c)$, and a discriminator, $D(x,c)$. The generator synthesizes an image x starting from random noise z, while the discriminator attempts to distinguish between real and generated images. Both the generator and discriminator are conditioned on particle data c and trained in an adversarial manner. After training, the generator is used to create new calorimeter response images. Formally, given a conditional input c and a random noise z sampled from k independent standard normal distributions, $z \sim \mathcal{N}(0,1)^k$, the generator $G(z,c)$ produces an output image $\hat{x} = G(z,c)$.

Diversity Regularization SDI-GAN [81] introduces a regularization approach to encourage diversity by minimizing the inverted ratio of the $L1$ distance between two generated images d_I, produced from distinct latent codes z_1 and z_2, and the $L1$ distance between these codes d_z, under the same conditioning vector c. The diversity metric relies on the pixel variance observed in the original dataset to scale the regularization term, accounting for varying levels of diversity across different conditioning inputs. For each unique conditioning value c, the variance of pixel values across samples is calculated during the preprocessing stage. The diversity values for each sample are normalized to a $[0,1]$ range by dividing by the dataset size. This measure is then scaled by the regularization term λ_{div}, adjusting its strength during training:

$$L_{div} = \sum_{i,j} \sqrt{\frac{\sum_t (x_{ij}^t - \mu_{ij})^2}{|X|}} \times \left(\frac{d_I(G(c,z_1), G(c,z_2))}{d_z(z_1,z_2)} \right)^{-1}, \tag{5.1}$$

where i and j represent pixel coordinates, t is the index for a sample $x \in \chi$, and μ_{ij} is the mean pixel value for coordinates (i,j).

Intensity Regularization While SDI-GAN performs well on filtered data, it can struggle with variations in Cherenkov light intensity across distributions. To address this, we introduce a regularization term based on an intensity measure f_{in} derived from the original dataset. During preprocessing, the intensity measure for each conditioning vector c is calculated as the sum of pixel values in the corresponding image x from χ:

$$f_{in}(x) = \sum_{i,j} x_{ij}, \tag{5.2}$$

where i and j denote pixel coordinates. The intensity difference between a generated image $\hat{x}$ and the corresponding real image x is computed with Mean Absolute Error (MAE). The loss is also weighted by a constant λ_{in} to control its impact:

$$L_{in} = |f_{in}(x_c) - f_{in}(\hat{x}_c)|. \tag{5.3}$$

Auxiliary Regressor A key aspect of calorimeter response is identifying the location of the shower center, where pixel intensities are the highest. To help the individual experts better capture these geometric properties, we integrate an auxiliary regressor that works alongside the main model. This regressor is tasked with pinpointing the 2D coordinates of the collision center. During preprocessing, these coordinates are determined for all training samples and used as targets for the regressor. The auxiliary loss is calculated as the mean squared error (MSE) between the predicted coordinates $(\hat{k_i}, \hat{l_i})$ and the actual coordinates (k_i, l_i) of the peak-intensity pixel in the image x_i. This loss is then added to the generator's overall loss function to improve its geometric accuracy, with its influence controlled by the parameter λ_{aux}:

$$L_{aux} = \frac{1}{N} \sum_{i=1}^{N} \left[(\hat{k_i} - k_i)^2 + (\hat{l_i} - l_i)^2 \right]. \tag{5.4}$$

Final GAN Expert Training Objective The overall training loss incorporates the traditional GAN loss along with the additional terms introduced above, resulting in the following composite objective:

$$\begin{aligned} L(G,D) = L_{adv}(G,D) + \lambda_{div} L_{div}(G) \\ + \lambda_{in} L_{in}(G) + \lambda_{aux} L_{aux}. \end{aligned} \tag{5.5}$$

This final loss function blends adversarial loss with diversity, intensity, and auxiliary regression components, each scaled by their respective weights λ_{div}, λ_{in}, and λ_{aux}, to balance the different training objectives.

To ensure a fair comparison between methods, we train all models using identical parameter settings, including the learning rates of the generator, discriminator, and auxiliary regressor, as well as consistent loss strengths across all models.

5.4.2. Router

The router model is implemented as a multilayer, fully connected neural network that takes particle properties as input and returns an assignment to one of three generative experts. The purpose of the router is to dynamically allocate inputs to the most suitable expert based on the underlying characteristics of the data. By leveraging conditional information on particle properties, the router determines which expert should generate the calorimeter response for each input. This process leads to specialization of experts on different subsets of input data. To achieve this task, we train the router to balance the workload among experts while also promoting their specialization. We achieve this goal with two losses described in the following

sections. To balance the load between experts we minimize the entropy-based expert utilization loss. At the same time, to promote their specialization, we introduce an expert differentiation loss that maximizes diversity between the mean outputs of all experts. The combined use of expert utilization and differentiation losses ensures that the router effectively balances task distribution while fostering distinctive behaviors among experts.

Expert Utilization Loss The expert utilization entropy loss L_{util} is calculated as follows:

$$L_{\text{util}} = -\sum_{i=1}^{N} \bar{p}_i \log(\bar{p}_i + \epsilon), \tag{5.6}$$

where N is the number of experts, $\bar{p}_i$ is the average gating probability for expert i, computed over the batch, and ϵ is a small constant added for numerical stability.

This loss encourages the router to distribute samples evenly across all experts. By maximizing the entropy, the router avoids over-reliance on any single expert, ensuring a balanced load that enhances the model's ability to specialize and handle diverse inputs effectively.

Expert Differentiation Loss The differentiation loss L_{diff} is calculated based on the mean intensities of images generated for each expert. The intensity $f_{\text{in}}(x)$ for an image x is defined as the sum of pixel values across the image as in the Equation 5.2. Mean intensities $\bar{f}_{\text{in}}^{(i)}$ for the generated images corresponding to each expert i are then used to compute the differentiation loss L_{diff}:

$$L_{\text{diff}} = -\sum_{i=1}^{N} \sum_{j=i+1}^{N} (\bar{f}_{\text{in}}^{(i)} - \bar{f}_{\text{in}}^{(j)})^2. \tag{5.7}$$

The purpose of the differentiation loss is to encourage diversity among the experts by penalizing cases where the mean intensities of the outputs are similar. The router fosters specialization among experts by maximizing these differences, enhancing its capacity to manage a diverse range of input variations.

Final Router Training Objective The final router training objective L_{router} combines the sum of the generator-discriminator losses across all experts with regularization terms for expert utilization and differentiation:

$$L_{\text{router}} = \sum_{i=1}^{N} L(G_i, D_i) + \lambda_{\text{util}} L_{\text{util}} + \lambda_{\text{diff}} L_{\text{diff}}, \tag{5.8}$$

Table 5.4.1. Comparison of mean WS metric across 4 quartiles of different intensity ranges. ExpertSim offers better WS across all quartiles when compared with standard GAN and SDI-GAN with intensity regularization (IR) and auxiliary regressor (AR). Boldface indicates the best-performing result, while underlining denotes the second-best.

Model	**Proton**				**Neutron**			
	1st Qrt	**2nd Qrt**	**3rd Qrt**	**4th Qrt**	**1st Qrt**	**2nd Qrt**	**3rd Qrt**	**4th Qrt**
GAN	4.07	4.20	3.76	18.13	4.82	5.37	7.75	10.39
SDI-GAN	4.37	5.07	4.78	15.08	4.67	5.18	7.52	9.67
SDI-GAN + IR	<u>3.33</u>	<u>3.42</u>	<u>3.47</u>	<u>9.92</u>	<u>4.48</u>	<u>4.91</u>	6.69	8.29
SDI-GAN + IR + AR	3.52	4.20	4.48	12.72	4.49	5.02	<u>6.34</u>	<u>7.05</u>
ExpertSim	**1.06**	**1.16**	**1.27**	**7.06**	**2.20**	**4.02**	**5.62**	**5.69**

Table 5.4.2. Comparison of mean WS metric across five runs. Our method outperforms standard GAN and SDI-GAN with intensity regularization (IR) and auxiliary regressor (AR). We follow Dubiński et al. [82] and train our method on data generated for the ZDC using a Monte Carlo-based simulation, which can be treated as the upper bound for our solution. For the Monte Carlo simulation, we calculate the Wasserstein metrics between two runs.

Model	**Proton**		**Neutron**	
	WS↓	**Std. Dev.**	**WS↓**	**Std. Dev.**
GAN	2.47	0.010	2.42	0.041
SDI-GAN	2.35	0.027	2.31	0.063
SDI-GAN + IR	2.29	<u>0.009</u>	2.04	<u>0.018</u>
SDI-GAN + IR + AR	<u>2.07</u>	0.025	<u>1.89</u>	0.029
ExpertSim	**1.59**	**0.008**	**1.34**	**0.010**
Monte-Carlo	0.47	0.033	0.34	0.014

where $L(G_i, D_i)$ represents the generator-discriminator loss for each expert i, λ_{util} is a regularization weight for the expert utilization entropy loss L_{util} which promotes balanced routing, and λ_{diff} is a regularization weight for the expert differentiation loss L_{diff} which encourages diversity among expert outputs.

5.5. Experiments

The proton and neutron datasets employed in this work consist of 344 thousand and 400 thousand samples, respectively, with the validation method incorporating an 80:20 train-test split ratio. Our evaluation methodology is founded on the analysis of five distinct channels outlined in the calorimeter's specifications [70]. Those five

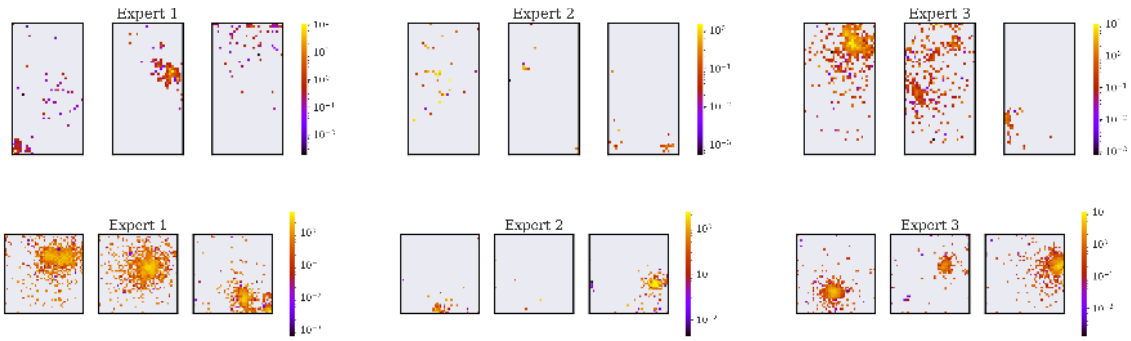


Figure 5.5.1. Examples of images generated by each expert. Top: ZP, Bottom: ZN. Please note the differing color bar scale for better visibility.

channel values correspond to the number of photons collected by each of the five distinct photomultipliers (PMTc1, PMT1-4) in ZP and ZN. We employ the standard 1st Wasserstein distance metric [224] to assess the fidelity of the simulations across all channels by comparing the generations to the ground truth simulations from the test set. The code to reproduce our experiments is available in the repository[1].

5.5.1. Results

We further compare the ExpertSim model with previous single-generator methods, GAN and SDI-GAN variants, using the Wasserstein distance metric (WS). As shown in Tab. 5.4.2, incorporating diversity regularization in SDI-GAN improves the metric compared to the standard GAN. Further regularization on intensity and the use of an auxiliary regressor provide the best performance among the single-generator models. However, ExpertSim provides a significant reduction in WS distance, achieving the best generation quality across all approaches. The inference time on GPU hardware is nearly identical to that of single-generator models, with the overhead introduced by the router network equal to 2%. Moreover, we achieve a speedup of more than an order of magnitude compared to the Monte Carlo simulation. We include more details on this comparison in Tab. 5.5.4.

In Fig. 5.5.2, we present the visual results for channels 4 and 5, as they contain the majority of the relevant information. The GAN and SDI-GAN models have visible problems with underproducing high-energy responses. The implementation of additional regularization and auxiliary regressor positively influences better alignment to true distribution but tends to oversample the high-energy responses. ExpertSim model better aligns with the true distribution for both low and high-value channels.

[1] `https://github.com/patrick-bedkowski/expertsim-mix-of-generative-experts`

We further evaluate the models' performance across different intensity ranges by dividing the entire range of sorted ascending intensities from the original dataset into four quartiles. For each quartile, we calculate the WS metric to assess how well the models perform within these specific intensity ranges and report the results in Tab. 5.4.1. Our ExpertSim method outperforms all other models by achieving the lowest WS distance across all intensity ranges.

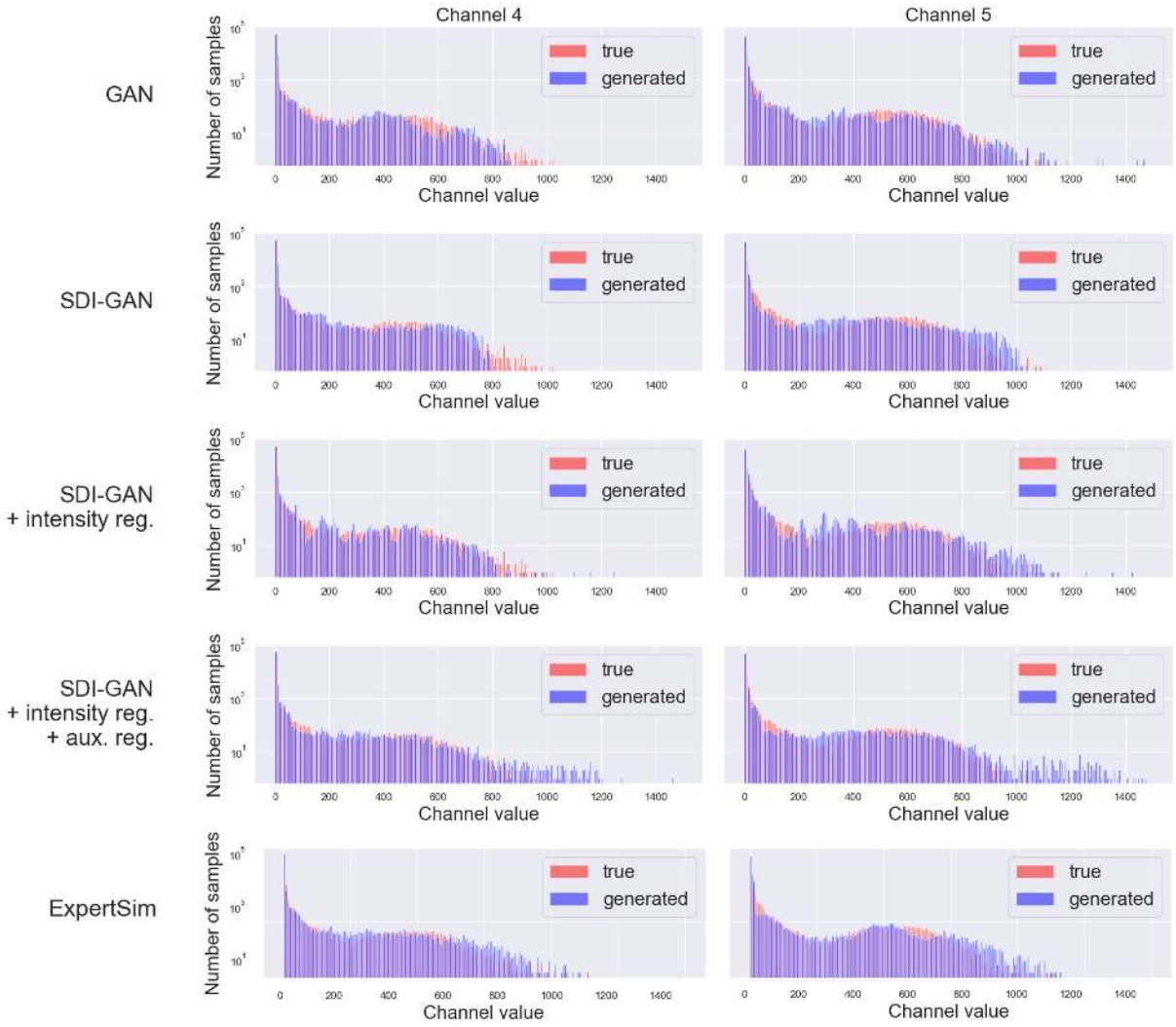


Figure 5.5.2. Histograms of true and generated distributions of ZP channel values.

5.5.2. Experts Specialization

To highlight the main benefit of our Mixture-of-Experts architecture, we propose to study the specialization of each individual expert. To this end, we calculate the mean intensity of responses generated by each expert and report their results in

Table 5.5.1. Mean intensities of responses generated by each expert. Our experts specialize in generating samples with different energies.

Model	Proton		Neutron	
	Mean	**Std. Dev.**	**Mean**	**Std. Dev.**
Expert 1	119	340	108	100
Expert 2	53	204	82	85
Expert 3	70	230	125	113

Table 5.5.2. Effect of the number of experts on the final performance of the generative model. Using 3 experts achieves the best performance for both proton and neutron responses.

Number of Experts	Proton WS↓	Neutron WS↓
2	2.71	1.76
3	**1.59**	**1.34**
4	3.26	1.63
5	3.07	3.12

Tab. 5.5.1. Moreover, in Fig. 5.5.1, we present exemplar outputs generated by each of the three experts.

As evidenced by the higher WS distance in Tab. 5.4.2, single generator methods are unable to capture the whole distribution of the training data. In contrast, in ExpertSim, specific GAN Experts learn different data subsets as shown in Tab. 5.5.1. Each expert specializes in different types of detector responses, which allows us to improve the final performance of the model. This leads to improved performance in our domain, where the ZDC produces responses of distinct types.

5.5.3. Ablation Study

In our design, we maintain a balance between methodological complexity and simulation fidelity. While our framework introduces additional hyperparameters compared to single-model approaches, each component contributes meaningfully to the overall performance. The hyperparameters related to individual experts follow values established in prior work [31], with our tuning efforts focused specifically on the router network parameters (λ_{util}, λ_{diff}) that govern expert specialization and utilization.

The number of experts we use (3) is motivated by the characteristics of our problem, as the calorimeters (both ZP and ZN) generally produce: 1) low-intensity, dispersed responses, 2) medium-intensity responses, and 3) high-intensity, focused

Table 5.5.3. Performance (WS) with varying router hyperparameter values. Our method remains robust across a wide range of λ_{util} and λ_{diff} values for both proton and neutron simulations.

Hyperparameters		**WS↓**		**Comment**
λ_{util}	λ_{diff}	**Proton**	**Neutron**	
0.01	0.0001	**1.59**	**1.34**	Default
0.1	0.0001	1.86	4.14	Higher λ_{util}
0.001	0.00001	2.34	2.52	Lower λ_{util}
0.01	0.001	1.92	1.91	Higher λ_{diff}
0.01	0.000001	1.78	1.74	Lower λ_{diff}
0.01	0	6.80	8.01	L_{diff} disabled
0	0.0001	2.10	1.89	L_{util} disabled

responses. To confirm our insight experimentally, we run an additional ablation study with a different number of experts. As presented in Tab. 5.5.2, using 3 experts outperforms other approaches, which confirms our assumptions.

λ_{util} and λ_{diff} are key components for our routing, where L_{diff} enforces expert specialization, and L_{util} promotes balanced distribution of tasks. We study the effect of their values in Tab. 5.5.3. As visible, disabling L_{diff} leads to poor specialization among agents, while without L_{util}, our routing collapses. However, apart from those extreme cases, our method is robust to the exact values of λ_{util} and λ_{diff}. For the remaining hyperparameters specific to individual generative agents, we select values provided by Dubiński et al. [81].

While we strive for simplicity in our approach, the number of hyperparameters in ExpertSim is driven by our goal of achieving optimal simulation fidelity. As demonstrated in Tab. 5.4.2, each additional component (diversity regularization, intensity regularization, auxiliary regression) progressively improves SDI-GAN's performance, with our ExpertSim ultimately surpassing all variants. These performance improvements justify the additional hyperparameters. As we build on top of bed [31], we maintain consistency by reusing their hyperparameter values for individual experts, focusing our tuning efforts only on router-specific parameters (λ_{util}, λ_{diff}) through a compact logarithmic grid search. To minimize tuning complexity, we share all hyperparameter values across experts. As Tab. 5.5.3 demonstrates, our framework exhibits considerable robustness to parameter selection within reasonable ranges.

5.5.4. Inference Time and Scalability.

The ALICE experiment at CERN relies on CPU-based infrastructure, including Monte Carlo simulations that run on the CERN computational grid. To ensure a fair comparison of simulation speed across methods, we benchmark all approaches

Table 5.5.4. Inference time across different numbers of experts. MC refers to a Monte Carlo ensemble.

Number of Experts	CPU Time [s]	GPU Time [s]
1	273	3.20
2	315	3.21
3	314	3.27
4	324	3.44
5	333	3.55
Monte-Carlo	11172	–

on a CPU to reflect this target deployment environment. Tab. 5.5.4 presents the inference time required to simulate 10,000 responses using our method with varying numbers of experts, as well as a baseline Monte Carlo (MC) ensemble. All CPU benchmarks are conducted on an Intel Core i7-9800X. Given that our method can also be efficiently executed on GPUs, we additionally report inference times with a single Nvidia A5000 GPU.

We can observe that there is a small overhead in the inference time as we increase the number of experts. This can be attributed to the fact that with each added expert, we additionally adapt the shared part of the network to accommodate the extended overall architecture. Nevertheless, the resulting performance is an order of magnitude faster than the standard Monte-Carlo approach.

5.6. Conclusions

In this work, we applied and expanded generative modeling techniques to simulate the responses of the Zero Degree Calorimeter within the ALICE experiment at CERN. Using Mixture-of-Generative-Experts approach, we leveraged multiple GAN-based experts to improve the fidelity of our simulation framework.

We incorporated SDI-GAN to boost the diversity of generated samples, and we implemented a tailored intensity regularization method to minimize differences between real and generated calorimeter responses. Additionally, the use of an auxiliary regressor enabled more accurate spatial feature learning, further enhancing the simulation's fidelity.

By training a router network that assigns samples based on physical characteristics, our approach not only improved the quality of generated data but also demonstrated the potential for efficient simulation of high-energy physics experiments. The results indicate that our MoE-based simulation method offers an

alternative to traditional Monte-Carlo methods, outperforming approaches based on a single generative model.

6. Bucks for Buckets (B4B): Active Defenses Against Stealing Encoders

Title	Bucks for Buckets (B4B): Active Defenses Against Stealing Encoders
Authors	Jan Dubiński*, Stanisław Pawlak*, Franziska Boenisch*, Tomasz Trzciński, Adam Dziedzic
Conference	Conference on Neural Information Processing Systems (NeurIPS)
Year	2023

Preface

After developing generative models tailored for high-energy physics simulations, we turn our attention to a different but equally important aspect of machine learning: protecting the intellectual value contained in models themselves. As deep learning systems become more costly to train and more widely deployed, the incentive to obtain them without authorization increases. One growing threat is model stealing, where an adversary interacts with a model through an exposed API and reconstructs a substitute that closely mimics its behaviour.

Encoder models, which transform raw data into structured representations, are especially attractive targets. They are central to many downstream tasks, from vision and language understanding to multimodal applications, and their reuse across pipelines makes them high-value assets. Yet, existing defenses are largely passive, focusing on limiting query access or adding noise, and they fail against determined attackers who adapt their strategies.

In this work, we introduce the first active defense against stealing encoders. Our method, called Bucks for Buckets (B4B), monitors user queries and detects extensive exploration of the embedding space. Once suspicious behaviour is identified, the system imposes adaptive costs that penalize extraction attempts, making replication impractical. To counter adversaries operating through multiple accounts, we additionally apply per-user transformations of the representation space, which prevents coordination across accounts and strengthens the defense.

We evaluate B4B against state-of-the-art stealing strategies across several encoder architectures. The results show that our defense reliably detects and disrupts extraction, while maintaining usability for benign users. This demonstrates that it is possible to move beyond passive defenses and actively intervene to protect valuable machine learning models.

With this chapter, the dissertation shifts from the challenge of building reliable generative models to the broader problem of safeguarding machine learning systems. In the following chapters, we extend this perspective from the protection of models themselves to the protection of the data on which they are trained, addressing privacy and copyright concerns in the era of large-scale generative architectures.

Abstract

Machine Learning as a Service (MLaaS) APIs provide ready-to-use and high-utility encoders that generate vector representations for given inputs. Since these encoders are very costly to train, they become lucrative targets for model stealing attacks during which an adversary leverages query access to the API to replicate the encoder locally at a fraction of the original training costs. We propose *Bucks for Buckets (B4B)*, the first *active defense* that prevents stealing while the attack is happening without degrading representation quality for legitimate API users. Our defense relies on the observation that the representations returned to adversaries who try to steal the encoder's functionality cover a significantly larger fraction of the embedding space than representations of legitimate users who utilize the encoder to solve a particular downstream task. B4B leverages this to adaptively adjust the utility of the returned representations according to a user's coverage of the embedding space. To prevent adaptive adversaries from eluding our defense by simply creating multiple user accounts (sybils), B4B also individually transforms each user's representations. This prevents the adversary from directly aggregating representations over multiple accounts to create their stolen encoder copy. Our active defense opens a new path towards securely sharing and democratizing encoders over public APIs.

6.1. Introduction

In model stealing attacks, adversaries extract a machine learning model exposed via a public API by repeatedly querying it and updating their own stolen copy based on the obtained responses. Model stealing was shown to be one of the main threats to the security of machine learning models in practice [212]. Also in research, since the introduction of the first extraction attack against classifiers [225], a lot of work on improving stealing [133, 169, 225, 227], extending it to different model types [19, 208], and proposing adequate defenses [87, 126, 129, 165] has been put forward. With the recent shift in learning paradigms from supervised to self supervised learning (SSL), especially the need for new defenses becomes increasingly pressing. From an academic viewpoint, the urge arises because it was shown that SSL models (*encoders*) are even more vulnerable to model stealing [85, 153, 205] than their supervised counterparts. This is because whereas supervised models' output is low dimensional, *e.g.,* per-class probabilities or pure labels, SSL encoders output high-dimensional representation vectors that encode a larger amount of information and thereby facilitate stealing. In addition, from a practical industry's viewpoint, defenses are

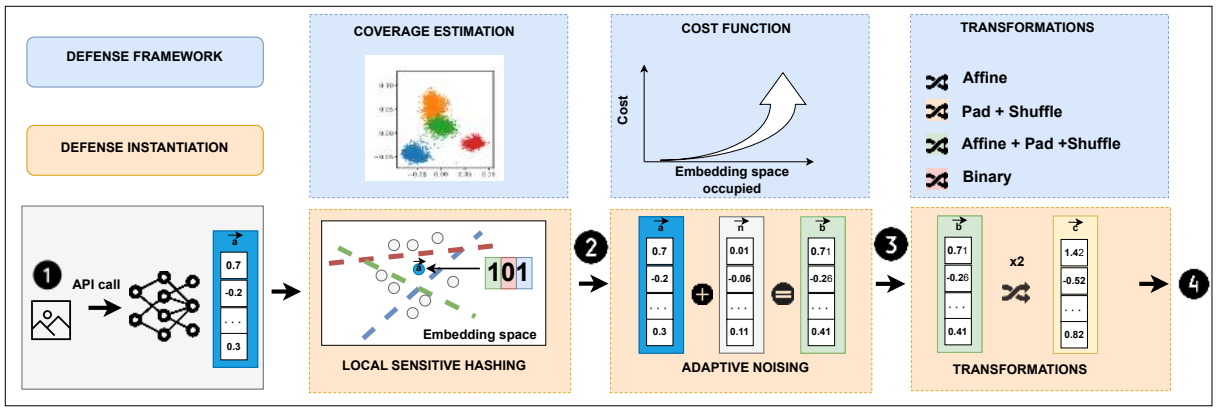


Figure 6.1.1. Overview of B4B. In the upper part, we present our B4B framework that consists of three modular building blocks: (1) A coverage estimation to track the fraction of embedding space covered by the representations returned to each user, (2) a cost function that serves to map the coverage to a concrete penalty to prevent stealing, and (3) per-user transformations that are applied to the returned representations to prevent sybil attacks. In the lower part, we present a concrete instantiation of B4B and the operation flow of our defense: ❶ The API calculates representations for the incoming queries. ❷ We instantiate the coverage estimation with local sensitive hashing and estimate the covered space as the fraction of *hash buckets* occupied. We calibrate the costs by adding noise to the representations according to the coverage. ❸ We apply a set of transformations on a per-user basis. ❹ The noised and transformed representations are returned to the user.

required since many popular API providers, such as Cohere, OpenAI, or Clarify [1–3] already expose their high-value SSL encoders via APIs to a broad range of users.

Most of the current defenses against encoder stealing are *reactive*, *i.e.,* they do not actively prevent the stealing but rather aim at detecting it by adding watermarks to the encoder [61, 85] or performing dataset inference to identify stolen copies [86]. Since at the point of detection, the damage of stealing has already been inflicted, we argue that reactive defenses intervene too late and we advocate for *active* defenses that prevent stealing while it is happening. Yet, active defenses are challenging to implement because they not only need to prevent stealing but also should preserve the utility of representations for legitimate users. The only existing active defense against encoder stealing [153] falls short on this latter aspect since it significantly degrades the quality of representations for all users.

To close the gap between required and existing defenses, we propose *Bucks for Buckets (B4B)*, the first active defense against encoder stealing that does not harm utility for legitimate users. B4B leverages the observation that the representations returned to adversaries who try to steal the encoder's functionality cover a significantly larger fraction of the full embedding space than representations of legitimate users who utilize the encoder to solve a particular downstream task. To turn this observation into a practical defense, B4B is equipped with three modular

building blocks: (1) The first building block is a tracking mechanism that continuously estimates the fraction of the embedding space covered by the representations returned to each user. The intuition why this is relevant is that by covering large fractions of the embedding space, the representations will suffice for an adversary to reproduce the encoder's functionality, *i.e.,* to successfully steal it. (2) B4B's second building block consists of a cost function to translate the covered fraction of the embedding space into a concrete penalty. We require this cost function to significantly penalize adversaries trying to steal the model while having only a minimal effect on legitimate users. (3) The third building block contains transformations that can be applied to the representations on a per-user basis to prevent adaptive attackers from circumventing our defense by creating multiple user accounts (sybils) and distributing their queries over these accounts such that they minimize the overall cost. We present the different building blocks of B4B in Figure 6.1.1.

While B4B's modularity enables different instantiations of the three building blocks, we propose a concrete end-to-end instantiation to showcase the practicability of our approach. To implement tracking of the covered embedding space, we employ *local sensitive hashing* that maps any representation returned to a given user into a set of hash ***buckets***. We base our cost function (*i.e.,* the ***"bucks"***) on utility and make B4B add noise to the representations with a magnitude that increases with the number of buckets occupied by the given user. While the scale of noise added to legitimate users' representations does not harm their downstream performance due to their small embedding space coverage, the representations returned to an adversary become increasingly noisy—significantly degrading the performance of their stolen encoder. Finally, we rely on a set of transformations (*e.g.,* affine transformations, shuffling, padding) that preserve downstream utility [86]. While, as a consequence, legitimate users remain unaffected by these transformations, adversaries cannot directly combine the representations obtained through different sybil accounts anymore to train their stolen copy of the encoder. Instead, they first have to remap all representations into the same embedding space, which we show causes both query and computation overhead and still reduces the performance of the stolen encoder.

In summary, we make the following contributions:

1. We present B4B, the first active defense against encoder stealing that does not harm legitimate users' downstream performance. B4B's three building blocks enable penalizing adversaries whose returned representations cover large fractions of the embedding space and prevent sybil attacks.
2. We propose a concrete instantiation of B4B that relies on local sensitive hashing

and decreases the quality of representations returned to a user once their representations fill too many hash buckets.

3. We provide an end-to-end evaluation of our defense to highlight its effectiveness in offering high utility representations for legitimate users and degrading the performance of stolen encoders in both the single and the sybil-accounts setup.

6.2. Related Work

Model Extraction Attacks. The goal of the model extraction attacks is to replicate the functionality of a victim model f_v trained on a dataset D_v. An attacker has a black box access to the victim model and uses a stealing dataset $D_s = \{q_i, f_v(q_i)\}_{i=1}^n$, consisting of queries q_i and the corresponding outputs $f_v(q_i)$ returned by the victim model, to train a stolen model f_s. Model extraction attacks have been shown against various types of models including classifiers [125, 225] and encoders [85, 205].

Self Supervised Learning and Encoders. SSL is an increasingly popular machine learning paradigm. It trains encoder models to generate representations from complex inputs without relying on explicit labels. These representations encode useful features of a given input, enabling efficient learning for multiple downstream tasks. Many SSL frameworks have been proposed [21, 49, 54, 55, 107, 113]. In our work, we focus on the two popular SSL vision encoders, namely SimSiam [55] and DINO [49], which return high-quality representations that achieve state-of-the-art performance on downstream tasks when assessed by training a linear classifier directly on representations. SimSiam trains with two Siamese encoders with directly shared weights. A prediction MLP head is applied to one of the encoders f_1, and the other encoder f_2 has a stop-gradient, where both operations are used for avoiding collapsing solutions. In contrast, DINO shares only architecture (not weights) between a student f_1 and a teacher model f_2, also with the stop-gradient operation, but not the prediction head. While SimSiam uses convolutional neural networks (CNNs), DINO also employs vision transformers (ViTs). Both frameworks use a symmetrized loss of the form $\frac{1}{2}g(f_1(x_1), f_2(x_2)) + \frac{1}{2}g(f_1(x_2), f_2(x_1))$ in their optimization objectives, where $g(\cdot,\cdot)$ is negative cosine similarity for SimSiam and cross-entropy for DINO. SimSiam and DINO's similarities and differences demonstrate our method's broad applicability across SSL frameworks. More details can be found in Section A.5.

Stealing Encoders. The stealing of SSL encoders was shown to be extremely effective [85, 153, 205]. The goal of extracting encoders is to maximize the similarity of the outputs from the stolen local copy and the original representations output by the victim encoder. Therefore, while training the stolen copy, the adversary

either imitates a self-supervised training using a contrastive loss function, *e.g.,* InfoNCE [54] or SoftNN [100] or directly matches both models' representations via the Mean Squared Error (MSE) loss. To reduce the number of queries sent to the victim encoder, the attack proposed in [153] leverages the key observation that the victim encoder returns similar representations for any image and its augmented versions. Therefore, a given image can be sent to the victim while the stolen copy is trained using many augmentations of this image, where the representation of a given augmented image is approximated as the one of the original image produced by the victim encoder.

Defending Encoders. Recently, watermarking [38, 126, 228] methods have been proposed to detect stolen encoders [61, 85, 249]. Many of these approaches use downstream tasks to check if a watermark embedded into a victim encoder is present in a suspect encoder. Dataset inference [157] is another type of encoder ownership resolution. It uses the victim's training dataset as a unique signature, leveraging the following observation: for a victim encoder trained on its private data as well as for its stolen copies, the distribution of the representations generated from the victim's training data differs from the distribution of the representations generated on the test data. In contrast, for an independently trained encoder, these two distributions cannot be distinguished, allowing the detection of stolen copies [86]. However, all the previous methods are *reactive* and aim at detecting the stolen encoder instead of *actively* preventing the attack. The only preliminary active defenses for encoders were proposed by [85, 153]. They either perturb or truncate the answers to poison the training objective of an attacker. These operations were shown to harm substantially the performance of legitimate users, which renders the defense impractical. In contrast, our B4B has negligible impact on the quality of representations returned to legitimate users.

6.3. Actively Defending against Model Stealing with B4B

B4B aims at actively preventing model stealing while preserving high-utility representations for legitimate users. Before introducing the three main building blocks of B4B, namely (1) the estimation of embedding space coverage, (2) the cost function, and (3) the transformation of representations (see Figure 6.1.1), we detail our threat model and the observation on embedding space coverage that represents the intuition behind our approach.

6.3.1. Threat Model and Intuition

Our setup and the resulting threat model are inspired by public APIs, such as Cohere, OpenAI, or Clarify [1–3] that expose encoders to users through a pre-defined interface. These encoders are trained using SSL on large amounts of unlabeled data, often crawled from the internet, and therefore from diverse distributions. We notice that to provide rich representations to multiple users, the training dataset of the encoder needs to be significantly more diverse than the individual downstream tasks that the users query for representations. For instance, if the encoder behind the API is trained on the ImageNet dataset, then the legitimate users are expected to query the API for downstream tasks, such as CIFAR10 or SVHN. Similarly, if the encoder is trained on CIFAR10, the expected downstream tasks are MNIST or Fashion MNIST. Yet, in the design of our defense, we consider adversaries who can query the encoder with arbitrary inputs to obtain high-dimensional representation vectors from the encoder. Our defense is independent of the protected encoder's architecture and does not rely on any assumption about the adversary's data and query strategy.

We argue that even in this restricted setup, our defense can distinguish between adversaries and legitimate users by analyzing the distribution of representations returned to them. In Figure 6.3.1, by using PCA to project representations for different datasets to a two-dimensional space, we visualize that representations for different downstream tasks cluster in *disjoint* and *small sub-spaces* of the full embedding space. The representations were obtained from a SimSiam encoder originally trained on ImageNet (we observe similar clustering for DINO shown in Section A.6). As a result, legitimate users can be characterized by their representations' small coverage of the embedding space. In contrast, the adversary does not aim at solving a particular downstream task. They instead would want to obtain representations that cover large fractions of the embedding space. This enables reproducing the overall functionality of the encoder (instead of only learning some local task-specific behavior). Indeed, it has been empirically shown by prior work, such as [85], that stealing with multiple distributions, *e.g.,* by relying on the complex ImageNet dataset, yields higher performance of the stolen encoder on various downstream tasks than stealing with a downstream dataset, such as CIFAR10. As a result, intuitively, we can identify and penalize adversaries

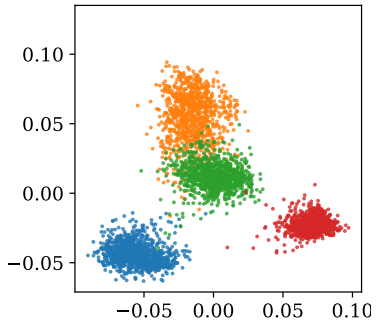


Figure 6.3.1. Representations from Different Tasks Occupy Different Sub-Spaces of the Embedding Space. Presented for FashionMNIST, SVHN, CIFAR10, and STL10.

based on their coverage of the embedding space, which will be significantly larger than the coverage of legitimate users. We leverage this intuition to build our B4B defense and present our three main building blocks in the following sections.

6.3.2. Building Block 1: Coverage Estimation of the Embedding Space

The first building block of our B4B serves to estimate and continuously keep track of the fraction of the embedding space occupied by any given user. Let $\mathcal{E}$ denote our embedding space of dimension s, further, let U be a user with a query dataset $D = q_1, \ldots, q_n \in \mathcal{D}$ and let $f_v : \mathcal{D} \to \mathcal{E}$ be our protected victim encoder that maps data points from the input to the embedding space. Assume user U has, so far, queried a subset of their data points $q_1, \ldots, q_j$ with $j \leq n$ to the encoder and obtained the representations $r_1, \ldots, r_j$ with each $r_i \in \mathbb{R}^s$. We aim to estimate the true fraction of the embedding space $\mathcal{E}_f^U$ that is covered by all returned representations $r_1, \ldots, r_j$ to user U and denote our estimate by $\tilde{\mathcal{E}}_f^U$.

Local Sensitive Hashing. One of the methods to approximate the occupied space by representations returned to a given user is via Local Sensitive Hashing (LSH) [213]. We rely on this approach for the concrete instantiation of our B4B and use it to track per-user coverage of the embedding space. Standard (cryptographic) hash functions are characterized by high dispersion such that hash collisions are minimized. In contrast, LSH hashes similar data points into the same or proximal, so-called *hash buckets*. This functionality is desired when dealing with searches in high-dimensional spaces or with a large number of data points. Formally, an LSH function $\mathcal{H}$ is defined for a metric space $\mathcal{M} = (M, d)$, where d is a distance metric in space M, with a given threshold $T > 0$, approximation factors $f > 1$, and probabilities P_1 and P_2, where $P_1 \gg P_2$. $\mathcal{H}$ maps elements of the metric space to buckets $b \in B$ and satisfies the following conditions for any two points $q_1, q_2 \in M$: (1) If $d(q_1, q_2) \leq T$, then $\mathcal{H}(q_1) = \mathcal{H}(q_2)$ (*i.e.,* q_1 and q_2 collide in the same bucket b) with probability at least P_1. (2) If $d(q_1, q_2) \geq fT$, then $\mathcal{H}(q_1) = \mathcal{H}(q_2)$ with probability at most P_2.

6.3.3. Building Block 2: Cost Function Design

Once we can estimate the coverage of an embedding space for a given user U as $\tilde{\mathcal{E}}_f^U$, we need to design a cost function $\mathcal{C} : \mathbb{R}^+ \to \mathbb{R}^+$ that maps from the estimated coverage to a cost. The cost function needs to be designed such that it does not significantly penalize legitimate users while imposing a severe penalty on adversaries to effectively prevent the encoder from being stolen. The semantics of the cost function's range depend on the type of costs that the defender wants to enforce. We discuss a broad range of options in Section A.3. These include monetary cost functions to adaptively

charge users on a batch-query basis depending on their current coverage, costs in the form of additional computation that users need to perform in order to obtain their representations, similar to the proof of work in [87], costs in the form of delay in the interaction with the encoder behind the API [4], or costs in form of disk space that needs to be reserved by the user (similar to a proof of space [88, 88]). Which type of cost function is most adequate depends on the defender's objective and setup.

Exponential Cost Functions to Adjust Utility of Representations. In the concrete instantiation of B4B that we present in this work, we rely on costs in the form of the utility of the returned representations. We choose this concrete instantiation because it is intuitive, effective, and can be directly experimentally assessed. Moreover, it is even suitable for public APIs where, for example, no monetary costs are applicable. We adjust utility by adding Gaussian noise with different standard deviation σ to the returned representations. Since we do not want to penalize legitimate users with small coverage but make stealing for adversaries with growing coverage increasingly prohibitive, we instantiate an exponential cost function that maps from the fraction of hash buckets occupied by the user to a value for σ. We choose the general form of this function as

$$f_{\lambda,\alpha,\beta}(\tilde{\mathcal{E}}_f^U) = \lambda \times (\exp^{\ln\frac{\alpha}{\lambda}\times\tilde{\mathcal{E}}_f^U\times\beta^{-1}} - 1) \tag{6.1}$$

where $\lambda < 1$ compresses the curve of the function to obtain low function values for small fractions of occupied buckets, and then we set a target penalty α for our cost function at a specified fraction of filled buckets β. For instance, if we want to enforce a σ of 5 at 90% of filled buckets (*i.e.,* for $\tilde{\mathcal{E}}_f^U = 0.9$), we would need to set $\alpha = 5$ and $\beta = 0.9$.

6.3.4. Building Block 3: Per-User Representation Transformations against Sybil Attacks

Given that our defense discourages users from querying with a wide variety of data points from different distributions, an adversary could create multiple fake user accounts (sybils) and query different data subsets with more uniform representations from each of these accounts. By combining all the obtained representations and using them to train a stolen copy, the adversary could overcome the increased costs of stealing. To defend against such sybil attacks, we propose individually transforming the representations on a per-user level before returning them. As a result, the adversary would first have to map all the representations to one single unified space

before being able to jointly leverage the representations from different accounts for their stolen copy.

Formally, for a given query q_i, the protected victim encoder produces a representation $r_i = f_v(q_i)$, which is transformed by a user-specific transformation $T_U(r_i)$ before being returned to the querying user U. For a new user U, the defense randomly selects the transformation T_U from all possible choices. Note that the randomness is also added on a per-transformation basis, instead of only on the level of selecting the transformations. For example, a permutation of the elements in the output representations should be different for each user.

We formulate two concrete requirements for the transformations. First, they should preserve utility for legitimate users on their downstream tasks, and second, they should be costly to reverse for an adversary.

Utility Preserving Transformations. As a concrete instantiation for our B4B, we propose a set of transformations that fulfill the above-mentioned two requirements: (1) *Affine* where we apply affine transformations to representations, (2) *Pad* where we pad representations with constant values, (3) *Add* where we add constant values at random positions within representations, (4) *Shuffle* where we shuffle the elements in the representation vectors, and (5) *Binary* where the original representations are mapped to binary vectors relying on a random partitioning of the representation space. To preserve the full amount of information contained in the original representations, in our binary transformations, we tune the length of binary representations. We visualize the operation of each of these transformations in Section A.3. All these transformations can additionally be combined with each other, which further increases the possible set of transformations applied per user. This renders it impossible for an adversary to correctly guess and reverse the applied representation. Instead, the adversary has to remap the representations over all accounts into a single embedding space in order to unify them and leverage them for training of their stolen encoder copy. We present an exhaustive list of strategies that adversaries can apply for the remapping in Section A.4. All the attack methods reduce to the minimum of remapping between representations of a pair of users, *i.e.,* they are at least as complex as mapping between two separate accounts. In the next section, we show that our defense already impedes stealing for an adversary with two accounts.

6.4. Empirical Evaluation

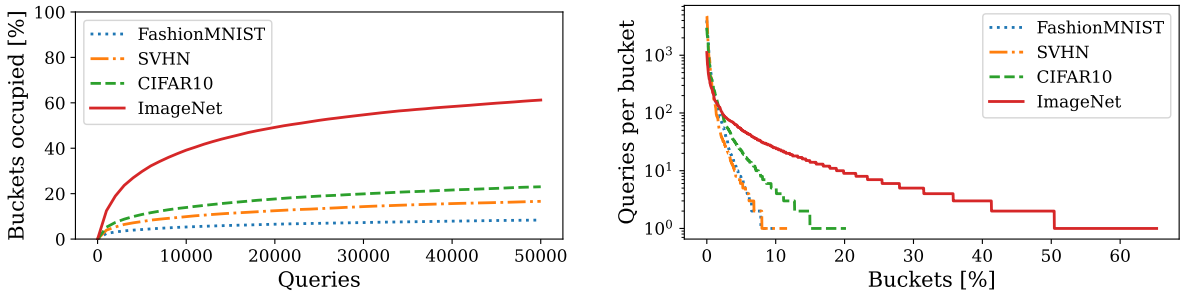
We first empirically evaluate our instantiation of B4B's three building blocks and show how to calibrate each of them for our defense. Finally, we provide an end-to-end evaluation that highlights B4B's effectiveness in preserving downstream utility for legitimate users while successfully preventing the stealing by adversaries.

Experimental Setup. We conduct experiments on various kinds of downstream tasks and two popular SSL encoders. To test our defense, we use FashionMNIST, SVHN, STL10, and CIFAR10 as our downstream datasets, each with standard train and test splits. For stealing, we utilize training data from ImageNet and LAION-5B. We rely on encoder models from the SimSiam [55] and the DINO [49] SSL frameworks. As our victim encoders, we use the publicly available ResNet50 model from SimSiam trained for 100 epochs on ImageNet and the ViT Small DINO encoder trained for 800 epochs on ImageNet, each using batch size 256. The ViT architecture takes as input a grid of non-overlapping contiguous image patches of resolution NxN. In this paper, we typically use $N = 16$. The Simsiam encoder has an output representation dimension of 2048, while DINO returns 1536 dimensional representations. We examine the utility of downstream classifiers using SimSiam's or DINO's representations obtained for the respective downstream datasets. To implement LSH, we rely on random projections [9] that we implement from scratch. For a detailed description of our stealing and downstream training parameters, we refer to Section A.6.

6.4.1. Local Sensitive Hashing for Coverage Estimation

We first observe that the choice of the total number of hash buckets in the LSH influences the effectiveness of our method. In the extreme, if we have a too large number of buckets, the number of buckets filled will correspond to the number of queries posed by a user which fails to capture that similar representations cover similar sub-spaces of the embedding space, and hence does not serve to approximate the total fraction of the embedding space covered. However, if we have too few buckets, even the representations for simple downstream tasks will fill large fractions of buckets, making it impossible to calibrate the cost function such that it only penalizes adversaries. We experimentally find that for our evaluated encoders, 2^{12} buckets represent a good trade-off. In Section A.6.6, we present an ablation study on the effect of the number of total buckets.

Our evaluation of the LSH to track coverage of the embedding space is presented in Figure 6.4.1. We observe that representations returned for standard downstream



(a) Fraction of Occupied Buckets. **(b)** Number of Queries per Bucket.

Figure 6.4.1. Estimating Embedding Space Coverage through LSH on SimSiam Encoder. We present the fraction of buckets occupied by representations of different datasets as a function of the number of queries posed to the encoder *(left)*. We observe that representations for the downstream datasets (FashionMNIST, SVHN, CIFAR10) occupy a smaller fraction of buckets than representations from the complex ImageNet dataset. Our evaluation of the number of queries whose representations are mapped to the same bucket *(right)* indicates that our total number of buckets (2^{12}) is well calibrated for the estimation of covered representation space: over all datasets, we experience hash collisions, *i.e.,* queries whose representations are mapped to the same buckets. This indicates that our LSH is capable of representing similarities in the representations.

tasks (FashionMNIST, SVHN, CIFAR10) occupy a significantly smaller fraction of the total number of buckets than complex data from multiple distributions (ImageNet, LAION-5B). We present additional experimental results on measuring the coverage of the representation space in Section A.6.5. Specifically, we show that our method of measuring the embedding space coverage has broad applicability across various encoders and datasets used for pretraining. We further observe that the fraction of buckets occupied by the representations saturates over time. These findings highlight that LSH is successful in capturing the differences between legitimate users and adversaries—even in a low-query regime. Finally, we note that our total number of buckets (2^{12}) is well calibrated since, over all datasets, it successfully maps multiple representations to the same hash bucket while still filling various fractions of the total number of buckets.

We experiment with different sets of hyperparameters to instantiate the cost function from Equation (6.1) in the previous section (6.3.3). As described there, we can calibrate the function (as shown in Figure 6.4.2) such that a desired penalty (in the form of a specific σ) will be assigned at a certain fraction of buckets occupied. For B4B, we aim at penalizing high embedding space coverage severely. Therefore, we need to identify and optimize for two components: 1) which value of σ leads to significant performance drops, and 2) for what fraction of coverage do we want to impose this significant drop. We base both components on empirical observations.

Our first observation is that for our four downstream tasks (FashionMNIST, SVHN, STL10, and CIFAR10), performance drops to 10% (*i.e.,* random guessing) at roughly $\sigma = 0.5$.

6.4.2. Calibrating the Cost Function

In Figure 6.4.1, we further see that with 50k queries, the downstream tasks occupy $< 30\%$ of the buckets. Ultimately, setting α and β are design choices that an API provider needs to make in order to specify what type of query behavior they want to penalize. As very loose bounds (weak defense), based on our observation, we consider $\sigma = 1$ as a high penalty, which leads to $\alpha = 1$, and select $\beta = 0.8$. This β corresponds to considering 80% of buckets filled as a too-large coverage of the embedding space. We empirically observe that coverage of 80% of buckets occurs, for example, after around 100k of ImageNet queries. By choosing our target β so loose, *i.e.,* significantly larger than the observed 30% for downstream tasks, we offer flexibility for the API to also provide good representations for more complex downstream tasks. Finally, to obtain a flat cost curve close to the origin—which serves to map small fractions of covered buckets to small costs—we find that we can set $\lambda = 10^{-6}$. In the Appendix, we evaluate our defense end-to-end with differently parameterized cost functions.

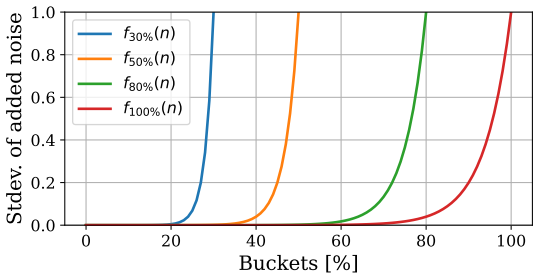


Figure 6.4.2. Cost Function Calibration.

6.4.3. Assessing the Effect of Transformations

Transformations Do Not Harm Utility for Legitimate Users. We evaluate the downstream accuracy for transformed representations based on training a linear classifier on top of them. To separate the effect of the noise added by our defense from the effect of the transformations, we perform the experiments in this subsection without adding noise to the returned representations. For example, on the CIFAR10 dataset and a SimSiam encoder pre-trained on ImageNet, without any transformations applied, we obtain a downstream accuracy of 90.41% ($\pm$ 0.02), while, with transformations, we obtain 90.24% ($\pm$ 0.11) for Affine, 90.40% ($\pm$ 0.05) for Pad+Shuffle, 90.18% ($\pm$ 0.06) for Affine+Pad+Shuffle, and 88.78% ($\pm$ 0.2) for Binary. This highlights that the transformations preserve utility for legitimate users. This holds over all datasets we evaluate as we show in Section A.6.

Adversaries Cannot Perfectly Remap Representations over Multiple Sybil Accounts. To understand the impact of our per-user account transformations on sybil-attack based encoder stealing, we evaluate the difficulty of remapping representations between different sybil accounts. For simplicity, and since we established in Section 6.3.4 that multi-account attacks reduce to a two-account setup, we assume an adversary who queries from two sybil accounts and aims at learning to map the transformed representations from account #2 to the representation space of account #1. Using more accounts for the adversary causes a larger query overhead and potentially more performance loss from remapping. Our evaluation here, hence, represents a lower bound on the overhead caused to the adversary through our transformations.

We learn the mapping between different accounts' representations by training a linear model on overlapping representations between the accounts. We assess the fidelity of remapped representations as a function of the number of overlapping queries between the accounts. As a fidelity metric for our remapping, we compare the cosine distance between representations (a and b defined as: $1 - \frac{a^T b}{||a||_2 \cdot ||b||_2}$). Once the remapping is trained, we evaluate by querying 10k data points from the test dataset through account #1 and then again through account #2. Then, we apply the learned remapping to the latter one and compute the pairwise cosine distances between the representations from account #1 and their remapped counterparts from account #2. Our results are depicted in Figure 6.4.3. We show that the largest cosine distance is achieved with the binary transformations, making them the most protective against the adversary since they best prevent perfect remapping, even with an overlap of as many as 10k queries between both accounts. However, these binary transformations also incur the highest drop in accuracy for legitimate users. The defender has the possibility of selecting their preferred types of transformations between representations taking into account the trade-offs between the effectiveness of the defense and the negative impact on legitimate users.

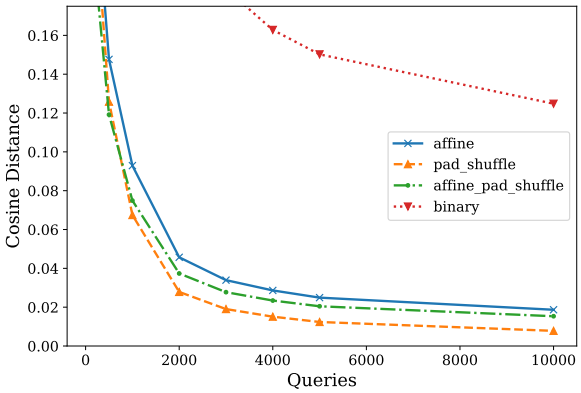


Figure 6.4.3. Quality of Remappings.

6.4.4. End-to-End Stealing of an Encoder under our Defense

We perform an end-to-end study to showcase how our B4B defense affects legitimate users vs adversaries. The hyperparameters for B4B are chosen according to

Table 6.4.1. Stealing and Using Encoders With and Without our Defense. The *USER* column represents the type of the APIs' user, where LEGIT denotes a legitimate user, ATTACKER stands for a standard single-account adversary, and SYBIL represents an adversary using two sybil accounts. We use InfoNCE loss for encoder extraction. # Queries stands for the number of queries used for stealing with ALL denoting that the entire downstream dataset was used. The *TYPE* column expresses how the dataset is used. We follow the stealing setup from [86]. In the first row, we present the undefended victim encoder's performance as the accuracy for downstream tasks trained on the encoder's returned representations. In the following row, we show downstream utility for legitimate users when the victim encoder is defended by our B4B. Finally, (in the remaining rows) we assess the performance of stolen encoders on the downstream tasks. Our results highlight that while the performance of the encoder for legitimate users stays high, our B4B renders stealing inefficient with the stolen encoders obtaining significantly worse performance on downstream tasks.

USER	DEFENSE	# QUERIES	DATASET	TYPE	CIFAR10	STL10	SVHN	F-MNIST
LEGIT	NONE	ALL	TASK	QUERY	90.41 $\pm$0.02	95.08$\pm$0.13	75.47$\pm$0.04	91.22$\pm$0.11
LEGIT	B4B	ALL	TASK	QUERY	90.24$\pm$0.11	95.05$\pm$0.10	74.96$\pm$0.13	91.70$\pm$0.01
ATTACK	NONE	50K	IMGNET	STEAL	65.2$\pm$0.03	64.9$\pm$0.01	63.1$\pm$0.01	88.5$\pm$0.01
ATTACK	B4B	50K	IMGNET	STEAL	35.72$\pm$0.04	31.54$\pm$0.02	19.74$\pm$0.02	70.01$\pm$0.01
ATTACK	NONE	100K	IMGNET	STEAL	68.1$\pm$0.03	63.1$\pm$0.01	61.5$\pm$0.01	89.0$\pm$0.07
ATTACK	B4B	100K	IMGNET	STEAL	12.01$\pm$0.07	13.94$\pm$0.05	19.96$\pm$0.03	69.63$\pm$0.07
ATTACK	NONE	100K	LAION	STEAL	64.92$\pm$0.03	62.51$\pm$0.03	59.02$\pm$0.02	84.54$\pm$0.01
ATTACK	B4B	100K	LAION	STEAL	40.96$\pm$0.03	40.69$\pm$0.05	34.43$\pm$0.01	72.92$\pm$0.01
SYBIL	B4B	2×50K	IMGNET	STEAL	39.56$\pm$0.06	38.50$\pm$0.04	23.41$\pm$0.02	77.01$\pm$0.08
SYBIL	B4B	3×33.3K	IMGNET	STEAL	33.87$\pm$0.05	38.57$\pm$0.06	21.16$\pm$0.01	72.95$\pm$0.05
SYBIL	B4B	4×25K	IMGNET	STEAL	33.98$\pm$0.04	34.52$\pm$0.08	21.21$\pm$0.02	70.71$\pm$0.05
SYBIL	B4B	5×20K	IMGNET	STEAL	32.65$\pm$0.05	32.45$\pm$0.05	29.63$\pm$0.01	70.12$\pm$0.08
SYBIL	B4B	6×16.7K	IMGNET	STEAL	26.62$\pm$0.04	26.85$\pm$0.05	24.32$\pm$0.02	70.51$\pm$0.04

the empirical evaluation of the previous sections with 2^{12} as the number of buckets, $\alpha = 1, \beta = 0.8, \lambda = 10^{-6}$ as the hyperparameter of the cost function, and different random affine transformations per-user account. Our main results are presented in Table 6.4.1. We observe that instantiating our framework with B4B has a negligible impact on legitimate users while substantially lowering the performance of stolen encoders in the case of single-user and sybil attackers.

Legitimate Users. We compare the accuracy of downstream classifiers trained on top of unprotected vs defended encoders. The victim encoder achieves high accuracy on the downstream tasks when no defense is employed. With B4B in place, we observe that across all the downstream tasks, the drop in performance is below 1%. For example, there is only a slight decrease in the accuracy of CIFAR10 from 90.41±0.02% to 90.24±0.11%. B4B's small effect on legitimate users stems from the fact that their downstream representations cover a relatively small part of the representations space. This results in a very low amount of noise added to their representations which preserves performance.

Adversaries. For adversaries who create a stolen copy of the victim encoder, we make two main observations. The most crucial one is that when our B4B is in place, the performance of the stolen copies over all downstream tasks significantly drops in comparison to when the victim encoder is unprotected (grey rows in Table 6.4.1). This highlights that our B4B effectively prevents stealing. Our next key observation concerns the number of stealing queries used by the adversary: When no defense is applied, the more queries are issued against the API (*e.g.,* 100k instead of 50k), the higher performance of the stolen encoder on downstream tasks (*e.g.,* CIFAR10 or FashionMNIST). In contrast, with B4B implemented as a defense, the performance decreases when using more stealing queries from a single account. This is because with more queries issued, the coverage of embedding space grows which renders the returned representations increasingly noisy and harms stealing performance.

Moreover, we show that B4B can also prevent model stealing attacks with data from a different distribution than the victim encoder's training set. We highlight this in Table 6.4.1 where we also use the LAION-5B dataset to steal an ImageNet pre-trained encoder. Our results highlight first that without any defense in place, the LAION dataset is highly effective to extract the ImageNet pre-trained encoder. Second, B4B effectively defends against such attacks, and yields a significant drop in downstream accuracy (on average above 20%) of the stolen encoder.

We also show that this performance drop cannot be prevented by sybil attacks. Therefore, we first consider an adversary who queries from two sybil accounts with 50k queries issued per account and the first 10k queries of both accounts used to learn the remapping of representations between them. When the adversary trains their stolen encoder copy on all the remapped representations, they increase downstream performance over querying from a single account. Yet, their performance is still significantly smaller than the performance of the victim encoder for legitimate users, or the encoder stolen from an undefended victim. Moreover, using more than two sybil accounts further reduces the attack performance as remapping complications accumulate. With ten sybils, remapping leaves no more usable data for training the stolen encoder. This demonstrates our method's advantage: increasing the number of sybil accounts makes encoder extraction impractical due to the growing remapping overhead. Overall, the results highlight that our B4B also successfully prevents sybil attacks.

6.4.5. Baseline Comparison

Finally, we compare our B4B against the current state-of-the-art baseline defense, namely a static addition of noise to all the returned representations (as proposed in [85] (Section A.4),[153, 205]). For direct comparability, we evaluate the defense

using the our end-to-end experiment setup from the previous section. We present our results in Table A.6.5 in Section A.6.4. Confirming the findings from [85] our results also show that defenses that rely on static noise have the great disadvantage to harm legitimate users and attackers equally. When adding noise with a small standard deviation of $\sigma = 0.1$, we observe a negligible (<1%) performance drop for both attackers and legitimate users. Adding noise with a large standard deviation of, for example, $\sigma = 10$, we observe that both legitimate users' and attackers' performance drops between 15% and >40%. In summary, these defenses can either effectively defend stealing (but harm legitimate users), or keep utility for legitimate users high (but not defend well against stealing). In contrast, our B4B is able to provide high performance for legitimate users while effectively defending the encoder against stealing attacks.

6.5. Conclusions

We design B4B a new and modular active defense framework against stealing SSL encoders. All the previous approaches were either reactive, acting after the attack happened to detect stolen encoders, or lowered the quality of outputs substantially also for legitimate users which rendered such mechanisms impractical. We show that B4B successfully distinguishes between legitimate users and adversaries by tracking the embedding space coverage of users' obtained representations. B4B then leverages this tracking to apply a cost function that penalizes users based on the current space coverage, for instance, by lowering the quality of their outputs. Finally, B4B prevents sybil attacks by implementing per-user transformations for the returned representations. Through our experimental evaluation, we show that our defense indeed renders encoder stealing inefficient while preserving downstream utility for legitimate users. Our B4B is therefore a valuable contribution to a safer sharing and democratization of high-utility encoders over public APIs.

7. Towards More Realistic Membership Inference Attacks on Large Diffusion Models

Title	Towards More Realistic Membership Inference Attacks on Large Diffusion Models
Authors	Jan Dubiński, Antoni Kowalczuk, Stanisław Pawlak, Przemysław Rokita, Tomasz Trzciński, Paweł Morawiecki
Conference	IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)
Year	2024

Preface

In the previous chapter, we introduced an active defense against stealing encoder models, addressing the risk of unauthorized replication of machine learning systems. We now shift our focus from protecting the models themselves to safeguarding the data on which they are trained. In large-scale generative models, the training data often contains material of significant value, including copyrighted works or sensitive personal information. Determining whether a specific dataset was used in training is therefore a central question for both privacy protection and intellectual property rights.

One line of work that aims to answer this question relies on membership inference attacks. These methods attempt to determine whether an individual sample was included in the training set of a model, and they have shown promising results in standard supervised learning benchmarks. However, applying them to modern diffusion models presents new challenges. These models are trained on billions of images collected from the internet, and the contribution of any single image to the training process is minimal. As a result, the strong signals observed in smaller settings largely vanish, raising doubts about the reliability of existing evaluation protocols.

In this work, we revisit membership inference attacks in the context of large diffusion models. We first identify the limitations of prior evaluations, which often rely on unrealistic assumptions or toy-scale settings that exaggerate attack success. To provide a more reliable picture, we design a new evaluation framework tailored to the scale of modern generative models. Central to this effort is the construction of a dedicated dataset that allows for a controlled comparison of attack methods, ensuring that conclusions drawn are representative of real-world deployments.

Using this framework, we conduct a series of experiments on Stable Diffusion, evaluating both established membership inference techniques and newly adapted ones. Our results reveal that attacks are far less effective than suggested by earlier studies, highlighting the difficulty of detecting individual training samples in large generative models. At the same time, the findings confirm that privacy and copyright concerns remain unresolved, as existing tools provide limited protection and unreliable auditing capabilities.

This chapter thus marks the beginning of the second part of the dissertation, where the focus lies on the protection of training data in generative models. Building on the lessons learned here, the following chapter introduces a method that moves beyond single-sample membership and instead provides a reliable way to detect the presence of entire datasets in the training of diffusion models.

Abstract

Generative diffusion models, including Stable Diffusion and Midjourney, can generate visually appealing, diverse, and high-resolution images for various applications. These models are trained on billions of internet-sourced images, raising significant concerns about the potential unauthorized use of copyright-protected images. In this paper, we examine whether it is possible to determine if a specific image was used in the training set, a problem known in the cybersecurity community as a membership inference attack. Our focus is on Stable Diffusion, and we address the challenge of designing a fair evaluation framework to answer this membership question. We propose a new dataset to establish a fair evaluation setup and apply it to Stable Diffusion, also applicable to other generative models. With the proposed dataset, we execute membership attacks (both known and newly introduced). Our research reveals that previously proposed evaluation setups do not provide a full understanding of the effectiveness of membership inference attacks. We conclude that the membership inference attack remains a significant challenge for large diffusion models (often deployed as black-box systems), indicating that related privacy and copyright issues will persist in the foreseeable future.

7.1. Introduction

In recent years, there have been rapid advancements in generative modeling techniques within the field of deep learning. Among these, generative diffusion models, particularly those utilising the Stable Diffusion framework, have gained prominence due to their capability to generate high-quality, diverse, and intricate samples. These models hold considerable potential for numerous applications, such as data augmentation, art creation, and design optimization. However, as these models become more widely adopted, addressing the privacy concerns linked to their use is essential. Recently, Getty Images filed a lawsuit against Stability AI, accusing it of *unlawfully copying and processing millions of copyright-protected images* [185]. This lawsuit comes on the heels of a separate case Getty lodged against Stability in the United Kingdom, as well as a related class-action lawsuit that California-based artists filed against Stability and other emerging companies in the generative AI sector [183].

One critical issue that arises in this context is determining whether a specific data point was used during the training process of a model. Extracting this information from a model - known as *membership inference attack* - can be crucial in cases where copyrighted or sensitive data are used without permission, leading to potential legal

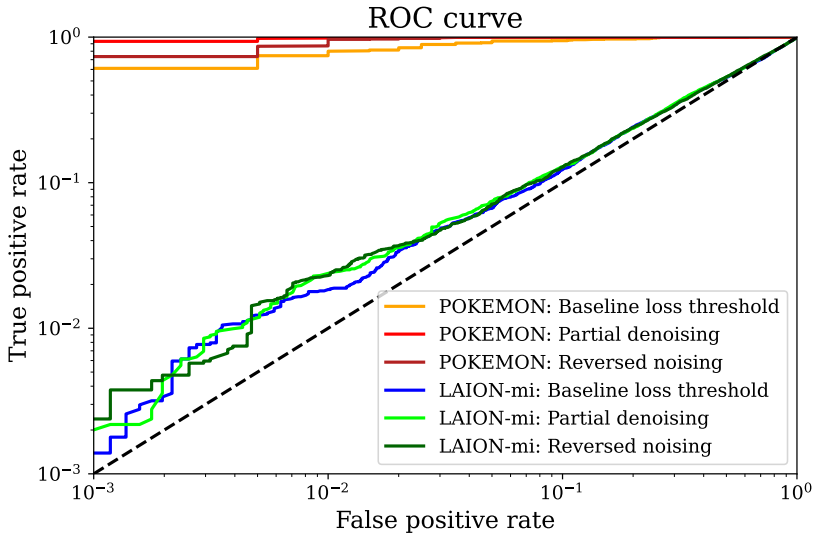


Figure 7.1.1. Pitfalls in the evaluation setting can lead to incorrect conclusions on the effectiveness of membership attacks against large diffusion models such as Stable Diffusion. An exemplary misleading setup involves finetuning the model on a very small dataset with a low internal variance (such as the POKEMON dataset), which gives a remarkable performance for the selected attacks. However, for the proposed new dataset, we observe a drastic performance drop. Our setup does not modify the state-of-the-art Stable Diffusion model but focuses on creating fair membership inference evaluation, possibly close to a real-life usage of the membership attacks.

issues. Although membership inference attacks have been extensively studied in the context of discriminative models [43, 47, 211, 253], the investigation of their effectiveness against generative diffusion models is still in its infancy.

In this paper, we contribute to the understanding of membership inference attacks in large diffusion models, particularly Stable Diffusion. We provide insights into these models, their susceptibility to different membership attacks, and the challenges of their evaluation due to the lack of distinct training and test data. To address these issues, we propose a new dataset for a fair and robust evaluation setup. We conduct attacks against Stable Diffusion and assess their effectiveness. Our findings underscore the complexity of data membership inference in large diffusion models. Our main contributions can be summarized as follows:

- We identify the pitfalls of the existing evaluation of membership inference attacks for large diffusion models.
- We provide a new dataset[1] along with a construction methodology. It allows us to have a more robust evaluation setup for membership inference attacks on the state-of-the-art Stable Diffusion model.
- With the proposed dataset, we thoroughly evaluate a set of membership infer-

[1] `https://drive.google.com/drive/folders/17lRvzW4uXDoCf1v_sIiaMnKGIARVunNU`

ence attacks, which are not prohibitively expensive against Stable Diffusion, including the loss threshold attack and its variants. We also introduce new attacks that focus on modifying the diffusion process to extract more information about membership from the model.

7.2. Background

7.2.1. Diffusion models

Over the past two years, diffusion models [214] have emerged as a novel class of generative models, overshadowing Generative Adversarial Networks [105] by achieving state-of-the-art results on numerous benchmarks [73] and becoming the core technology behind widely popular image generators such as Stablxe Diffusion [24], Midjourney [221], Runway [24], Imagen [193] and DALL-E 2 [180, 181].

In essence, *Denoising Diffusion Probabilistic Models* [119] are probabilistic generative models trained by progressively adding noise to the data and then learning to reverse this process.

During training, a noised image $x_t \leftarrow \sqrt{a_t}x + \sqrt{1-a_t}\epsilon$ is produced by adding Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ to a clean image x, with a decaying parameter $a_t \in [0, 1]$ such that $a_0 = 1$ and $a_T = 0$. The diffusion model f_θ is trained to remove the noise ϵ and recover the original image x by predicting the added noise. This is achieved by stochastically minimizing the objective $\frac{1}{N}\sum_i \mathbb{E}_{t,\epsilon}\mathcal{L}(x_i, t, \epsilon; f_\theta)$, where

$$\mathcal{L}(x, t, \epsilon; f_\theta) = \|\epsilon - f_\theta(x_t, t)\|_2^2 \tag{7.1}$$

Despite being trained with a simple denoising objective, diffusion models have shown the ability to generate high-quality images. The process involves sampling a random vector x_T from a normal distribution $\mathcal{N}(0, I)$ and then applying the diffusion model f_θ to remove the noise from this random image. However, instead of removing all noise at once, the model gradually removes part of the noise iteratively in each generation step.

The final image x_0 is obtained from x_T using a noise schedule σ_t (dependent on a_t), where the model iteratively applies the rule $x_{t-1} = f_\theta(x_t, t) + \sigma_t\mathcal{N}(0, I)$ to $\sigma_1 = 0$. The effectiveness of this process is based on the fact that the diffusion model was trained to denoise images with varying levels of noise. Applying this iterative generation process with large-scale diffusion models yields results that closely resemble real images.

Certain diffusion models are designed to generate specific types of images by incorporating conditional inputs in addition to the noised image. Class-conditional diffusion models utilise a class label, such as "car" or "plane", to generate a desired image class. Text-conditioned models extend this concept by taking the text embedding of a prompt, such as "a photograph of an astronaut riding a horse in space," which is created by a pretrained language encoder like CLIP [179].

7.2.2. Stable Diffusion

Stable Diffusion is the largest and most popular open-source diffusion model [24]. This model is an 890-million-parameter text-conditioned diffusion model trained on 2.3 billion images.

Diffusion models can achieve state-of-the-art synthesis results on image data and other applications. However, the optimisation of powerful diffusion models that operate directly in pixel space can consume hundreds of GPU days, and inference can be expensive due to sequential evaluations. To overcome this challenge, the authors of Stable Diffusion [24] propose applying diffusion models in the latent space of powerful pretrained autoencoders. This approach allows for training and inference on limited computational resources while retaining the quality and flexibility of diffusion models.

Formally, given an image x, the encoder E encodes x into a latent representation $z = E(x)$, and the decoder D reconstructs the image from the latent, giving $\tilde{x} = D(z) = D(E(x))$. To preprocess conditional information y from various modalities (such as language prompts), the Stable Diffusion framework introduces a domain-specific encoder τ_θ that projects y to an intermediate representation. Overall, the Stable Diffusion model is trained by stochastically minimizing the objective $\frac{1}{N}\sum_i \mathbb{E}_{t,\epsilon}\mathcal{L}(z_i, t, \epsilon; f_\theta)$, where

$$\mathcal{L}(z, t, \epsilon; f_\theta) = \|\epsilon - f_\theta(z_t, t, \tau_\theta(y)\|_2^2 \tag{7.2}$$

7.3. Membership inference attack

Membership inference attack [211] answers the question *"was this example in the training set?"*. Currently, two most common approaches are loss based attacks and shadow models.

7.3.1. Loss based attacks

In principle, loss-based membership inference attacks are based on the following simple observation [43]. A model training minimises a loss function on a training set, hence we expect the loss to be lower for training samples than for test ones. In most cases, such methods treat the attacked model a as white-box, assuming that the attacker has access to the model, its source code and trained weights. This assumption is often not met in practice, as API-based generative machine learning services such as Midjourney [221] increase in popularity. In general, methods based solely on the analysis of the loss of the model are less effective than methods utilising shadow models [47, 48].

7.3.2. Shadow models

Membership inference attacks based on *shadow models* involve creating multiple models that imitate the behaviour of the target model, but whose training datasets are known to the attacker. By studying the labelled inputs and outputs of these shadow models, researchers can gain insight into the target model's behaviour and develop attacks that can exploit its vulnerabilities. For diffusion models [48] introduced membership inference attacks called LiRA. This approach involves training a collection of shadow models on random subsets of the training dataset. Once the shadow models have been trained, LiRA computes the loss for each example under each shadow model. By analysing the distribution of losses, LiRA can then determine whether a given example belongs to the training dataset or not. Although shadow models have proven to be a powerful tool in the development of membership inference attacks, this approach has its own disadvantages. Such methods are computationally very expensive, as they require the training of multiple copies of the target model. In particular, for large diffusion models such as Stable Diffusion, the cost of developing multiple shadow models is in practice too high.

7.4. Attack Challenges and Pitfalls for Large Diffusion Models

Lack of nonmembers To perform and evaluate a *membership inference attack* we need two sets: members and nonmembers. Typically, member data samples are drawn from the training set, whereas nonmembers from the test set. Unfortunately, for the Stable Diffusion model, we cannot follow this approach. The original Stable Diffusion model was trained on the data from the LAION-2B EN dataset, a subset of

the LAION-5B dataset [203]. Since the dataset is huge (more than 2 billion images) and was not divided into a test and training set, nonmember samples are not easily available.

Shadow models cost As stated in Section 7.3.2 the shadow models are computationally expensive. The method requires training several dozens of models from scratch. For huge models, such as Stable Diffusion, this approach is practically infeasible. In particular, the cost of training a single Stable Diffusion model is estimated at 600,000$. Moreover, it would take 80.000 A100 GPU-hours to complete the training.

Pitfall 1: Evaluation based on fine-tuning In [79] authors propose to tackle the lack of nonmembers by fine-tuning the Stable Diffusion on a dataset that was not used for training the original model. However, it has been demonstrated that fine-tuning the model can easily lead to overfitting [123]. As shown in [48], better diffusion models are more vulnerable to membership inference attacks: the quality of the generated samples is proportional to the success rate of the attack. This is especially the case for models that overfit to the limited training data during fine-tuning [43], leading to inflated performance and misleading conclusions.

Pitfall 2: Distribution mismatch between members and nonmembers Another approach to dealing with the absence of a natural nonmember dataset is to draw samples from a dataset similar to the training data that was not actually used in the training. However, for a fair evaluation of membership inference attacks, it is important that the members and nonmembers share the same feature distribution. If these two groups can be easily distinguished based on feature distribution mismatch, then there is a high risk that the attack method will learn to distinguish between the features of the two data groups [168] instead of the behaviour of the model on these two sets.

7.5. The LAION-mi dataset

To address the absence of nonmembers samples for the Stable Diffusion model we propose a new dataset consisting of members and nonmembers called LAION-mi. This new dataset aims to facilitate a realistic evaluation of membership inference attacks against large diffusion models. We do not fine-tune nor modify the Stable Diffusion model in any other way, in order to avoid the first pitfall from Sec. 7.4. To mitigate the second pitfall, we apply a sanitization process on the member set. Figure 7.5.1 shows a general scheme of how the LAION-mi dataset is constructed.

7.5.1. Sources of members and nonmembers sets in LAION-mi

Members Stable Diffusion-v1.4 was trained on all data points from the LAION Aesthetics v2 5+ dataset (see Appendix B.3 for details), so all samples from this dataset can serve as member candidates for our new dataset.

Nonmembers To obtain the nonmembers set, we use LAION-2B Multi Translated dataset [23]. This dataset is created by LAION-5B authors from LAION-2B Multi, a subset of LAION-5B, but unlike LAION-2B EN, LAION-2B Multi consists of samples which descriptions are in other languages (not English). LAION-2B Multi Translated is obtained by translating these descriptions to English. Since SD-v1.4 was fine-tuned using samples with an aesthetic score above 5 (obtained by LAION-Aesthetics_Predictor V2 [201]), to build our nonmembers set we also use samples with aesthetic score above 5. The score is precomputed by the LAION-2B Multi Translated dataset authors.

7.5.2. Adapting members and nonmembers sets to ensure the validity of the evaluation setting.

As mentioned before, for a fair evaluation of the membership inference attack we should ensure that the underlying data distribution is the same for the member and nonmember samples. We solve it by introducing the adaptation step for members and nonmembers subsets constituting LAION-mi dataset. During the adaptation, we first deduplicate the nonmembers set (Sec. 7.5.3) and then filter the member set using sanitization process (Sec. 7.5.4). Finally, we obtain members and nonmembers sets which have the same, indistinguishable underlying distribution.

7.5.3. Deduplication

Member samples in the nonmembers source dataset Duplicate samples are images present in a dataset more than once. It has been shown [243] that LAION-2B EN contains approximately 30% duplicates when it comes to the image data. We can expect that the whole LAION-5B contains samples, which are present both in LAION-2B EN and LAION-2B Multi Translated. These samples can potentially contaminate LAION-mi nonmembers subset with members samples and therefore compromise the fairness and correctness of our solution. The following procedure aims to address this issue.

Solution In order to obtain the nonmembers set free of contamination by members samples we perform two-step deduplication. The first step aims to propose a set of duplicate candidates for each nonmember sample. The second step filters out

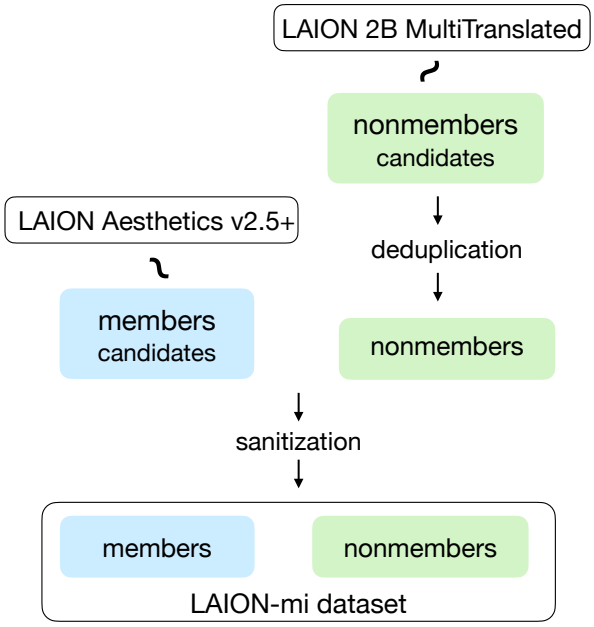


Figure 7.5.1. A general scheme of constructing LAION-mi dataset. First, members and nonmembers are sampled from LAION Aesthetics v2 5+ and LAION-2B Multi Translated datasets, respectively. Then, we remove nonmember that are duplicates of samples from the member set. Finally, to ensure that the distribution of member samples is indistinguishable from nonmembers distribution, we execute an extensive sanitization algorithm on the member set.

nonmembers samples, which we suspect have a duplicate in the members source dataset. We end up with the clean nonmembers set.

Duplicate candidates We need to somehow obtain the samples from the LAION-2B EN dataset which are the most similar to the given sample from the nonmember source. We achieve this by querying the Clip Retrieval Client [41]. This service searches through the LAION-5B dataset and returns the requested amount of samples that are the most similar to the input image. In our approach, we obtain up to 40 duplicate candidates per nonmember sample. One important limitation of this service is that it doesn't distinguish between subsets of LAION-5B. LAION-5B is split into LAION-2B EN and LAION-2B Multi using the CLD3 [233] language identifier on the captions of images. Because we care only about the duplicates from the LAION-2B EN, we need to check if the returned candidate is from this dataset. We use CLD3 to check if the given candidate is from the LAION-2B EN dataset. Only samples from the LAION-2B EN dataset are considered duplicate candidates.

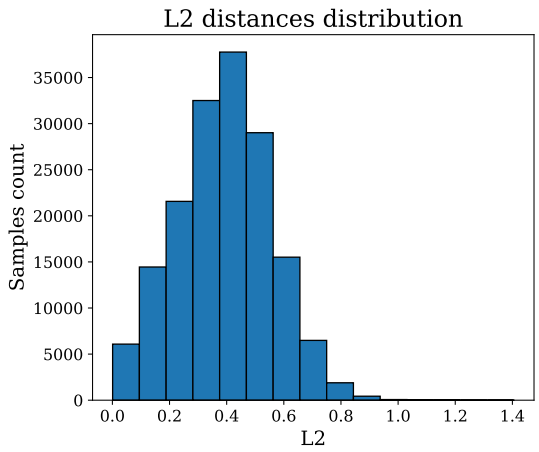
Duplicates detection and filtering When it comes to filtering out the duplicates we propose the following approach: we define the distance metric between the nonmember and its duplicate candidate, and then if the distance is below some

threshold we decide that this sample has a duplicate in the members set, effectively discarding it from the final nonmember set.

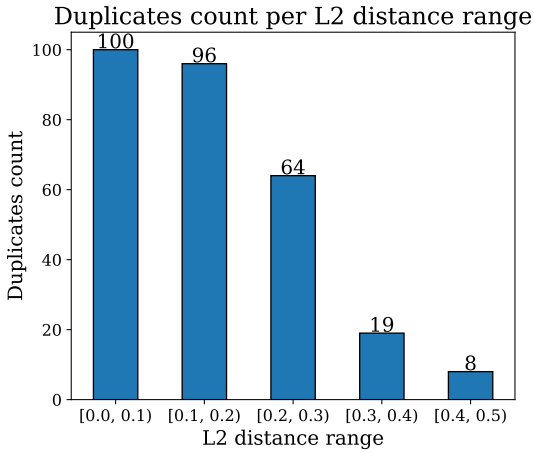
Firstly, for each sample, we calculate the L2 distances of CLIP image embeddings between the sample and all of its duplicate candidates. Then the final duplicate candidate for each sample is the one with the lowest L2 distance score. We then end up with the approximately normal distribution of L2 distances (see Fig. 7.5.2a).

We then pick the threshold below which we reject samples and mark them as duplicates. Our goal here is to filter out as many duplicates as possible, to avoid contamination of the nonmembers set with members samples. At the end of this process, we have less than 1% of the duplicates by setting this threshold at 0.5, using *the rule of three* [127]. We manually confirm it by sampling random 300 samples and checking for the duplicates, without finding any. For the threshold of 0.5 we reject approximately 75% of nonmembers as having a duplicate in the members source dataset. It is a really conservative approach, but as we show next, it is necessary in order to achieve the cleanest nonmembers set possible.

To further confirm that we pick the correct threshold we perform a manual analysis of the duplicates ratio in different L2 score intervals and show the results in Figure 7.5.2b. Since our goal is to make the cleanest nonmembers dataset possible, we pick a threshold of 0.5 to avoid duplicates.



(a) Distribution of L2 distances.



(b) Duplicates count per 100 samples for L2 distances buckets.

Figure 7.5.2. The distribution of the L2 distances between duplicate candidates and the original images approximately follow a normal distribution, with a mean around 0.4 (left). For increasing L2 threshold value the duplicates count decreases sharply, with the interval $[0.4, 0.5)$ containing approximately 8% duplicates and 92% non-duplicates.

7.5.4. Sanitization

Differences between sets As we have stated before, one of the most important challenges of membership attack evaluation is ensuring that the member and nonmember samples come from the most similar distribution possible, in our case both images and their descriptions coming from these subsets should be indistinguishable from each other. However, our source of nonmembers is obtained by translating captions from LAION-2B Multi to English using machine translation[2]. Therefore, we expect the distribution of captions' CLIP [179] embeddings to be different for members and nonmembers sets. We need to address this issue to not fall into the second pitfall, which we line up in the Sec. 7.4.

Assessment approach In order to assess the magnitude of this problem and the efficacy of our sanitization approach we use three metrics. All are based on the CLIP embeddings of the descriptions and images. The metrics are as follows:

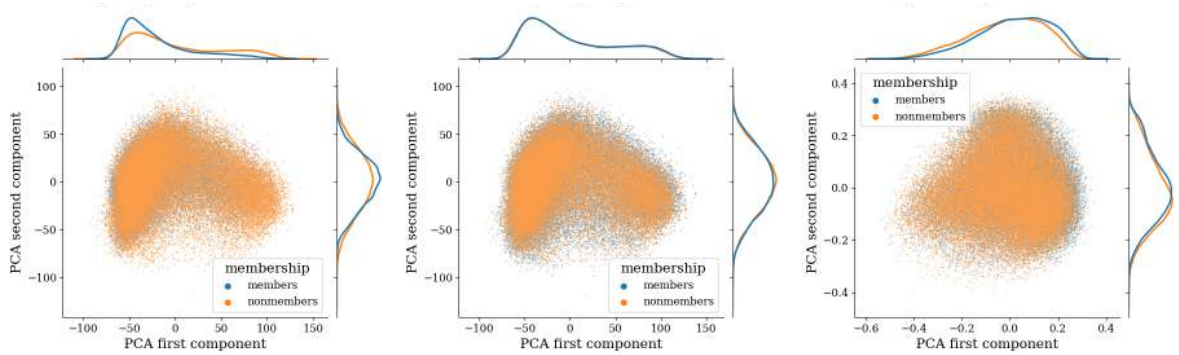
- Fréchet Inception Distance (FID) [115]: in order to compare the resulting metric we compute it for two cases: internal (between two random samples of 10k examples from the same set) and comparative (between 10k random members and 10k random nonmembers). If the difference between these two is significant, we assume that there is a mismatch between distributions.
- Visual analysis of PCA 2D projection: we use PCA decomposition on the embeddings to project them into a 2D space using scikit-learn [11] implementation. The mismatch in the underlying distributions should be indicated by a mismatch of the distributions of the PCA components of the projected embeddings.
- Classification: we train a binary classifier in order to distinguish between embeddings of the descriptions. High accuracy indicates a significant difference between the two sets.

Scale of the problem Our experiments confirm the seriousness of the issue. Firstly, we observe that FID for the comparative case is way greater than for any of the internal cases, see Tab. 7.5.1. Secondly, visual analysis of embeddings projected to 2D space using PCA, Fig. 7.5.3a) confirms our concerns that these text embeddings are in fact different. Finally, a simple logistic regression model separates prompt embeddings with a 90% accuracy.

Santization algorithm The goal of this process is to create a member set that is as similar as possible to our deduplicated nonmembers set.

Main intuition behind the sanitization algorithm (Alg. 1) is to train a set of binary classifiers to label samples as members or nonmembers in an iterative fashion, and then pick only the samples from one of these sets, for which all of the models predict

[2] Facebook's M2M100 1.2B model [93]



(a) Prompts embeddings before sanitization

(b) Prompts embeddings after three iterations

(c) Image embeddings after three iterations

Figure 7.5.3. Sanitization effect on prompts and image embeddings distribution of the members and nonmembers sets. Figure 7.5.3a shows that there is a significant difference between prompts embeddings of nonmembers and members samples before the sanitization process. After three iterations of our sanitization Algorithm 1 these distributions match closely – Figure 7.5.3b. Despite the fact that the algorithm uses only prompts embeddings, we observe in Figure 7.5.3c that image embeddings distributions are also well aligned after the third iteration of our algorithm. We suspect that it is due to the close match between images and their descriptions so that aligning text distributions leads to aligning image distributions as well.

wrong label. In effect, at each iteration, one of these sets becomes closer to the other one in terms of the text embeddings distribution.

In general, we can use both members and nonmembers sets to perform this classifier-based filtering. In our case, we filter only the huge members set (LAION Aesthetics v2 5+ consists of 600M samples, our deduplicated nonmembers set has 42.5k samples). The main reason is that the deduplication process (see Sec. 7.5.3) is a great bottleneck of our system; to get duplicate candidates for all 160k nonmember candidates, we query the retrieval API for 50h.

Table 7.5.1. FID comparison for 10k samples. For text data we calculate FID for CLIP embeddings. To calculate FID for image data we first resize images to 512x512 and then use Pytorch FID implementation [204]. To calculate internal FID we divide the dataset into 2 equal random subsets, each of 10k samples.

	FID	
DATA SUBSET	TEXT	IMAGES
MEMBERS INTERNAL - RANDOM	9.84	7.00
MEMBERS INTERNAL - SANITIZED	9.77	7.06
NONMEMBERS INTERNAL	9.73	7.01
COMPARATIVE - RANDOM	66.43	13.90
COMPARATIVE - SANITIZED	**13.54**	**8.87**

Algorithm 1 Sanitization algorithm

$F \leftarrow \emptyset$ {trained binary classifiers}
$NM \leftarrow$ deduplicated nonmembers
$M \leftarrow$ global set of members
$M_i \leftarrow \emptyset$ {sanitized members after i-th iteration}
$TrainSet \leftarrow \emptyset$ {training dataset}
for $i \leftarrow 1, 2, ..., n$ **do**
  $TrainSet \leftarrow \emptyset$
  if $i = 1$ **then**
    $TrainSet \leftarrow$ sample of size $|NM|$ from M
  else
    $TrainSet \leftarrow M_{i-1}$
  end if
  $TrainSet \leftarrow TrainSet \cup NM$
  $F_i \leftarrow$ trained classifier on $TrainSet$
  while $|M_i| < |NM|$ **do**
    $M_{tmp} \leftarrow$ sample from M
    for $j \leftarrow 1, 2, ..., i$ **do**
      if F_j predicts member label for sample **then**
        $M_{tmp} \leftarrow M_{tmp} \setminus sample$
      end if
    end for
    $M_i \leftarrow M_i \cup M_{tmp}$
  end while
end for
$SM \leftarrow M_n$ {final sanitized members set}

Results To obtain our final set of 40k sanitised members we apply the algorithm for three iterations. To obtain this subset we filter approximately 5M samples from LAION Aesthetics v2 5+ dataset, which takes only 2h on a single NVidia RTX 2080Ti. Using our assessment methodology we confirm its efficacy. FID score in the comparative case drops significantly compared to the starting members set. We see that the PCA components' distribution align (see Fig. 7.5.3b) and a binary classifier's accuracy is almost random.

Images embeddings Regarding the image embeddings for LAION-2B EN and LAION-2B Multi Translated, we expect them to have the same characteristics as they come from the same source. This assumption is confirmed by a low FID value between them, see Tab. 7.5.1. Additionally, the FID value decreases even further after the text-focused sanitization and a binary classifier's accuracy is almost random. Therefore, additional sanitization efforts focused on image embeddings are not necessary, and the PCA components are also aligned, see Fig. 7.5.3c).

7.6. Experiments

7.6.1. Threat model

A membership inference attack is defined as follows. We consider an adversary A that aims to infer whether a single data point x was included in the training set D of a generative model M. The attacker has no knowledge about the dataset D and is only able to query the model M. We distinguish three scenarios according to the attacker's capabilities.

- In the **black-box** scenario, an adversary queries a generative model with a text prompt and gets a generated image. The attacker has no knowledge about the model architecture and no access to its weights.
- In the **grey-box** scenario an adversary has access to the visual and text encoders of the attacked model. Thus, they are able to calculate the latent representation (embedding) of a given image and text prompt. However, the attacker still has no access to the model weights.
- Finally, in the **white-box** scenario, an adversary has full access to the model, its source code and trained weights.

We start with a baseline white-box model loss threshold attack, which is based on the fact that machine learning models learn by minimising the loss in the training samples. We extend our analysis by covering metrics related to model inference. Moreover, we introduce new white-box attack methods and show that they outperform the commonly used baseline method. We also describe and evaluate a grey and a black-box scenario. The attacks are based on the intuition that generative models tend to synthesise samples similar to their training set. For all attacks, we evaluate a variant in which the losses are obtained as the average of 5 losses (5 different passes through the model, each time with a different noise) following the findings of [47]. We explore more attack methods in the Appendix B.4.

7.6.2. Threshold attack

A general *threshold* attack is formulated as follows. For a selected threshold τ, the attack classifies the image x as a member if $\mathcal{L} < \tau$. Otherwise, x belongs to the nonmember set. Commonly used *threshold* attacks focus only on a model loss. We extend our analysis by Pixel and Latent error, defined as follows:

Model loss We monitor the loss of the diffusion model given by Eq. 8.2 $\mathcal{L}(x,t,\epsilon;f_\theta) = \|\epsilon - f_\theta(x_t,t)\|_2^2$.

Pixel error We define the pixel error as the reconstruction error between an original image x and the generated image x' defined as $\mathcal{L}(x,x') = \|x - x'\|_2^2$.

Latent error This measurement is similar to the pixel error. However, it focuses on the reconstruction error between a latent representation z of the original image x and the latent representation z' generated by the diffusion model. The error $\mathcal{L}(z,z')$ is defined as $\|z - z'\|_2^2$.

7.6.3. Attack methods

We present a baseline and the best-performing attack methods evaluated in our experiments. Most methods are only applicable under the white-box scenario but we also examine the attacks in the grey- and black-box scenario. An exhaustive description and analysis of these different attack methods are given in Appendix B.4.1.

Baseline loss threshold In [48] the authors show that evaluating the model loss at timestep 100 for the latent with applied noise scale α_t at $t = 100$ yields the best results for the membership inference attacks based on model loss. We follow this method and evaluate the model loss at timestep 100 for the latent noised with scale α_{100}.

Methods exploiting the noise There are many possible variants of adding or removing noise in the diffusion process before we make a final decision based on the loss. The intuition (or hopeful assumption) behind such attacks is that member samples would behave more robustly than nonmembers under noisy conditions. We explore many different settings and refer a reader to Appendix B.4.1 for details.

Generation from prompt To perform this attack we pass only the prompt associated with the original images to the model. We do so to simulate a real-world scenario, where we have access to the model only via the API. In the black-box scenario, we calculate the Pixel error between the original image and the image generated by the model using the default method of 50 timesteps. In the grey-box scenario, we obtain the generated images in the same way as in the black-box scenario, but then we calculate the latent representation of the original and generated images using the visual encoder of the attacked model and then calculate the latent error between them.

7.6.4. Targeted datasets

LAION-mi In our paper we use our LAION-mi dataset proposed in Section 7.5. For each attack, we use the same subset of 5000 member samples and 5000 nonmembers samples, further referred to as the attack set. We find that different

training and evaluation set splits can produce significantly different results, some almost 10 times better than others (see Appendix B.5). Following these findings, we evaluate our attacks on 100 random subsets (evaluation sets) of 500 members and 500 nonmember samples drawn from the attack set and then report the mean and the standard deviation of the performance.

POKEMON POKEMON dataset [175] is a Text2Image dataset. We use it to first finetune the original StableDiffusion v1.4 model using a subset of 633 samples (members), leaving the remaining 200 samples as nonmembers. We evaluate our attacks on 1000 subsets obtained from random 200 member and all nonmember samples. We also conduct an analysis of the influence of overfitting on the attack performance in Appendix B.6.

7.7. Results

We evaluate the attacks for two setups. First, we attack the Stable Diffusion v1.4 model, which we do not modify in any way. We draw data samples of members and nonmembers from our LAION-mi dataset. Then, we evaluate the effectiveness of the attacks on the same model, which is finetuned on the POKEMON dataset [175]. Here, we use test and training data splits (633/200 samples) as member and nonmember sets.

Metric A metric to evaluate the membership attack is true-positive rate (TPR) calculated at a low false-positive rate (e.g. FPR=1%). For privacy-related problems, it is a much better metric than common aggregate metrics, such as accuracy or AUC [47].

Discussion In Table 7.7.1 we observe a severe discrepancy in the effectiveness of the attacks achieved for the LAION-mi dataset and fine-tuning on POKEMON. In the second case, two white-box attacks (*reversed noising* and *partial denoising*) achieve a very high TPR. However, when the attacks are applied against our LAION-mi dataset, we observe a huge drop in performance. Here we clearly see the effects of the first pitfall from Sec. 7.4, namely the fine-tuned model overfits, and the membership inference task becomes trivial. We explore this topic further in Appendix B.6. The obtained results demonstrate that evaluation on a small dataset (used for finetuning the model) is misleading and a more careful setup, such as our proposal, is required.

Moreover, the results show the limited performance of loss-based attacks in the black- and grey-box scenarios, even for the simple POKEMON setting. This highlights an important issue, as many image-generation services work as a black-box API. As mentioned in 7.3.1 this trend is unlikely to change in the future.

In theory, identifying training samples in black-box scenarios can be approached

Table 7.7.1. *Threshold* membership inference attacks results on LAION-mi and POKEMON datasets. We demonstrate the importance of evaluating membership inference attacks in a fair setting. On the POKEMON dataset some of the attacks are almost perfect, with *partial denoising* reaching **99.5**% TPR@FPR=1%, but on ours LAION-mi dataset with original SD-v1.4 we reach at most **2.51**%. Our proposed methods outperform the *Baseline loss threshold* method.

Scenario	Loss	Method	TPR@FPR=1%. ↑ LAION-mi	TPR@FPR=1%. ↑ POKEMON
White-box	Model Loss	Baseline loss thr.	1.92%±0.59	80.9%±2.27
		Reversed noising	2.51%±0.73	97.3%±0.93
		Partial denoising	2.31%±0.61	94.5%±1.34
		Reversed denoising	2.25%±0.64	91.5%±1.63
	Latent Error	Reversed noising	1.26%±0.62	11.5%±1.84
		Partial denoising	2.42%±0.62	99.5%±0.4
		Reversed denoising	2.17%±0.64	61.1%±2.74
	Pixel Error	Reversed noising	1.90%±0.51	8.36%±1.66
		Reversed denoising	2.03%±0.55	12.0%±1.97
		Partial denoising	1.75%±0.68	25.38%±2.55
Grey-box	Latent Error	Generation from prompt	0.93%±0.41	7.15%±1.5
Black-box	Pixel Error	Generation from prompt	0.35%±0.19	12.0%±1.9

by extracting training samples from the model, as in [48]. However, this approach requires the generation of hundreds of images per text prompt, which is computationally expensive. This approach is also limited to identifying training samples that have been memorised by the model. For those reasons, such methods are not well suited for identifying the membership of a sample. For the white-box scenario, the state-of-the-art approach is a method based on the shadow models. However, as stated in 7.3.2, this strategy is too costly for large diffusion models such as Stable Diffusion. We further discuss the applicability of shadow models in Appendix B.7.

7.8. Conclusion

We showed that evaluation of membership inference attacks with the model finetuning approach may lead to false conclusions. As an alternative, we proposed a new carefully crafted dataset, which mitigates the main limitation of the original LAION dataset, which is a lack of a test set. Having the proposed dataset and reliable set of nonmembers, we evaluated several membership inference attacks and obtained results, which contradict previous findings.

Our dataset could help the community evaluate attacks on large generative

diffusion models such as Stable Diffusion in a more rigorous and fair setting. A clear picture of how successful the membership attacks are is essential for a sound policy on matters such as data ownership and privacy. We argue that for large diffusion models, where shadow models are prohibitively expensive, membership inference remains a very challenging task.

8. CDI: Copyrighted Data Identification in Diffusion Models

Title	CDI: Copyrighted Data Identification in Diffusion Models
Authors	Jan Dubiński*, Antoni Kowalczuk* Franziska Boenisch, Adam Dziedzic
Conference	The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
Year	2025

Preface

In the previous chapter, we examined the limitations of membership inference attacks when applied to large diffusion models. While these methods can, in principle, reveal whether a single sample was included in training, our analysis showed that they fail to provide reliable results at scale. This raises a crucial question: how can data owners verify whether their collections were used in the training of modern generative systems? Since such collections often consist of thousands or millions of items, it is more natural to ask about the presence of entire datasets rather than individual examples.

In this chapter, we propose a method tailored to this dataset-level perspective. The key observation is that data owners usually control sets of samples, not isolated points, and that weak signals from individual membership tests can be aggregated into a strong, reliable indicator when considered collectively. Building on this idea, we design Copyrighted Data Identification (CDI), a framework that combines membership signals across multiple samples and subjects them to a statistical testing procedure. This approach produces a robust measure of whether a given dataset was part of the training data of a diffusion model.

We validate CDI across a range of diffusion architectures and diverse datasets. The results show that CDI achieves high reliability in detecting copyrighted datasets, significantly outperforming existing approaches. Importantly, it does so without requiring white-box access to the model, making it suitable for real-world settings where generative models are often deployed behind black-box APIs.

By reframing the problem from single-sample membership to dataset identification, this chapter provides a practical tool for defending the rights of data owners in the era of large-scale generative models. In the final part of this dissertation, we extend the analysis beyond diffusion architectures to the emerging class of image autoregressive models, examining their particular privacy risks and adapting our defenses accordingly.

Abstract

Diffusion Models (DMs) benefit from large and diverse datasets for their training. Since this data is often scraped from the Internet without permission from the data owners, this raises concerns about copyright and intellectual property protections. While (illicit) use of data is easily detected for training samples perfectly re-created by a DM at inference time, it is much harder for data owners to verify if their data was used for training when the outputs from the suspect DM are not close replicas. Conceptually, membership inference attacks (MIAs), which detect if a given data point was used during training, present themselves as a suitable tool to address this challenge. However, we demonstrate that existing MIAs are not strong enough to reliably determine the membership of individual images in large, state-of-the-art DMs. To overcome this limitation, we propose `Copyrighted Data Identification` (`CDI`), a framework for data owners to identify whether their *dataset* was used to train a given DM. `CDI` relies on *dataset inference* techniques, i.e., instead of using the membership signal from a single data point, `CDI` leverages the fact that most data owners, such as providers of stock photography, visual media companies, or even individual artists, own datasets with multiple publicly exposed data points which might all be included in the training of a given DM. By selectively aggregating signals from existing MIAs and using new handcrafted methods to extract features from these datasets, feeding them to a scoring model, and applying rigorous statistical testing, `CDI` allows data owners with as little as 70 data points to identify with a confidence of more than 99% whether their data was used to train a given DM. Thereby, `CDI` represents a valuable tool for data owners to claim illegitimate use of their copyrighted data. We make our code available at `https://github.com/sprintml/copyrighted_data_identification`.

8.1. Introduction

In recent years, large diffusion models (DMs) [214] have rapidly gained popularity as a new class of generative models, surpassing the performance of prior approaches, such as Generative Adversarial Networks [105]. DMs now power several state-of-the-art image generators including Stable Diffusion [24], Midjourney [221], Runway [24], Imagen [193], and DALL-E 2 [180, 181].

To reach their powerful performance, DMs need to be trained on large amounts of high-quality and diverse data. This data is usually scraped from the Internet, often without respecting the copyrights of the data owners. Especially since it has been shown that DMs are capable of generating verbatim copies of their training

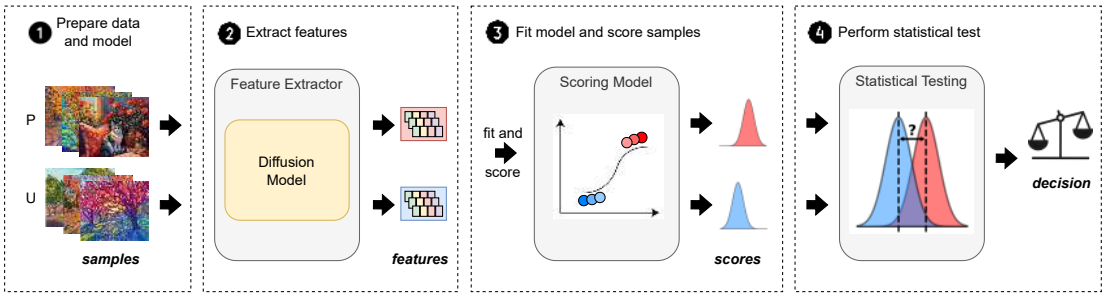


Figure 8.1.1. Overview of our Dataset Inference method for Diffusion Models from our previous work. Our approach consists of the following stages: ❶ Prepare the query data to verify if the *published* suspect samples $\mathbf{P}$ were used to train the DM. The *unpublished* samples $\mathbf{U}$ from the same distribution as $\mathbf{P}$ serve as the validation set. ❷ Run inference on all the inputs $\{\mathbf{P}, \mathbf{U}\}$ to extract their membership features. Use current MIAs and our handcrafted features. ❸ Find useful features and learn a discriminator. $\mathbf{P}$ and $\mathbf{U}$ sets are split into $\mathbf{P}_{\text{ctrl}}$, $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{ctrl}}$, $\mathbf{U}_{\text{test}}$. The features for $\mathbf{P}_{\text{ctrl}}$ and $\mathbf{U}_{\text{ctrl}}$ are used to train a scoring model to selectively combine features and differentiate between the samples from $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$. ❹ Apply a statistical t-test to verify if the scores obtained for public suspect data point $\mathbf{P}$ are statistically significantly higher than scores for $\mathbf{U}$, in which case, $\mathbf{P}$ is marked as being a part of the DM's training set. Otherwise, the test is inconclusive and the DM's training set is resolved as independent of $\mathbf{P}$. The method

data at inference time [48], this represents a violation of intellectual property rights. Recently, Getty Images, a leading visual media company, filed a lawsuit against Stability AI, the creators of Stable Diffusion, alleging the unauthorized use of copyright-protected images [28, 185]. This case has sparked a wave of additional lawsuits, with many more now addressing intellectual property infringement by generative AI companies [183, 184]. Unfortunately, as it becomes obvious during the lawsuits—particularly for training data points that are not output in a verbatim form during inference time—verifying that these data points have been illegitimately used for training the DMs is a challenging task.

Membership inference attacks [211] that aim at identifying whether a specific data point was used to train a given model, in theory, present themselves as a solution to the problem. Unfortunately, prior work [33] indicates that performing a realistic MIA on large DMs is a very challenging task. One of the practical challenges lies in the prohibitive costs of training state-of-the-art DMs (*e.g.,* $600.000 for Stable Diffusion) which renders potent MIAs utilizing multiple *shadow model* copies [47, 211] infeasible. To further explore the practicality of MIAs for identifying copyrighted samples used to train large DMs, we perform an extensive study, evaluating the success of existing MIAs against DMs' training data for various open DMs. Our findings demonstrate that **current MIAs for DMs are limited in confidently identifying DMs' training data points in case of models trained on large datasets**—showcasing that individual MIAs cannot reliably support copyright claims.

In light of this result, we, however, observe that in most cases, data owners, such as stock photography, visual media companies, or even individual artists, typically seek to verify the use of not just a single data point but a collection of their work as training data for a given DM. This moves the idea of *dataset inference* [157] (DI) into focus. DI was first proposed to detect stolen copies of supervised classifier models and then subsequently extended to self-supervised models [86]. It leverages the observation that, while MIAs on individual data points do not produce a strong signal, selectively aggregating signals across a subset of the training dataset and applying statistical testing can reveal a distinct signature of the model. This dataset-based signature allows for the detection of stolen model copies with a confidence level exceeding 95%. Yet, to date, it remains unexplored whether the principles of DI actually transfer to DMs and are suitable to identify subsets of their training data rather than resolving model ownership—given the vast amount of heterogeneous data DMs are initially trained on. Additionally, it is unclear how large the required training data subsets for verification would have to be. Finally, we do not know the specific features needed to extract a strong signal over the training data.

To close these gaps, **we propose** `Copyrighted Data Identification`**, designed to answer the critical question:** ***Was this DM trained on a copyrighted collection of images?*** The overall schema of our method is illustrated in Fig. 8.1.1. To design `CDI`, we move beyond simply aggregating features extracted by existing MIAs, as these often produce signals that are too weak to achieve highly confident DI, rendering the approach impractical. Instead, we firstly, extend the feature extraction methods by our newly proposed features. Secondly, we design a scoring function which maps the extracted information into sample membership probability, learning the features relevant for each DM. Finally, in contrast to MIAs which usually refer to metrics like True Positive Rate (TPR) or Area Under Curve (AUC) which do not give any confidence estimate, we equip `CDI` with rigorous statistical testing as the final component.

We demonstrate the success of our method on diverse large-scale DM architectures (LDM [24], DiT [174], U-ViT [40]), including unconditioned, class-conditioned and text-conditioned models, trained on various image resolutions. Our results show that `CDI` achieves a confident detection rate of data (illegitimately) used for training DMs. `CDI` remains effective when only **a part of the investigated data was actually used in DM training**. Moreover, we demonstrate that `CDI` does not yield false positives, making it a reliable tool for detecting and confidently claiming the use of copyrighted data in DMs.

In summary, we make the following contributions:

- We demonstrate that existing MIAs for DMs show limited effectiveness in confidently identifying the training data points of large, state-of-the-art models.
- To address this issue, we propose CDI, a method that empowers data owners to identify whether their data has been (illegitimately) used to train a DM, incorporating rigorous statistical testing to ensure confidence in the results.
- We perform thorough feature engineering to amplify the signal in CDI, proposing novel feature extraction methods and enabling data owners even with smaller datasets to benefit from our method.
- We evaluate CDI on a wide range of DMs and their pre-training datasets and provide a unified open-source codebase[1] with a common interface to all prior MIAs and our new CDI, serving as a valuable evaluation testbed for the community.

8.2. Background

Diffusion Models [119, 216] are generative models trained by progressively adding noise to the data and then learning to reverse this process. The forward diffusion process adds Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ to a clean image x in order to obtain a noised image $x_t \leftarrow \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon$, where $t \in [0, T]$ is the diffusion timestep, and $\alpha_t \in [0, 1]$ is a decaying parameter such that $\alpha_0 = 1$ and $\alpha_T = 0$. The diffuser f_θ is trained to predict the ϵ for various timesteps, by minimizing the objective $\frac{1}{N}\sum_i \mathbb{E}_{t,\epsilon}\mathcal{L}(x_i, t, \epsilon; f_\theta)$, where N is the training set size, and

$$\mathcal{L}(x, t, \epsilon; f_\theta) = \|\epsilon - f_\theta(x_t, t)\|_2^2. \tag{8.1}$$

The generation iteratively removes the noise prediction $f_\theta(x_t, t)$ from x_t for $t = T, T-1, ..., 0$, starting from $x_T \sim \mathcal{N}(0, I)$, and obtaining a generated image $x_{t=0}$. To guide this process, for conditional image generation the diffuser f_θ receives an additional input y, which represents a class label for the class-conditional DMs [119] or a text embedding, obtained from a pretrained language encoder like CLIP [179], for the text-to-image DMs [24, 181, 193].

Latent diffusion models [24] (LDMs) improve DMs by conducting the diffusion process in the latent space, which significantly reduces computational complexity, making training scalable and inference more efficient. For the LDMs, the encoder $\mathcal{E}$ transforms the input x to the latent representation $z = \mathcal{E}(x)$ and Equation 8.1 becomes

[1] https://github.com/sprintml/copyrighted_data_identification

$$\mathcal{L}(z,t,\epsilon;f_\theta) = \|\epsilon - f_\theta(z_t,t)\|_2^2. \tag{8.2}$$

Membership Inference Attacks. MIAs aim to determine whether a specific data point was used to train a given machine learning model [211]. Extensive research has explored MIAs against supervised machine learning models [16, 43, 47, 226]. On the high level, MIAs operate on the premise of overfitting, assuming that training data points (members) exhibit smaller training loss compared to data points not encountered during training (non-members). Initial MIAs against DMs [19] focus on assessing the membership of samples by evaluating the model's noise prediction loss. Their findings establish that the loss value at the *diffusion timestep* $t = 100$ proves most discriminative between member and non-member samples. Intuitively, if t is too small ($t < 50$), the noisy image resembles the original, making the noise prediction too easy. Otherwise, if t is too large ($t > 300$), the noisy image resembles random noise, making the task overly challenging. Among the recent MIA approaches targeting DMs, the Step-wise Error Comparing Membership Inference (SecMI) attack [79] infers membership by estimating errors between the sampling and inverse sampling processes applied to the input x also at timestep $t = 100$. Following the same overfitting principle, the Proximal Initialization Attack (PIA) [137] enhances SecMI by assessing membership based on the difference in the model's noise prediction for a clean sample x at timestep $t = 0$ and a noised sample x_t at $t = 200$, where the method was found to be most discriminative.

Protecting Intellectual Property in DMs Protecting intellectual property (IP) in DMs involves safeguarding against unauthorized usage of trained models and attributing generated data to their source models, while also protecting the IP of the data used for training. Several attribution methods focus on watermarking at both the model and input levels, embedding invisible watermarks into generated images or subtly influencing the sampling process to create model fingerprints [98, 150]. Other techniques explore fingerprinting methods, where unique patterns or signals are embedded into generated data for identification purposes [206, 255]. However, those methods protect the IP in trained models and generated data, leaving the IP of *training data* out of scope. To solve this issue, various approaches aim to protect against style mimicry and unauthorized data usage by adding perturbations to images or detecting unauthorized data usage through injected memorization or protective perturbations [104, 229, 236, 242, 251]. However, existing methods have important drawbacks, such as limiting the data usage for consensual applications and providing no protection if the data IP has already been breached. Moreover, a

malicious party may attempt to overcome the safety mechanism by image purification methods [45]. Our proposed method fills in those gaps by enabling data owners to identify whether their data has been illegitimately used for training, without any requirements to modify the protected content. While the previous work showed the possibility of computing the influence of the training data points on the generated outputs [265], we propose to go a step further and exactly detect which data points are used for training.

8.3. Limitations of MIAs in Member Detection

We rigorously evaluate existing MIAs to test their ability to detect training members in large, complex DMs. Prior studies [79, 137] reported success with MIAs in accurately identifying DMs training members; however, these results were often based on toy models or datasets (*e.g.,* CIFAR100 [141]) that do not reflect the complexity of high-dimensional, diverse DM setups. Our analysis on state-of-the-art DMs trained on extensive datasets (*i.e.,* ImageNet-1k [71] or COCO [231]) reveals significant performance limitations of existing MIAs and key factors that contributed to overestimated effectiveness in previous work. Full details are provided in section C.4.

8.3.1. Evaluated MIAs

1. *Denoising Loss* [48]: The loss is computed from Equation 8.2 five times for the diffusion timestep $t = 100$, as indicated in the original paper. The final membership score is the average loss, where a lower value indicates that the sample is a member.
2. $SecMI_{stat}$ [79]: The membership score extracted by SecMI aims to approximate the posterior estimation error of f_θ on the latent z (obtained from the image encoder part), claiming it should be lower for members than for non-members.
3. *PIA* [137]: The score extracted by PIA aims at capturing the discrepancy between the noise prediction on a clean sample's latent z and the noise prediction on its noised version z_t at time t. This discrepancy should be lower for members.
4. *PIAN* [137]: This MIA is an adaptation of the original PIA to further strengthen the membership signal. The noise prediction on z is normalized, so it follows the Gaussian distribution. Similar to PIA, the scores returned from PIAN are expected to be lower for members than for non-members.

8.3.2. Experimental Setup

Models. We evaluate class-conditioned, as well as text-conditioned state-of-the-art DMs of various architectures, namely LDM [24], U-ViT [40], and DiT [174]. We employ already trained checkpoints provided by the respective papers [18, 22, 26]. For class-conditioned generative tasks, LDM offers one model checkpoint with the resolution of 256x256 (LDM256). For U-ViT and DiT, we have access to models operating on resolutions of 256x256 and 512x512 (U-ViT256, U-ViT512, DiT256, DiT512). Additionally, we conduct experiments on text-conditioned models based on U-ViT architecture (U-ViT256-T2I, U-ViT256-T2I-Deep) and a newly trained unconditional U-ViT256-Uncond model (see App. C.9).

Datasets. For the class-conditional evaluation, we use models trained on ImageNet-1k [71]. This dataset contains large-sized colored images with 1000 classes. There are 1,281,167 training images and 50,000 test images. For text-conditional task evaluation, we use models trained on COCO-Text dataset [231], a large-scale object detection, segmentation, and captioning dataset which contains 80,000 training images and 40,000 test images, each with 5 captions.

8.3.3. Performance of MIAs in Member Detection

Our results indicate that the existing MIAs achieve performance comparable to random guessing. We present the aggregated max and average **TPR@FPR=1%** across 8 DMs in table 8.3.1, and defer the full evaluation to App. C.21. For completeness, we provide AUC (Table C.21.3), accuracy (Table C.21.2), and ROC curves (Fig. C.21.2) of MIAs there.

Table 8.3.1. TPR@FPR=1% for MIAs. Performance of existing MIAs in identifying training members is limited.

Attack	Max TPR@FPR=1%	Mean TPR@FPR=1%
Denoising Loss [47]	2.24	1.61
SecMI$_{stat}$ [79]	2.44	1.50
PIA [137]	5.57	2.18
PIAN [137]	1.53	1.03

8.4. Our CDI Method

Recognizing the limitations of MIAs on large, state-of-the-art DMs, we shift our focus to DI and introduce our CDI method. To achieve reliable and confident detection of data collections used in model training, we go beyond simply aggregating

features from existing MIAs. Our CDI consists of four stages: (1) data and model preparation, (2) feature engineering and extraction, extended by our three newly proposed detection methods (3) a scoring function that maps these features to scores, and (4) a rigorous statistical hypothesis testing, enabling high-confidence decisions. We visualize and describe CDI in Figure 8.1.1.

Dataset Inference. DI was initially introduced as a tool for detecting model stealing attacks [12]. In the context of supervised models [157], DI involves crafting features for a set of training data points, inputting them into a binary classifier, and applying statistical testing to establish model ownership. The features of supervised learning are based on the fact that classifiers are trained to maximize the distance of training examples from the model's decision boundaries while test examples typically lie closer to these boundaries, as they do not influence the model's weights during training. DI was extended to self-supervised learning (SSL) [86] by observing that training data representations exhibit a markedly different distribution from test data representations. Building on this intuition, we design specific features based on the DM's behavior for a set of data points that we want to test for potential (illegitimate) use in training the DM. We then map those features to scores on which we apply statistical testing. Unlike traditional DI, which focuses on ownership resolution for the entire model, our approach is tailored for data verification, allowing owners of small subsets of the DM's training data to verify their use in model training.

Notation and Setup. We denote $\mathbf{P}$ as a set of samples that we suspect to be (illegitimately) used for training the DM. Those are *published* samples provided by the data owner who wants to make a claim for their intellectual property. We refer to $\mathbf{U}$ as a set of *unpublished* samples, from the same distribution as $\mathbf{P}$, that serves as the validation set. In real scenarios, $\mathbf{U}$ might come from a creator's unpublished data or sketches of their released work. We assume $\mathbf{P}$ to be *i.i.d.* with U.

Data Preparation and Processing. We split $\mathbf{P}$ and $\mathbf{U}$ into $\mathbf{P}_{\text{ctrl}}$, $\mathbf{P}_{\text{test}}$, and $\mathbf{U}_{\text{ctrl}}$, $\mathbf{U}_{\text{test}}$. We extract the final full set of features for $\mathbf{P}_{\text{ctrl}}$ and $\mathbf{U}_{\text{ctrl}}$ and train the scoring model s to tell apart members from non-members, such that s eventually outputs higher values when presented with a member. Then, we apply s to the features extracted from $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$. Finally, we perform statistical testing to find whether the scores returned by s on $\mathbf{P}_{\text{test}}$ are significantly higher than those on $\mathbf{U}_{\text{test}}$, which would indicate that $\mathbf{P}$ was, indeed, used to train the DM.

Threat Model. We design CDI as a tool for use in legal proceedings. Consequently, the CDI procedure is carried out by a third trusted party, referred to as an *arbitrator*. The arbitrator is approached by a victim, whose private data might have been

potentially used in a training of a DM. The arbitrator executes CDI either in the gray-box model access, (can only obtain outputs, *i.e.,* noise predictions, for given inputs to a DM at an arbitrary timestep t) or in the white-box model access (where DM's internals and parameters can be inspected). The access type depends on the requirements of the features used in CDI (we provide more details in App. C.14).

8.4.1. Features

We utilize MIAs (section 8.3.1) as the source of features for CDI. Additionally, to increase the discriminative capabilities of our CDI, we propose the following three novel features that can be extracted from a DM to provide additional information on a sample's membership score. Our final feature extractor implements a function $f_e : \mathcal{R}^{C\times H\times W} \rightarrow \mathcal{R}^k$, with C, H, W denoting the channels, height, and width of an input sample, respectively, and k being the dimensionality of the extracted feature vector.

Gradient Masking (GM). This feature aims at capturing the difference in the ability to restore destroyed semantic information between members and non-members. It is inspired by the *Degrade, Restore, Compare* (DRC) idea from Fu et al. [101] who identify that for members, a restoration is more successful. To compute the feature, we first capture the gradient $\mathbf{g} = |\nabla_{z_t}\mathcal{L}(z_t, t, \epsilon; f_\theta)|$. Intuitively, $\mathbf{g}$ indicates the influence each feature value in the latent z_t has on the loss $\mathcal{L}$. We are interested in the features from z_t that exhibit the highest influence on the loss. Therefore, we create a binary mask $\mathbf{M}$ for the top 20% values in $\mathbf{g}$. This mask indicates significant regions of the latent representation z_t. Next, we obtain $\hat{z_t} = \epsilon \cdot \mathbf{M} + z_t \cdot \neg\mathbf{M}$, with the significant values of z_t destroyed by replacing them with random noise $\epsilon \sim \mathcal{N}(0, I)$, and the rest left unchanged. Finally, we compute $||(\epsilon - z_t)\cdot\mathbf{M} - f_\theta(\hat{z_t}, t)\cdot\mathbf{M}||_2^2$ as the feature. This feature expresses the reconstruction loss over the semantically most relevant regions and should be lower for members. We calculate the feature at multiple diffusion timesteps t, to further strengthen the signal. Note that we differ in our feature computation from Fu et al. [101] in two significant aspects. (1) While their DRC employs powerful third-party self-supervised vision encoders like DINO [49] to identify semantically significant regions, we utilize only the information from within the DM to obtain the mask $\mathbf{M}$. (2) Additionally, we utilize the model's loss as our final signal, instead of computing cosine similarity between representations returned from DINO for clean and restored samples, rendering our method more self-contained and independent of the signal from other models.

Multiple Loss (ML). To increase the membership signal from the model prediction loss, we compute Eq. 8.2 at multiple (10) diffusion timesteps $t = 0, 100, ..., 900$ to provide more information to train the scoring function s.

Noise Optimization (NO). We leverage an insight initially observed in supervised classifiers, namely that the difficulty of changing the predicted label of a sample through adversarial perturbations differs between members and non-members [146]. In particular, it takes a stronger perturbation to change the prediction for members. The reason is that ML models return more confident predictions on training samples (members). To craft our feature, we adapt this intuition to DMs. We note that perturbing a noised sample z_t to minimize the model noise prediction loss expressed in Equation 8.2 achieves better results, *i.e.,* lower loss values for member samples. Specifically, we conduct an unbounded optimization of the perturbation δ applied to the noised latent representation z_t at timestep $t = 100$. Our objective function is defined as: $\arg\min_\delta ||\epsilon - f_\theta(z_t + \delta, t)||_2^2$. To optimize this objective, we employ the 5-step L-BFGS algorithm [7] (which is commonly used to generate adversarial perturbations [46, 218]). We use the resulting values of the noise prediction error $||\epsilon - f_\theta(z_t + \delta, t)||_2^2$ and the amount of perturbation $||\delta||_2^2$ as features.

We provide further analysis of our features in App. C.20.

8.4.2. Scoring Model

Based on the set of features extracted for $\mathbf{P}_{\text{ctrl}}$ and $\mathbf{U}_{\text{ctrl}}$, we train a logistic regression model, $s : \mathbb{R}^k \to [0, 1]$. We then apply s to the features extracted for $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$ and use the resulting logits $s(f_e(\mathbf{P}_{\text{test}}))$ and $s(f_e(\mathbf{U}_{\text{test}}))$ as membership confidence scores, with higher values referring to higher confidence that the given input sample is a member.

The motivation behind relying on the feature vector extracted by f_e is that a single feature-based score is prone to high variance [47], in turn making it more difficult to perform successful data detection. In contrast, combining multiple features should amplify the signal and improve the performance. Our experiments confirm this intuition, as we show in Sec. 8.3.3 and 8.5.1. Moreover, the scoring model addresses the challenge of determining which features provide the strongest membership signal for a given model. By aggregating information across multiple features, s highlights the most relevant signals for detecting membership.

8.4.3. Statistical Testing

Finally, we perform a two-sample single-tailed Welch's t-test. Our null hypothesis expresses that mean scores for $\mathbf{P}_{\text{test}}$ are not significantly higher than the ones for $\mathbf{U}_{\text{test}}$, *i.e.,* $H_0 : \overline{s(f_e(\mathbf{P}_{\text{test}}))} \leq \overline{s(f_e(\mathbf{U}_{\text{test}}))}$, where $\overline{s(f_e(\mathbf{P}_{\text{test}}))}$ and $\overline{s(f_e(\mathbf{U}_{\text{test}}))}$ are the mean of scores returned from s on the features of $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$, respectively. Rejecting H_0

at a significance level $\alpha = 0.01$, *i.e.,* obtaining p-value < 0.01, confirms that samples from $\mathbf{P}_{\text{test}}$ has been (illegitimately) used to train the queried DM.

Our choice of such a low α parameter is motivated by the **TPR@FPR=1%** metric established for MIAs [47], based on the intuition that the false positives are more harmful than false negatives in real-world applications, *e.g.,* court cases. To improve the soundness of our statistical tests, we perform CDI 1000 times on randomly sampled subsets of $\mathbf{P}$ and $\mathbf{U}$, and aggregate the obtained p-values [139, 234] (we provide more details in App. C.5).

8.5. Empirical Evaluation

Our CDI Setup. We use the diffusion models and datasets as specified in section 8.3.2. To instantiate our CDI, we draw samples from the train sets to represent $\mathbf{P}$ and samples from the test sets to represent $\mathbf{U}$. We set $|\mathbf{P}| = |\mathbf{U}|$ for all experiments. The maximum total size of $|\mathbf{P}| + |\mathbf{U}|$ we use for our experiments is 40,000 samples. Note, that this number is chosen only as a starting point and the number of data points for $\mathbf{P}$ and $\mathbf{U}$ that CDI requires to confidently reject H_0 is much lower and depends on the targeted model (see Fig. 8.5.1).

To maximize the use of both $\mathbf{P}$ and $\mathbf{U}$ while minimizing the number of samples required for our method, we implement a k-fold cross-validation with $k = 5$. The features extracted from the public samples ($\mathbf{P}$) and unpublished samples ($\mathbf{U}$) are divided into 5 folds. In each iteration, one fold is designated as the test set, containing $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$ features, while the remaining $k-1$ folds form the control set, comprising $\mathbf{P}_{\text{ctrl}}$ and $\mathbf{U}_{\text{ctrl}}$ features, which are used to train the scoring model s. This process is repeated across all splits, ensuring that each sample in $\mathbf{P}$ and $\mathbf{U}$ is used exactly once in the test set as part of $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$.

This procedure ensures that the statistical testing is performed with $|\mathbf{P}_{\text{test}}| = |\mathbf{P}|$ and $|\mathbf{U}_{\text{test}}| = |\mathbf{U}|$, allowing us to extract signals that identify training data from the entirety of the $\mathbf{P}$ and $\mathbf{U}$ sets.

8.5.1. Our CDI Confidently Identifies Collection of Data Samples as Training Data

We summarize the success of our CDI for diverse DMs and datasets in Fig. 8.5.1 (following the standard evaluation of DI as proposed in [157]). We report p-values as the confidence in the correct verification for different sizes of suspect data sets $\mathbf{P}$. Our results highlight that CDI already enables a confident ($p < 0.01$) dataset identification

with as little as 70 samples (for instance, for U-ViT256-T2I-Deep DM trained on the COCO dataset) provided by the data owner. For DMs trained on larger datasets like ImageNet, we observe the need to increase the size of **P** to confidently reject the null hypothesis. More details on the impact of the size of **P** on the confidence of CDI can be found in App. C.6. In general, in our results, we identify the following trends: (1) For a given DM architecture, trained on given dataset, the number of samples required for the confident identification of the training data decreases with increasing input resolution (see App. C.10). (2) The larger the overall training set of the model, the more samples are needed for a confident claim (see also App. C.10). (3) The higher the number of model training steps the stronger the signal for identifying training data as shown in fig. C.9.1 in App. C.11.

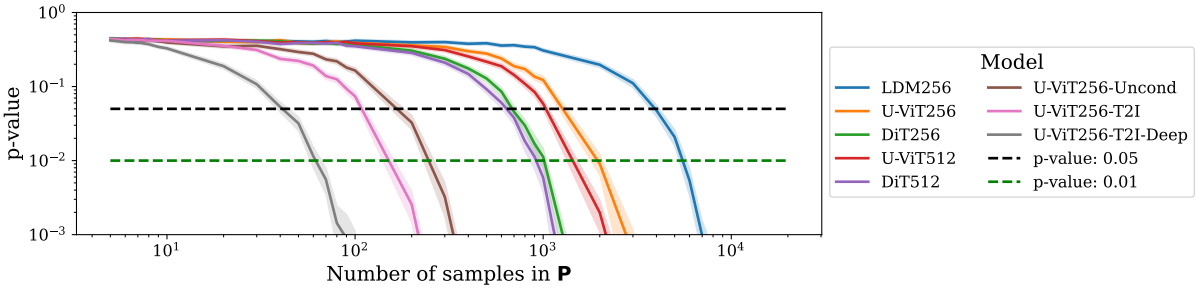


Figure 8.5.1. Results of CDI on various DMs. Solid lines indicate p-values aggregated over 1000 randomized trials for each size of **P**, shaded areas around the lines are 95% confidence intervals. CDI confidently rejects H_0 with as low as 70 suspect samples from the data owner. CDI's performance increases with larger model sizes and smaller training sets.

8.5.2. Analysis of the Success of CDI

We perform multiple ablations on CDI's building blocks to deepen understanding of its success. First, we show the importance of statistical testing as a core component of CDI. Then, we show that our features indeed boost the performance of CDI. Next, we demonstrate that our method remains effective even when not all samples from the suspect set were used in the DM's training set. Finally, we show that CDI does not return false positives. We include additional evaluation of CDI in App. C.18 and C.19, analysis of scoring model s in App. C.16, and time complexity in App. C.9.2. In Section C.8 we evaluate CDI on additional DMs and showcase how CDI can be extended with additional feature extraction methods.

Statistical Testing is Crucial for DI. In this ablation study, we assess the impact of removing the t-test from CDI and demonstrate that simply aggregating the MIA results for multiple samples is insufficient to reliably identify data collections used in DM training. To conduct this comparison, we aggregate membership scores across

a set of samples to determine if any members are present in the set. We define a *set membership score* as the highest membership score within a subset, hypothesizing that sets composed of members will yield higher scores than non-member sets. For evaluation, we sample 1000 subsets each from **U** (non-members) and **P** (members). We refer to this approach as *set-level MIA*, and execute this procedure for scores obtained from the scoring model component of CDI. We report **TPR@FPR=1%** in table 8.5.1.

A direct comparison between CDI's p-values and **TPR@FPR=1%** for set-level MIA is challenging due to the differing metrics. To align them, we compute the power of the t-test with $\alpha = 0.01$, which is equivalent to **TPR@FPR=1%** (see App. C.17). This approach allows us to directly compare set-level MIA to CDI without altering its methodology. The results in table 8.5.1 underscore the critical role of statistical testing in CDI. Set-level MIA without statistical testing underperforms, while CDI with its rigorous testing achieves near-perfect performance for most of the models.

Table 8.5.1. Impact of the statistical testing. The values in the table are **TPR@FPR=1%** and are in %. Results represent the *set-level* MIA (without the statistical testing) vs CDI with the statistical testing. The size of **P** is 1000. Statistical testing is essential for CDI.

	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Set-level MIA (no t-test)	10.20	22.90	10.50	6.50	0.00	33.40	23.20	32.50
CDI **(Ours)**	**24.92**	**62.74**	**93.00**	**74.43**	**93.76**	**100.00**	**100.00**	**100.00**

Table 8.5.2. Robustness of CDI against false positives. We depict averaged p-values returned by our method based on the data used within **P** with $|\mathbf{P}| = 10000$. We sample 1000 **P** and **U** sets. The results show that when testing non-members (both **P** and **U** contain only nonmembers), we obtain high p-values, significantly above the significance level ($\alpha = 0.01$), *i.e.,* we cannot reject the null hypothesis and do not identify the data from **P** as members. In contrast, when testing with member data points (**P** contains members and **U** contains nonmembers), our results are always significant, *i.e.,* $p < 0.01$, and we correctly identify the given set as members.

Data in $\mathbf{P}_{\text{test}}$	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Members	10^{-6}	10^{-21}	10^{-59}	10^{-31}	10^{-66}	10^{-266}	0.00	0.00
Non-members	0.40	0.39	0.39	0.39	0.40	0.40	0.39	0.38

Our Novel Features Significantly Decrease the Number of Samples Required for Verification. Our results in fig. 8.5.2 indicate that introducing our new features improves the efficiency of CDI substantially in comparison to the joint features extracted by the MIAs from section 8.3.1, in the following referred to as *existing MIAs*. Applying our new features in CDI leads to a remarkable reduction of the number of samples needed to reject the null hypothesis with $p < 0.01$, especially in

the initially most challenging cases of models trained on higher-resolution large datasets. For instance, for U-ViT512 the number of samples required from a data owner for confident verification decreases from 20000 to 2000. This makes CDI more practical and applicable to more users who have smaller datasets that they would like to verify.

Regarding our novel features, the most influential one is GM. We note that utilizing only existing MIAs + GM, we are able to obtain performance very close to CDI. The extension of the feature set only with NO yields the smallest improvement over the alternatives in most cases, however, discarding it entirely (see: existing MIAs + GM + ML vs All Features) results in worse performance in the case of U-ViT512, highlighting the need for a diverse source of the signal to obtain confident predictions. Our ML feature succeeds in capturing a better membership signal than its existing MIAs-based counterpart (Denoising Loss), striking a middle ground between GM and NO.

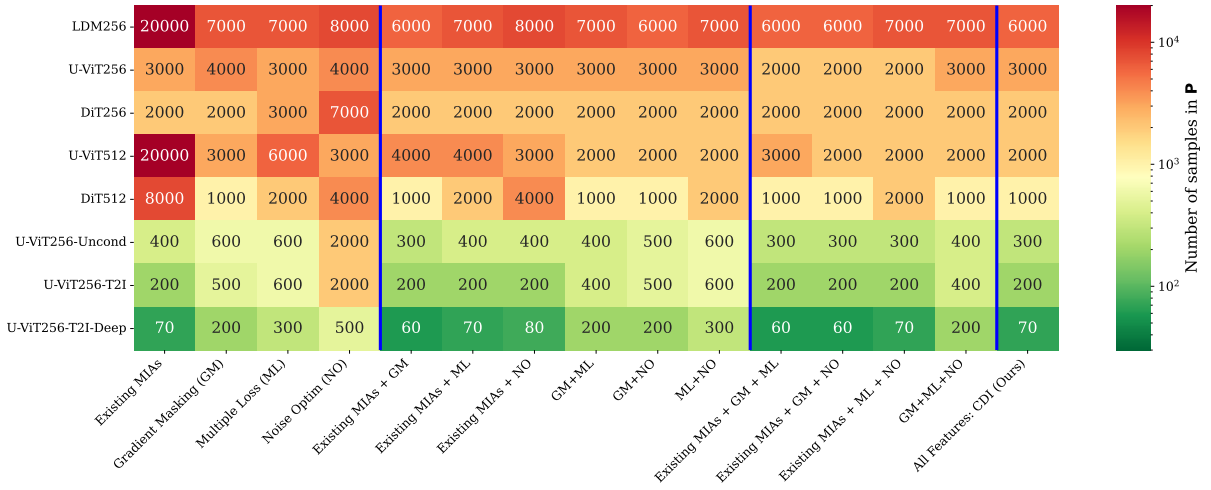


Figure 8.5.2. Impact of feature selection. The values in the cells indicate the minimum size of **P** needed to reject H_0. Blue vertical lines separate results by the complexity of the feature set used to fit s: (left) our novel individual features from section 8.4.1 and a joint existing MIAs feature, (second from left) all possible combinations of two, and of three features (second from right), and (right) all available features.

CDI is Effective Even When Not All Data Samples Were Used as Training Data. We investigate CDI 's behavior in cases where only a part of the samples in the suspect set **P** was used to train the DM, *i.e.,* **P** contains a certain ratio of non-members, while the remaining samples in **P** are members. Practically, this corresponds to the situation where a data owner has a set of publicly exposed data points and suspects that all of them might have been used to train the DM, whereas, in reality, some were not used. This can happen, *e.g.,* due to internal data cleaning on the side of the party who trained the DM. In particular, the data owner does not

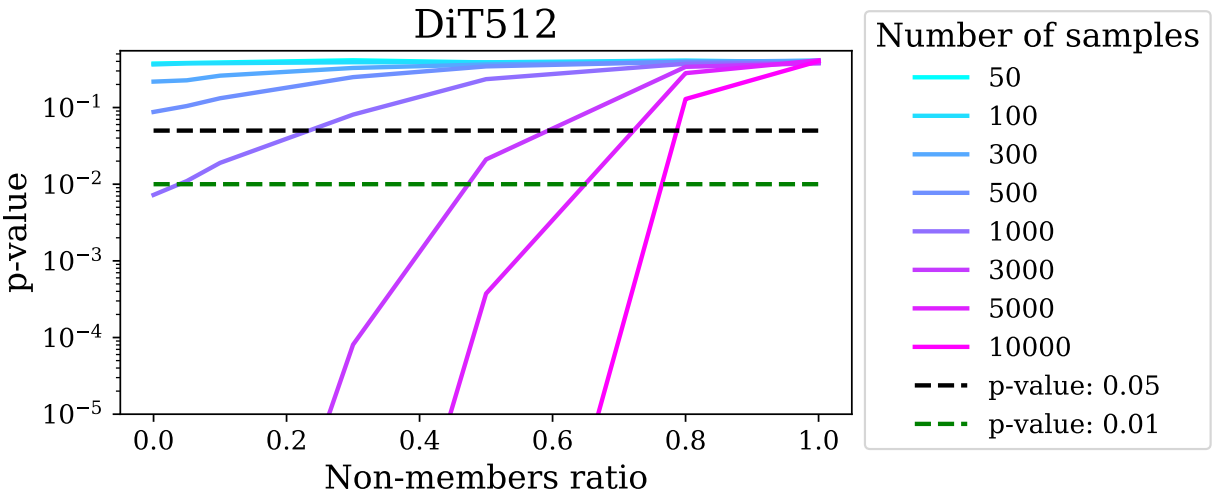


Figure 8.5.3. Impact of non-members ratio in P on `CDI`. The lines represent p-values for a given non-member ratio while varying sizes of **P**.

know which of their samples and how many of them have not been included into training. In Fig. 8.5.3 (extended by Fig. C.9.2 in App. C.12), we present the success of `CDI` under different ratios of non-member samples in **P**. Note that our evaluation is the result of 1000 randomized experiments for each non-members ratio, model, and **P** size. We observe that `CDI` remains effective when the non-member ratio is over 0.5 and 0.8 for some models, *i.e.,* it still correctly identifies **P** as training data of the model. Overall `CDI`'s robustness is higher when the data owner provides larger suspect datasets **P**. This is reflected in the p-value at the same non-member ratio decreasing as the number of samples in **P** increases.

CDI is Effective Even Under Gray-box Model Access. We analyze the effectiveness of performing `CDI` in the gray-box model access scenario, as defined in the threat model (section 8.4). Therefore, we include only the original MIA features and Multiple Loss in `CDI`. Even in this case, `CDI` remains effective under gray-box model access and can reject the null hypothesis. In this more difficult scenario, across the eight tested DMs, `CDI` requires on average **one-third** more samples in **P** compared to the white-box access (where all features can be used). We refer to App. C.15 for a detailed comparison.

CDI is Robust Against False Positives. While `CDI` can correctly identify suspect datasets even when not all samples were used as training data, it raises the concern of false positives, *i.e.,* reporting data as used for training a DM when it was not. In particular, `CDI` should only reject the null hypothesis if **P** contains (some) members and yield inconclusive results otherwise. To show that `CDI` is robust against false positives, we instantiate **P** only with non-member samples. Our results in Table 8.5.2

highlight CDI 's reliability in distinguishing between non-member and member sets without false positives.

8.6. Conclusions

We introduce CDI as a method for data owners to verify if their data has been (illegitimately) used to train a given DM. While existing MIAs alone are insufficient to confidently determine whether a specific data point was used during training, CDI overcomes this limitation. By selectively combining features extracted from MIAs with novel handcrafted features and applying them across a larger data set, we achieve a reliable discriminator for identifying datasets used in DM training. Our rigorous feature engineering amplifies the signal in CDI, enabling individual artists even with smaller collections of art to benefit from our method.

9. Privacy Attacks on Image AutoRegressive Models

Title	Privacy Attacks on Image AutoRegressive Models
Authors	Antoni Kowalczuk*, Jan Dubiński*, Franziska Boenisch, Adam Dziedzic
Conference	International Conference on Machine Learning (ICML)
Year	2025

Preface

In the previous chapter, we introduced Copyrighted Data Identification, a method for reliably detecting whether entire datasets were used to train large diffusion models. That work demonstrated how weak membership signals can be aggregated to form strong, actionable evidence of data usage. Yet diffusion is not the only paradigm shaping the current landscape of generative modeling. Recently, image autoregressive models have emerged as a powerful alternative, achieving state-of-the-art performance in image synthesis and raising new questions about their reliability and privacy.

Autoregressive models differ fundamentally from diffusion-based architectures. Instead of gradually denoising a latent representation, they generate samples sequentially, token by token. This design gives them strong expressive power, but it also introduces distinct privacy challenges. In particular, the step-by-step generation process makes memorization of training data more pronounced, which may increase the risk of leakage compared to diffusion models.

In this work, we investigate these risks by adapting privacy attacks to the autoregressive setting. First, we revisit membership inference and dataset identification, tailoring them to account for the sequential structure of autoregressive generation. Then, we explore the extent to which these models memorize training data, developing methods that extract content directly from the autoregressive outputs. Across all evaluations, we observe that image autoregressive models are especially prone to privacy leakage, often exceeding the vulnerabilities of diffusion models.

Building on the insights from Copyrighted Data Identification, we further demonstrate that dataset-level signals remain effective in this new paradigm. Aggregating membership evidence across multiple samples enables us to detect the unauthorized use of copyrighted data, providing a foundation for defending the rights of data owners against misuse.

With this chapter, the dissertation concludes its exploration of trustworthy generative models. Having examined both reliability in scientific simulations and safeguards for models and data, we now turn to the final discussion, where we reflect on the broader implications of these contributions and outline directions for future work.

Abstract

Image AutoRegressive generation has emerged as a new powerful paradigm with image autoregressive models (IARs) matching state-of-the-art diffusion models (DMs) in image quality (FID: 1.48 vs. 1.58) while allowing for a higher generation speed. However, the privacy risks associated with IARs remain unexplored, raising concerns regarding their responsible deployment. To address this gap, we conduct a comprehensive privacy analysis of IARs, comparing their privacy risks to the ones of DMs as reference points. Concretely, we develop a novel membership inference attack (MIA) that achieves a remarkably high success rate in detecting training images (with a True Positive Rate at False Positive Rate = 1% of 86.38% vs. 6.38% for DMs with comparable attacks). We leverage our novel MIA to provide dataset inference (DI) for IARs, and show that it requires as few as 6 samples to detect dataset membership (compared to 200 for DI in DMs), confirming a higher information leakage in IARs. Finally, we are able to extract hundreds of training data points from an IAR (e.g., 698 from VAR-$d30$). Our results suggest a fundamental privacy-utility trade-off: while IARs excel in image generation quality and speed, they are *empirically* significantly more vulnerable to privacy attacks compared to DMs that achieve similar performance. We release the code at `https://github.com/sprintml/privacy_attacks_against_iars` for reproducibility.

9.1. Introduction

The field of visual generative modeling has seen rapid advances in recent years, primarily due to the rise of Diffusion Models (DMs) [214] that achieve impressive performance in generating highly detailed and realistic images. For this ability, they currently act as the backbones of commercial image generators [24, 193, 221]. Yet, recently, their performance was closely matched or further surpassed through novel image autoregressive models (IARs). Over the last months, IARs have been achieving new state-of-the-art performance for class-conditional [145, 223, 256] and text-conditional [94, 109, 220] generation. The crucial improvement of their training cost and generation quality results from the *scaling laws* that previously were observed for large language models (LLMs) [132] with which they share both a training paradigm and architectural foundation. As a result, with more compute budget, and larger datasets, IARs can achieve better performance than their DM-based counterparts.

At the same time, the privacy risks of IARs remain largely unexplored, posing

challenges for their responsible deployment. While privacy risks, such as the leakage of training data points at inference time, have been demonstrated for DMs and LLMs [19, 48, 77, 78, 110, 112, 122, 244], no such evaluations currently exist for IARs. As a result, the extent to which IARs may similarly expose sensitive information remains an open question, underscoring the necessity for rigorous privacy investigations in this context.

To address this gap and investigate the privacy risks associated with IARs, we conduct a comprehensive analysis using multiple perspectives on privacy leakage. First, we develop a new membership inference attack (MIA) [211], which aims to determine whether a specific data point was included in an IAR's training set—a widely used approach for assessing privacy risks. We find that existing MIAs developed for DMs [48, 79, 137, 258] or LLMs [161, 209], are ineffective for IARs, as they rely on signals specific to their target model. We combine elements of MIAs from DMs and LLMs into our new MIA based on the shared properties between the models. For example, we leverage the fact that IARs, similarly to LLMs, perform per-token prediction to obtain signal from every predicted token. However, while LLMs' training is fully self-supervised (e.g., by predicting the next word), the training of IARs can be conditional (based on a class or prompt) as in DMs. We exploit this property, previously leveraged for DMs [258], and compute the difference in outputs between conditional and unconditional inputs as an input to MIAs. This approach allows us to achieve a remarkably strong performance of **86.38**% TPR@FPR=1%.

We employ our novel MIA to provide an efficient dataset inference (DI) [157] method for IARs. DI generalizes MIAs by assessing membership signals over entire datasets, providing a more robust measure of privacy leakage. Additionally, we optimize DI for IARs by eliminating the stage of MIA selection for a given dataset, which was necessary for prior DIs on LLMs [158, 264] and DMs [83]. Since our MIAs for IARs consistently produce higher scores for members than for non-members, all MIAs can be utilized without any selection. This optimizations reduced the number of samples required for DI in IARs to as few as 6 samples, which is significantly fewer than at least 200 samples required for DI in DMs. Finally, we examine the privacy leakage from IARs through the lens of memorization [97, 116, 122, 237–239, 244]. Specifically, we assess the IARs' ability to reproduce verbatim outputs from their training data during inference. We experimentally demonstrate that the evaluated IARs have a substantial tendency to verbatim memorization by extracting 698 training samples from VAR-$d30$, 36 from RAR-XXL, and 5 from MAR-H. These results highlight the varying degrees of memorization across models and reinforce the importance of mitigating privacy risks in IARs. Together, these approaches form a comprehensive framework for empirically evaluating the privacy risks of IARs.

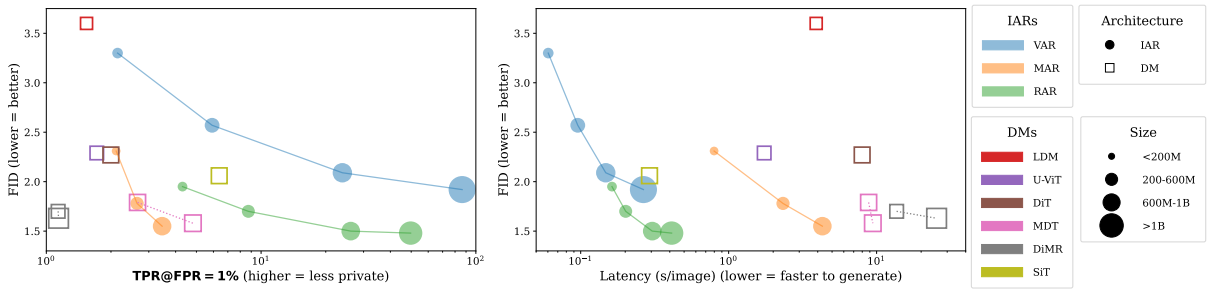


Figure 9.1.1. Privacy-utility and generation speed-performance trade-off for IARs compared to DMs. 1) IARs achieve better and faster image generation, but reveal more information to potential training data identification attacks. 2) In particular, large IAR models are most vulnerable. 3) In case of large IARs, even the identification of individual training samples (MIAs) has a high success rate. 4) MAR models are more private than other IARs. We attribute it to the inclusion of a diffusion module in this architecture.

Our empirical analysis of state-of-the-art IARs and DMs across various scales suggests that IARs that match their DM-counterparts in image generative capabilities are notably more susceptible to privacy leakage. We also explore the trade-offs between privacy risks and other model properties. Specifically, we find that, while IARs are more cost-efficient, faster, and more accurate in generation than DMs, they empirically exhibit significantly greater privacy leakage (see Figure 9.1.1) measured against SOTA privacy attacks tailored against the respective model types. These findings highlight a critical trade-off between performance, efficiency, and privacy in IARs.

In summary, we make the following contributions:

- Our new MIA for IARs achieves extremely strong performance of even **86.38%** TPR@FPR=1%, improving over naive application of MIAs by up to **69%**
- We provide a potent DI method for IARs, which requires as few as **6** samples to assess dataset membership signal.
- We propose an efficient method of training data extraction from IARs, and successfully extract up to **698** images.
- IARs can outperform DMs in generation efficiency and quality but suffer **order-of-magnitude** higher privacy leakage in MIAs, DI, and data extraction compared to DMs that demonstrate similar FID.

9.2. Background and Related Work

Notation. We first introduce the notation used throughout the remainder of this paper in Table 9.2.1.

Table 9.2.1. Notation used in our work.

Symbol	Description
C, H, W, N	Channels, height, width, sequence length
$x \in \mathbb{R}^{C \times H \times W}$	Original image
$\hat{x} \in \mathbb{R}^{C \times H \times W}$	Generated image
$t \in \mathbb{N}^N$	Tokenized image
$\hat{t} \in \mathbb{N}^N$	Generated token sequence

Image AutoRegressive modeling. Originally, Chen et al. [52] defined image autoregressive modeling as:

$$p(x) = \prod_{n=1}^{N} p(t_n \mid t_1, t_2, \ldots, t_{n-1}), \tag{9.1}$$

where N is the number of pixels in the image, t_i is the value of i^{th} pixel of image $x \sim \mathcal{D}_{\text{train}}$ (training data), where pixels follow raster-scan order, row-by-row, left-to-right. During training, the goal is to minimize negative log-likelihood:

$$L_{AR} = \mathbb{E}_{x \sim \mathcal{D}_{\text{train}}} \left[-\log(p(x)) \right]. \tag{9.2}$$

However, learning pixel-level dependencies directly is computationally expensive. To address the issue, VQ-GAN [91] transforms the task from next-pixel to next-token prediction. First, the VQ-GAN's encoder maps an image into (lower resolution) latent feature vector, which is then quantized into a sequence of tokens, by a learnable codebook. In effect, the sequence length is short, which enables higher-resolution and high-quality generation. Then, tokens are generated and projected back to the image space by VQ-GAN's decoder. All the subsequent IARs we introduce, utilize tokens from VQ-GAN. This token-based formulation aligns image generation more closely with natural language processing. Additionally, similarly to autoregressive language models such as GPT-2 [177], which generate text by sequentially predicting tokens, modern IARs also employ transformer-based [230] architectures to model dependencies between image tokens. We focus on the recent state-of-the-art IARs.

VAR [223] is a novel approach to image generation, which shifts the focus of traditional autoregressive learning from next-token to next-scale prediction. Unlike classical IARs, which generate 1D token sequences from images by raster-scan orders, VAR introduces a coarse-to-fine multi-scale approach, encoding images into hierarchical 2D token maps and predicting tokens progressively from lower to higher resolutions. This preserves spacial locality and significantly improves scalability and inference speed.

RAR [256] introduces bidirectional context modeling into IAR. Building on find-

ings from language modeling, specifically BERT [72], RAR highlights the limitations of unidirectional approach, and enhances training by randomly permuting token sequences and utilizing bidirectional attention. RAR optimizes eq. (9.2) over all possible permutations, enabling the model to capture bidirectional dependencies, resulting in higher quality generations.

MAR [145] uses a small DM to model $p(x)$ from eq. (9.1), and samples tokens from it during inference. MAR is trained with the following loss objective:

$$L_{DM} = \mathbb{E}_{\epsilon,s}\left[||\epsilon - \epsilon_\theta\left(t_n^s \mid s, z\right)||^2\right], \tag{9.3}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, ϵ_θ is the DM, $t_n^s = \sqrt{\bar{\alpha}_s} t_n + \sqrt{1-\bar{\alpha}_t}\epsilon$ and $\bar{\alpha}_s$ is DDIM's [215] noise schedule, s is the timestep for diffusion process, and z is conditioning input, obtained from the autoregressive backbone, from the previous tokens. This loss design allows MAR to operate with continuous-valued tokens, contrary to VAR and RAR, which use discrete tokens. MAR also integrates masked prediction strategies from MAE [113], into the IAR paradigm. Specifically, MAR predicts masked tokens, based on unmasked ones, formulated as $p(x \cdot \neg M \mid x \cdot M)$, where $M \in [0,1]^N$ is random binary mask. Like to RAR, MAR utilizes bidirectional attention during training. Its autoregressive backbone differs from other IARs, as MAR employs a ViT [75] backbone.

Sampling for IARs is based on $p(x)$, which models the distribution of the next token conditioned on the previous ones in the sequence. For VAR and RAR, operating on discrete tokens, the next token can be predicted via greedy or top-k sampling. In contrast, MAR samples tokens by the DM module, which performs 100 DDIM [215] denoising steps. During a single sampling step, VAR outputs a 2D token map, RAR predicts a single token, and MAR generates a batch of tokens.

9.3. Privacy Evaluation Frameworks

We assess IARs' privacy risks from the three perspectives of membership inference, dataset inference, and memorization.

9.3.1. Membership Inference

Membership Inference Attacks (MIAs) [211] aim to identify whether a specific data point was part of the training dataset for a given machine learning model. Many MIAs have been proposed for DMs [48, 79, 137, 258], but these methods are tailored to DM-specific properties and do not transfer easily to IARs. For instance, some

directly exploit the denoising loss [48], while others [137], leverage discrepancies in noise prediction between clean and noised samples. CLiD [258] sources membership signal from the difference between conditional and un-conditional prediction of the DM. Since IARs are also trained with conditioning input, we leverage CLiD to design our MIAs in section 9.5.1.

MIAs are also popular against LLMs [161, 209] where they often work with per-token logit outputs of the model. For example, Shi et al. [209] introduce the MIN-K% PROB metric, which computes the mean of lowest k%-log-likelihoods in the sequence, where k is a hyper-parameter. Zlib [19] leverages the compression ratio of predicted tokens using the *zlib library* [102] to adjust the metric to the level of complexity of the input sequence. Hinge [42] metric computes the mean distance between tokens' log-likelihood and the maximum of the remaining log-likelihoods. SURP [259] computes the mean of log-likelihood of the tokens with the lowest k%-log-likelihoods in the sequence, where k is some pre-defined threshold. MIN-K%++ [260] is based on MIN-K% PROB, but the per-token log-likelihoods are normalized by the mean and standard deviation of the log-likelihoods of preceding tokens. CAMIA [51] computes the mean of log-likelihoods that are smaller than the mean log-likelihood, and the mean of log-likelihoods that are smaller than the mean of the log-likelihoods of preceding tokens, as well as the slope of log-likelihoods. More detailed description of MIAs can be found in section D.5.2. While LLM MIAs seem to be a natural choice for membership inference on IARs, it is completely unclear whether approaches from the language domain transfer to IARs. In our work we show that the success of this transferability is limited (see section 9.5.1), hence, we design novel MIAs, by exploiting unique properties of IARs. Our methods achieve significant improvements over initial MIAs with up to **69**% higher TPR@FPR=1% compared to the baselines.

9.3.2. Dataset Inference

Dataset Inference (DI) [157] aims to determine whether a specific dataset was included in a model's training set. Therefore, instead of focusing on individual data points like MIAs, DI aggregates the membership signal across a larger set of training points. With this strong signal, it can uniquely identify whether a model was trained on a given (private) dataset, leveraging strong statistical evidence. Similarly to MIAs, DI can serve as a proxy for estimating privacy leakage from a given machine learning model: DI provides insight into how easily one can determine which datasets were used to train a model, for instance, by analyzing the effect size from statistical tests. A higher success rate in DI indicates greater potential privacy leakage.

Previous DI Methods. For supervised models, DI involves the following three steps: (1) obtaining specific features from data samples, based on the observation

that training data points are further from decision boundaries than test samples, then (2) aggregating the extracted information through a binary classifier, and (3) applying statistical tests to identify the model's train set. This approach was later extended to self-supervised learning models [85, 86], where training data representations differ from test data, and then to LLMs [158, 264] and DMs [83] to identify the training datasets in large generative models. Since DI relies on model-specific properties, it is unclear how it can be applied to IARs. We propose how to make DI applicable and effective for IARs.

Setup for DI. DI relies on two data sets: (suspected) member and (confirmed) non-member sets. First, the method extracts features for each sample using MIAs. Next, it aggregates the features for each sample, and obtains the final score, which is designed so that it should be higher for members. Then, it formulates the following hypothesis test: $H_0 : mean(\text{scores of suspected member samples}) \leq mean(\text{scores of non-members})$, and uses the Welch's t-test for evaluation. If we reject H_0 at a confidence level $\alpha = 0.01$, we claim that we confidently identified suspected members as actual members of the training set.

Since the strength of the t-test depends on the size of both sample sets, the goal is to reject H_0 *with as few samples as possible*. Intuitively, as the difference in a model's behavior between member and non-member samples increases, rejecting H_0 becomes easier. A larger difference also indicates greater information leakage, allowing us to use DI to compare models in terms of privacy risks. For instance, if model A allows rejection of H_0 with 100 samples, while model B requires 1000 samples, model A exhibits higher leakage than model B. Throughout this paper, we refer to the minimum number of samples required to reject the null hypothesis as P.

Assumptions about Data. For the hypothesis test to be sound, the suspected member set and non-member set must be independently and identically distributed. Otherwise, the result of the t-test will be influenced by the distribution mismatch between these two sets, yielding a false positive prediction.

9.3.3. Memorization

Memorization in generative models refers to the models' ability to reproduce training data exactly or nearly indistinguishably at inference time. While MIAs and DI assess if given samples were used to train the model, memorization enables extracting training data directly from the model [19, 48]—-highlights an *extreme* privacy risk.

In the vision domain, a data point x is memorized, if the distance $l(x, \hat{x})$ from the original x and the generated $\hat{x}$ image is smaller than a pre-defined threshold τ [48]. We use the same definition when evaluating our extraction attack in Section 9.5.3.

Intuitively, in LLMs, memorization can be understood as the model's ability to reconstruct a training sequence t when given a prefix c [19]. Specifically, $t = \text{argmax}_{t' \in \mathbb{N}^N} p_\theta(t'|c)$, where p_θ is the probability distribution of the sequence t', parameterized by the LLM's weights θ, akin to eq. (9.1). This formulation states we can extract the training sequence t by constructing a prefix c that makes the model output t, with greedy sampling.

Similarly to LLMs, IARs complete an image given an initial portion of it (a prefix), which we leverage for designing our data extraction attack. In contrast, extraction from DMs can rely only on the conditioning input (class label or text prompt), which is both costly and highly inefficient, *e.g.,* work by Carlini et al. [48] requires to generate **175M** images in order to find just 50 memorized images, and no memorization has been shown for other large DMs. In contrast, we extract up to **698** training samples from IARs by conditioning them on a part of the tokenized image, requiring only **5000** generations.

9.4. Experimental Setup

We evaluate state-of-the-art IARs: VAR-d{16, 20, 24, 30} (d = model depth), RAR-{B, L, XL, XXL}, MAR-{B, L, H}, trained for class-conditioned generation. The IARs' sizes cover a broad spectrum between 208M for MAR-B, and 2.1B parameters for VAR-d30. We use IARs shared by the authors of their respective papers in their repositories, with details in section D.6. As these models were trained on ImageNet-1k [71] dataset, we use it to perform our privacy attacks. For MIA and DI, we take 10000 samples from the training set as members and also 10000 samples from the validation set as non-members. To perform data extraction attack, we use all images from the training data. Additionally, we leverage the known validation set to check for false positives.

9.5. Our Methods for Assessing Privacy in IARs

In the following we investigate privacy risks of IARs. We start from baseline, LLM-based approaches, and show how to tailor them to IARs to increase privacy leakage. As we find that IARs leak more than DMs we provide insights to explain why does it happen.

Table 9.5.1. Performance of our MIAs vs baselines. We report the standard TPR@FPR=1% for best MIAs per model. *Baselines* refers to a unmodified naive application of LLM-specific MIAs to IARs.

Model	VAR-$d16$	VAR-$d20$	VAR-$d24$	VAR-$d30$	MAR-B	MAR-L	MAR-H	RAR-B	RAR-L	RAR-XL	RAR-XXL
Baselines	1.62	2.21	3.72	16.68	1.69	1.89	2.18	2.36	3.25	6.27	14.62
Our Methods	**2.16**	**5.95**	**24.03**	**86.38**	**2.09**	**2.61**	**3.40**	**4.30**	**8.66**	**26.14**	**49.80**
Improvement	**+0.54**	**+3.73**	**+20.30**	**+69.69**	**+0.40**	**+0.73**	**+1.22**	**+1.94**	**+5.41**	**+19.87**	**+35.17**

9.5.1. Tailoring Membership Inference for IARs

Baselines. We comprehensively analyze how existing MIAs designed for LLMs transfer to IARs. Our results in table 9.5.1 (detailed in section D.9) indicate that off-the-shelf MIAs for LLMs perform poorly when directly applied to IARs. We report the TPR@FPR=1% metric to measure the true positive rate at a fixed low false positive rate, which is a standard metric to evaluate MIAs [47]. For smaller models, such as VAR-$d16$, MAR-B, and RAR-B, all MIAs exhibit performance close to random guessing ($\sim 1\%$). As model size and the number of parameters increase, the membership signal strengthens, improving MIAs' performance in identifying member samples. Even in the best case (CAMIA with TPR@FPR=1% of 16.69% on the large VAR-$d30$), the results indicate that the problem of reliably identifying member samples remains far from being solved. These findings align with results reported for other types of generative models, as demonstrated by zha [35], Duan et al. [80], Maini et al. [158] in their evaluation of MIAs on LLMs and by [33, 258] for DMs, where the utility of MIAs for models trained on large datasets was shown to be severely limited.

Our MIAs for VARs and RARs. To provide powerful MIAs for IARs, we leverage the models' key properties. Specifically, we exploit the fact that IARs utilize classifier-free guidance [118] during training, *i.e.,* in the forward pass, images are processed both with and without conditioning information, such as class label. This distinguishes IARs from LLMs, which are trained without explicit supervision (no conditioning). Consequently, MIAs designed for LLMs fail to take advantage of this additional conditioning information present in IARs. We build on CLiD [258], and compute $p(x|c) - p(x|c_{null})$, where c—class label, c_{null}—null class, and use this difference as an input to MIAs, instead of per-token logits. We differ from CLiD in the following way: (1) Our method works directly on $p(x)$, whereas CLiD uses model loss to perform the attack. (2) Our attack is parameter-free —CLiD requires hyperparameter search and a set of samples to fit a Robust-Scaler to stabilize the MIA signal. We provide a more generalized approach, moreover our results in table 9.5.1 demonstrate even up to a **69.69%** increase in the TPR@FPR=1% for the VAR-$d30$ model.

Our MIAs for MARs. Many MIAs for LLMs (Hinge, MIN-K%++, SURP) require logits to compute their membership scores. However, we cannot apply these MIAs to MAR since MAR predicts continuous tokens instead of logits. We instead use per-token loss values obtained from eq. (9.3) to adapt other LLM MIAs (Loss, Zlib, MIN-K% PROB, CAMIA). As the tokens for MAR are generated using a small diffusion module, we can apply insights from MIAs designed for DMs and target the diffusion module directly in our attack. We detail our MIA improvements for MAR, which counter randomness from the diffusion process and binary masks.

Improvement 1: Adjusted Binary Masks. MAR extends the IAR framework by incorporating masked prediction strategies, where masked tokens are predicted based on visible ones. We hypothesize that adjusting the masking ratio during inference can amplify membership signals. We increase this parameter from 0.86 (training average) to 0.95, which improves MIA and suggests that an optimal masking rate exposes more membership information.

Improvement 2: Fixed Timestep. Carlini et al. [48] reported that MIAs on DMs perform best when executed for a specific denoising step t. Since tokens in MAR are generated using a small diffusion module, we can take advantage of this by executing MIAs at a fixed timestep t rather than a randomly chosen one. Interestingly, we find that $t = 500$ is the most discriminative, differing from the findings for full-scale DMs, for which $t = 100$ gives the strongest signal Carlini et al. [48].

Improvement 3: Reduced Diffusion Noise Variance. The MAR loss in eq. (9.3) exhibits high variance due to its dependence on randomly sampled noise ϵ. To mitigate this, we increase the noise sampling count from the default 4 used during training to 64, computing the mean loss to obtain a more stable signal.

More detailed description of these improvements can be found in section D.8. Our results in table 9.5.2 highlight the importance of our changes to evaluate MAR's privacy leakage correctly. Thanks to our improved MIAs we do not under-report the privacy leakage they exhibit.

Table 9.5.2. Ablation of improvements to MAR MIAs. Each modification further strengthens the membership signal. We report TPR@FPR=1% values and gains.

Method	MAR-B	MAR-L	MAR-H
Baseline	1.69	1.89	2.18
+ Adjusted Binary Mask	1.88 (+0.19)	2.25 (+0.36)	2.88 (+0.70)
+ Fixed Timestep	1.88 (+0.00)	2.41 (+0.17)	3.30 (+0.42)
+ Reduced Noise Variance	**2.09 (+0.21)**	**2.61 (+0.20)**	**3.40 (+0.10)**

Overall Performance and Comparison to DMs We present our results in Figure 9.1.1, evaluate overall privacy leakage and compare IARs to DMs based on the TPR@FPR=1% of MIAs. For DMs we use the strongest attack available at the time

of writing this paper—CLiD [258]. In general, smaller and less performant models exhibit lower privacy leakage, which increases with model size. Notably, VAR-$d30$ and RAR-XXL achieve TPR@FPR=1% values of 86.38% and 49.80%, respectively, indicating a substantially higher privacy risk in IARs compared to DMs. In contrast, the highest TPR@FPR=1% observed for DMs is only 6.38% for SiT-XL/2 (see also table D.9.5).

Possible Reasons Behind Higher Leakage of IARs With IARs emerging as a less private alternative to DMs, we investigate the causes behind that phenomenon. First, we ask if IARS inherently leak more because of their design. We identify three key characteristics of IARs that cause greater leakage: (1) Access to $p(x)$—IARs expose it at the output, contrary to DMs. (2) AutoRegressive training exposes IARs to more data per update. (3) Each token predicted by an IAR leak unique information about the model, amplifying leakage. We provide more details in section D.2.1. Next, we scrutinize architecture-agnostic causes of leakage: training duration, and model size. Our results in table D.2.1 in section D.2.2 show that indeed, these two factors correlate with the leakage metrics. Interestingly, for IARs the vulnerability differs with model size, while for DMs—with training duration. We also test a binary factor "Is IAR" (1 if the model is IAR, 0 otherwise), which also correlates with metrics, further confirming our intuitions about the inherent causes of leakage in IARs. We note taht MIAs are significantly less effective at identifying member samples in MARs. We attribute this to MAR's use of a diffusion loss function (eq. (9.3)) for modeling per-token probability, which replaces categorical cross-entropy loss and eliminates the need for discrete-valued tokenizers.

Vulnerability of IARs Through a Lens of a Unified MIA Finally, we look into the DM- and IAR-specific MIAs used in our study. We acknowledge that because DMs and IARs are two different classes of models, the MIAs that target each of the architectures also differ. Effectively, that variability might be the root cause of the observed discrepancy in MIA success. To evaluate that idea, we design a *Unified MIA*—an identical MIA for DMs and IARs—based on model- and architecture-agnostic Loss Attack [254]. We discard any IAR-specific improvements introduced in this section, and any DM-specific improvements from prior work [48]. Effectively, with Unified MIA we mitigate the potential influence of discrepancy in the MIA design on the final privacy assessment. Our results in table D.4.1 show that Unified MIA performs better than random guessing against IARs, while DMs show no leakage from that attack.

Table 9.5.3. DI for IARs. We report the reduction in the number of samples required to carry out DI. Our improvements allow to successfully run DI on IARs even with fewer than 10 samples. *Baseline* refers to LLM DI [158].

Model	VAR-*d*16	VAR-*d*20	VAR-*d*24	VAR-*d*30	MAR-B	MAR-L	MAR-H	RAR-B	RAR-L	RAR-XL	RAR-XXL
Baseline	2000	300	60	20	5000	2000	900	500	200	40	30
+Optimized Procedure	600	200	40	8	4000	2000	800	300	80	30	10
Improvement	-1400	-100	-20	-12	-1000	0	-100	-200	-120	-10	-20
+Our MIAs for IARs	**200**	**40**	**20**	**6**	**2000**	**600**	**300**	**80**	**30**	**20**	**8**
Improvement	-400	-160	-20	-2	-2000	-1400	-500	-220	-50	-10	-2

9.5.2. Dataset Inference

While our results in table 9.5.1 demonstrate impressive MIA performance for large models (such as VAR-*d*30 with 2.1B parameters), privacy risk assessment for smaller models (such as VAR-*d*16 with 310M parameters) needs improvement. To address this, we draw on insights from previous work on DI [83, 158], which has proven effective when MIAs fail to achieve satisfactory performance. The advantage of DI over MIAs lies in its ability to aggregate signals across multiple data points while utilizing a statistical framework to amplify the overall membership signal, yielding more reliable privacy leakage assessment. We find that while the framework of DI is applicable to IARs, its crucial parts must be improved to boost DI's effectiveness on IARs. In the following sections, we detail these improvements.

Improvement 1: Optimized DI Procedure. Existing DI techniques for LLMs [158] and DMs [83] follow a four-stage process, with the third stage involving the training of a linear classifier. This classifier is used to weight, scale, and aggregate signals from individual MIAs, where each MIA score serves as a separate feature. This step is crucial for selecting the most effective MIAs for a given dataset while suppressing ineffective ones that could introduce false results. However, we observe that MIA features for IARs are well-behaved, meaning that, on average, they are consistently higher for members than for non-members. Thus, instead of training a linear classifier on MIA features, which requires additional auditing data, we adopt a more efficient approach: we first normalize each feature using MinMaxScaler to the [0,1] interval, and then we sum them to obtain the final per-sample score, used by the t-test. This eliminates the need to allocate scarce auditing data for training a linear classifier.

Our results for the optimized DI procedure are presented in table 9.5.3. We observe a significant reduction in the number of samples required to perform DI for smaller models, with reductions of up to 70% for VAR-*d*16.

Improvement 2: Our MIAs for IARs. Our results in table 9.5.3 indicate that as model size increases, the membership signal is amplified, enabling DI to achieve better performance with fewer samples. However, the main problem is the mixed reliability of DI when utilizing baseline MIAs as feature extractors. This issue

is especially evident for smaller models, such as VAR-$d16$ and MAR-B, where DI requires thousands of samples to successfully reject the null hypothesis when the suspect set is part of the training data. Building on the performance gains of our tailored MIAs (table 9.5.1) we apply them to the DI framework as the more powerful feature extractors to further strengthen DI for IARs. Our improvements through stronger MIAs further enhance DI, fully exposing privacy leakage in IAR models. As a result, the number of required samples to execute DI drops to a few hundred, for example, down to only 200 for VAR-$d16$. Overall, as shown in table 9.5.3, replacing the linear classification model with summation and transitioning to our MIAs for IARs as feature extractors significantly reduces the number of samples required to reject H_0.

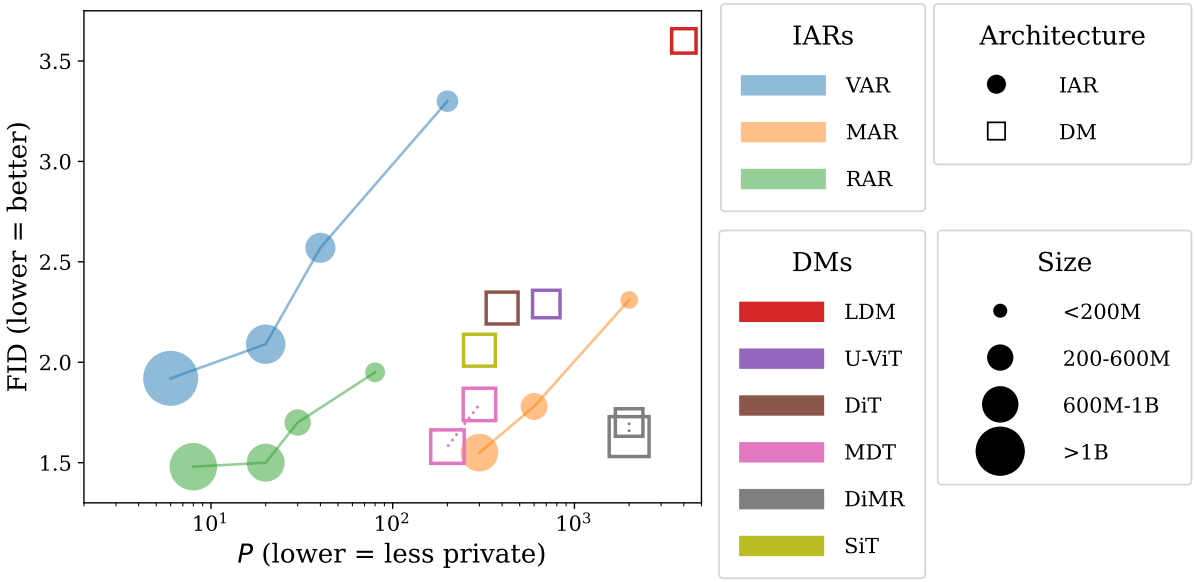


Figure 9.5.1. DI success for IARs vs DMs. We report the generative quality expressed with the FID score vs the number of suspect samples P required to carry out DI.

Overall Performance and Comparison to DMs. We present our results in Figure 9.5.1, evaluating the overall privacy leakage and comparing IARs to DMs based on the number of required samples (P) to perform DI. Recall that a lower P under the DI framework indicates greater privacy vulnerability, as it means fewer data points are needed to reject the null hypothesis—H_0. Our findings indicate that the same trend observed in MIAs extends to DI. Overall, models with a higher TPR@FPR=1% in Table 9.5.1 for MIAs also require smaller suspect sets P for DI. Specifically, DI shows that larger models exhibit greater privacy leakage, with VAR-$d30$ and RAR-XXL being the most vulnerable. Crucially, our results clearly demonstrate that IARs are significantly more susceptible to privacy leakage than DMs. While MDT shows lower generative quality (as indicated by a higher FID

score), it requires substantially more samples for DI (higher P value), resulting in much lower privacy leakage.

Why do We (Again) Observe Higher Leakage of IARs? MIAs are the backbone of the DI framework, extracting features from the samples to capture differences between members and non-members. When they succeed more for one class of the models, we expect that DI will also perform better for that class. With MIAs, we observe higher leakage of IARs, which stems from the increased difference between the distributions of the MIA-specific score for member and non-member samples. Because we use these scores to perform the t-test, when the difference between these distributions increase, we need a smaller P to reject H_0. Importantly, all insights about leakage from MIAs (section 9.5.1) also hold for DI. Results for correlation (table D.2.1) and DI performance with Unified MIA as the feature extractor (table D.4.1) corroborate the ones for MIA, and provide an alternative perspective into the privacy of IARs.

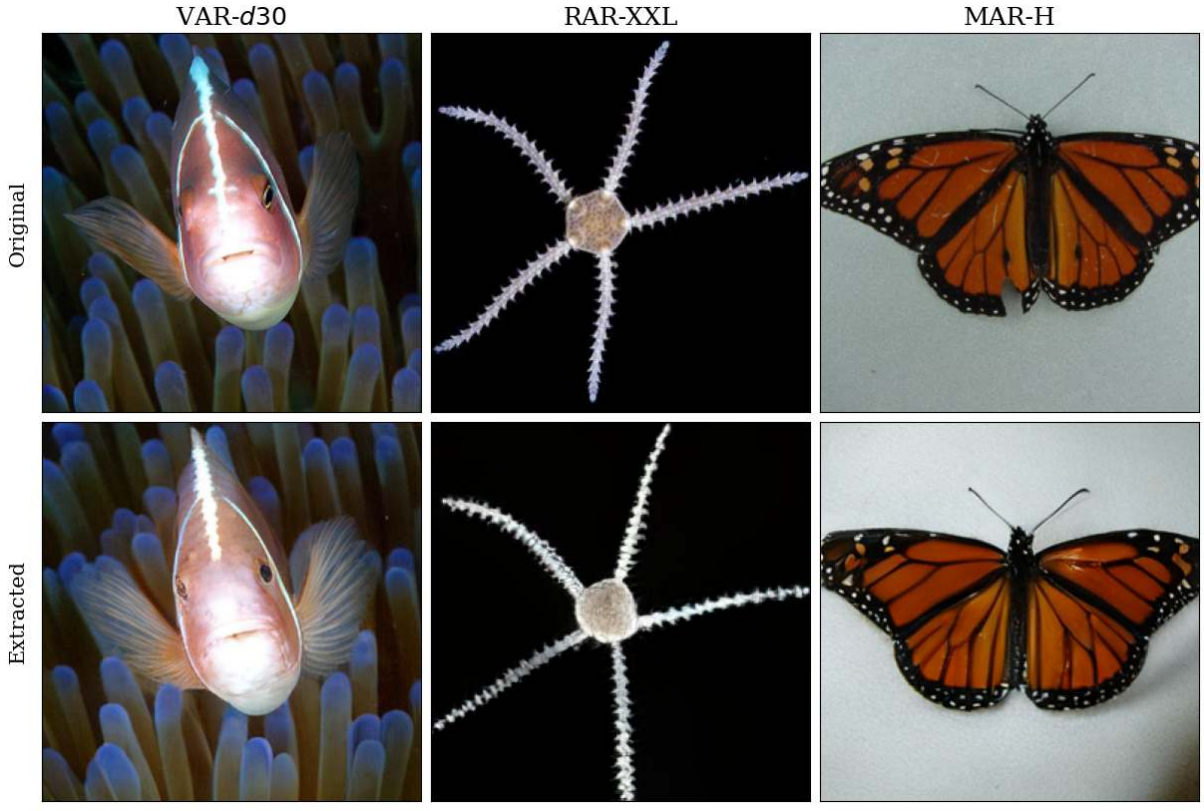


Figure 9.5.2. Extracted Training Samples. We note that IARs can reconstruct verbatim images from their training data. The first row shows the original training samples and the second one presents the extracted images.

9.5.3. Extracting Training Data from IARs

To analyze memorization in IARs, we design a novel training data extraction attack for IARs. This attack builds on elements of data extraction attacks for LLMs [19] and DMs [48]. Integrating elements from both domains is required since IARs operate on tokens (similarly to LLMs), which are then decoded and returned as images (similarly to DMs). In particular, we make the observation that, on the token level, IARs exhibit a similar behavior that was previously observed for LLMs [19]. Namely, for memorized samples, they tend to complete the *correct ending* of a token sequence when prompted with the sequence's prefix. We exploit this behavior and 1) identify candidate samples that might be memorized, 2) generate them by starting from a prefix in their token space, and sampling the remaining tokens from the IAR, and finally 3) compare the generated image with the original candidate image. We report a sample as memorized when the generated image is near identical to the original image. In the following, we detail the individual building blocks of the attack.

1) Candidate Identification. To reduce the computational costs, we do not simply generate a large pool of images, but identify promising candidate samples that might be memorized, before generation. Specifically, we feed an entire tokenized image t into the IAR, which predicts the full token sequence $\hat{t}$ in a *single step*. Then, we compute the distance between original and predicted sequence, $d(t,\hat{t})$, which we use to filter promising candidates. This approach is efficient, since for IARs the entire token sequence can be processed at once, significantly faster than if we sampled them iteratively. For VAR and RAR we use per-token logits, and apply greedy sampling, with $d(t,\hat{t}) = 100 - \frac{100 \cdot \sum_{i=1}^{N} \mathbb{1}(t_i = \hat{t_i})}{N}$—an average prediction error. For MAR, we sample 95% of the tokens from the remaining 5% unmasked in a single step, and set $d(t,\hat{t}) = ||t - \hat{t}||_2^2$, as MAR's tokens are continuous. Following the intuition that $\hat{t}$ is memorized if $\hat{t} = t$, for each model, for each class we select top-5 samples with the smallest d, and obtain 5000 candidates per model. Our candidate identification steps greatly improves the extraction efficiency over previous approaches [48]. We show the success of our filtering in section D.12.3.

2) Generation. Then, following the methodology established for LLMs by [19]. for each candidate we select the first i tokens as a prefix. The parameter i is a hyperparameter and we present our best choices for the models in Table D.12.1. We perform iterative greedy sampling of the remaining tokens in the sequence for VAR and RAR, and for MAR we sample from the DM batch by batch. We do not use classifier-free guidance during generation. We note that our method *does not* produce false positives, *i.e.,* we do not generate samples from the validation set.

3) Assessment. Finally, we decode the obtained $\hat{t}$ into images, and assess the

similarity to the original t. Following Wen et al. [244], we use SSCD [176] score to calculate the similarity, and set the threshold $\tau = 0.75$ such that every sample with a similarity $\geq \tau$ will be considered as memorized.

Results. In fig. 9.5.2 we show example memorized samples from VAR-$d30$, RAR-XXL, and MAR-H. We are not able to extract memorized images from smaller versions of these IARs. In table 9.5.4 we see that the extent of memorization is severe, with VAR-$d30$ memorizing **698** images. We observe lower memorization for MAR-H and RAR-XXL, which is intuitive, as results from Sections 9.5.1, and 9.5.2 show that VAR-$d30$ is the most vulnerable to MIA and DI. Surprisingly, there is no memorization in token space, *i.e.,* $t \neq \hat{t}$, we observe it only in the pixel space. We provide more examples of memorized images in section D.12.1.

Table 9.5.4. Count of Extracted Training Samples per IAR.

Model	VAR-$d30$	MAR-H	RAR-XXL
Count	698	5	36

Memorization Insights. Many memorized samples follow a pattern: their backgrounds deviate from the "default" or typical scene, as shown in fig. D.12.2 and section D.12.1. We hypothesize that when a prefix contains part of this "unusual" background, the IAR is conditioned to reproduce the specific training image that originally featured it. Additionally, several extracted images appear as poorly executed center crops with skewed proportions—see, for instance, the wine bottle in fig. D.12.3. These findings suggest memorization is driven by distinct visual cues in the prefix and can lead to the generation of replicas of its training data. Moreover, the same **5** samples were extracted from both VAR-$d30$ and RAR-XXL, *i.e.,* the same 5 training images are memorized by both models. One sample is memorized by both VAR-$d30$ and MAR-H (Fig. D.12.2 and D.12.1),suggesting some images are more prone to memorization across architectures.

Our results contrast with findings on DMs [48], where extracting training data requires far more computation. The high memorization in IARs likely stems from their size, as VAR-$d30$ has 2.1B parameters—more than twice the number of parameters in DMs investigated in prior work. Importantly, our results also show a link between IAR size and memorization, with bigger IARs memorizing more. Scaling laws suggest that as IARs grow larger, their performance improves, but so does their tendency to memorize, making privacy risks more severe in high-capacity models.

9.6. Mitigation Strategies

Our privacy assessment methods rely on precise outputs from IARs to be effective. We exploit this insight to design defenses that mitigate privacy risks by perturbing model outputs, *e.g.,* with random noise. For VAR and RAR, we noise the logits, while for MAR, we add noise to continuous tokens after sampling. Our preliminary evaluation in section D.11 shows that the defenses are insufficient for VAR and RAR, as reducing the success of privacy attacks is achieved at the cost of substantially lower performance. In contrast, our proposed defense helps to protect MAR even more, with a relatively low drop in performance. However, MAR already exhibits the lowest success rate of the privacy attacks. This further emphasizes that leveraging diffusion techniques is a promising direction towards strong privacy safeguards for IARs, though further investigation is needed to confirm its effectiveness.

9.7. Discussion and Conclusions

IARs are an emerging competitor to DMs, matching or surpassing them in image quality at a higher generation speed. However, our comprehensive analysis demonstrates that IARs *empirically* exhibit significantly higher privacy risks than DMs, given the current state of privacy attacks against the respective model types. Concretely, we develop novel MIA for IARs that leverages components of the strongest MIAs from LLMs and DMs to reach an extremely high **86.38%** TPR@FPR=1%, as opposed to merely 6.38% for the strongest *DM-specific* MIAs in respective DMs. Our DI method further confirms the high privacy leakage from IARs by showing that only 6 samples are required to detect dataset membership, compared to at least 200 for reference DMs of comparable image generation utility. We also create a new data extraction attack for IARs that reconstructs even up to 698 training images from VAR-$d30$, while previous work showed only 50 images extracted from DMs. Our results indicate the fundamental privacy-utility trade-off for IARs, where their higher performance comes at the cost of more severe privacy leakage. We explore preliminary mitigation strategies inspired primarily by diffusion-based approaches, however, the initial results indicate that dedicated privacy-preserving techniques are necessary. Our findings highlight the need for stronger safeguards in the deployment of IARs, especially in sensitive applications.

10. Conclusions and Final Remarks

This dissertation addressed two complementary aspects of generative modelling: the development of reliable models for solving demanding real-world scientific problems, with a particular focus on high-energy physics at CERN, and the safeguarding of intellectual value in machine learning, both in trained models and in the data that underpins them. The contributions of this work spanned both dimensions, ensuring that generative systems could be applied to pressing scientific challenges while also remaining protected against misuse.

The first part of the dissertation concentrated on the design of reliable generative models for high-energy physics simulations. Conventional Monte Carlo methods, while highly accurate, are computationally prohibitive, and standard machine learning alternatives such as conditional GANs often suffer from mode collapse and limited fidelity. These shortcomings restrict their usefulness in faithfully reproducing the variability of particle collisions. To address this challenge, we introduced the *Selective Diversity Increase GAN (SDI-GAN)* in Chapter 4, which selectively increased diversity in generated samples without sacrificing physical plausibility. Building on this result, we developed *ExpertSim* in Chapter 5, a Mixture-of-Generative-Experts framework designed to capture the inherently multimodal nature of calorimeter responses. Together, these contributions demonstrated that generative models could provide both accurate and computationally efficient alternatives to traditional simulation pipelines at CERN.

As generative models matured and proved their utility in such scientific domains, their growing value also highlighted a second challenge: the need to safeguard trained models against unauthorized replication. Encoder architectures in particular represent significant intellectual assets due to the resources invested in their training and their wide applicability across tasks, yet they are vulnerable to model stealing attacks. In Chapter 6, we introduced *Bucks for Buckets (B4B)*, the first active defense against encoder stealing. This method monitored query coverage in the embedding space, detected extensive probing indicative of extraction, and injected adaptive perturbations to disrupt adversarial reconstructions while preserving performance for legitimate users. By establishing an effective protection mechanism, this work demonstrated how valuable machine learning models can be shielded against exploitation.

The third focus of the dissertation was the protection of training data, which constitutes a core part of the value of generative systems. Previous research on membership inference attacks suggested that it might be possible to identify whether individual samples were part of a model's training set. However, we showed in Chapter 7 that such techniques provided overly optimistic results when applied to modern large-scale diffusion models trained on billions of internet-sourced images. To overcome these limitations, we proposed a new evaluation protocol and dataset that enabled more realistic assessments and revealed that reliable sample-level inference remained extremely challenging. We then advanced from single-sample to dataset-level auditing by introducing *Copyrighted Data Identification (CDI)* in Chapter 8, a framework that aggregated membership signals across many samples and applied statistical testing to determine whether a dataset had been used in training. Finally, in Chapter 9, we extended our analysis to the emerging class of image autoregressive models. We demonstrated that these architectures were particularly prone to privacy leakage, being able to reproduce verbatim training samples, and adapted our dataset identification methods to audit this new generative paradigm. These contributions provided data owners with more powerful and reliable tools for verifying the use of their datasets in large-scale generative models.

In summary, this dissertation advanced the state of generative modelling along two essential dimensions. On the one hand, it established methods for building models that are reliable enough to support demanding scientific applications, exemplified by detector simulations in high-energy physics. On the other hand, it developed strategies to safeguard both models and data, ensuring that the intellectual value embedded in generative systems remains with their creators and rightful owners. Together, these contributions represent a step toward a trustworthy and accountable ecosystem for generative machine learning.

10.1. Thesis Contributions

A. Generative Models for High-Energy Physics Simulation

- **Publication:** "Selectively increasing the diversity of GAN-generated samples," ICONIP 2022 (CORE A, 140 MEiN p.) [81], Chapter 4
  Introduces SDI-GAN, a method that selectively increases diversity of GAN-generated samples to address mode collapse in conditional GANs for calorimeter simulation.
  Author's contribution: First author; developed the idea, implemented the method, ran experiments, analysed results, and wrote the manuscript.

- **Publication:** "ExpertSim: Fast Particle Detector Simulation Using Mixture of Generative Experts," ECAI 2025 (CORE A, 140 MEiN p.) [44], Chapter 5
  Proposes a mixture-of-experts design combining specialised SDI-GAN models to capture multimodal calorimeter responses with improved fidelity and compute efficiency.
  Author's contribution: Second author; co-designed the model architecture, built individual expert models, analysed results, refined the method, and co-authored the paper.

B. Protection of Machine Learning Models

- **Publication:** "Bucks for Buckets (B4B): Active Defenses Against Stealing Encoders," NeurIPS 2023 (CORE A*, 200 MEiN p.) [27], Chapter 6
  Introduces the first active defense against encoder stealing by monitoring latent-space query coverage and injecting adaptive perturbations that degrade stolen models while preserving utility for legitimate users.
  Author's contribution: Equal first author; designed the mechanism for monitoring latent-space query coverage, built the end-to-end attack and defense pipeline, implemented experiments, analysed outcomes, and led the manuscript preparation.

C. Protection of Training Data

- **Publication:** "Towards More Realistic Membership Inference Attacks on Large Diffusion Models," WACV 2024 (CORE A, 140 MEiN p.) [33], Chapter 7
  Shows that prior membership inference evaluations overstate performance for large-scale diffusion models and introduces a realistic evaluation setup enabling fair assessment.
  Author's contribution: Equal first author; identified evaluation flaws in previous work, designed the evaluation protocol, ran experiments, analysed results, and co-authored the paper.
- **Publication:** "CDI: Copyrighted Data Identification in Diffusion Models," CVPR 2025 (CORE A*, 200 MEiN p.) [83], Chapter 8
  Proposes a dataset-level auditing framework that aggregates membership signals and uses statistical testing to determine whether a dataset was used in model training.
  Author's contribution: Equal first author; co-developed the method, implemented the attacked models, ran experiments, analysed results, and co-authored the manuscript.

- **Publication:** “Privacy Attacks on Image Autoregressive Models,” ICML 2025 (CORE A*, 200 MEiN p.) [36], Chapter 9
  First work to propose privacy attacks against autoregressive image models, showing they can leak sample and dataset-level information and reproduce training samples verbatim.
  Author’s contribution: Equal first author; co-designed the sample and dataset-level auditing approach, co-implemented the experimental pipeline, ran experiments, analysed results, and co-authored the manuscript.

10.2. Detailed Thesis Contributions

10.2.1. Generative Adversarial Networks with Selective Generation Diversity for High-Energy Physics Simulation

Publication: Jan Dubiński, K. Deja, S. Wenzel, P. Rokita, and T. Trzciński. “Selectively increasing the diversity of GAN-generated samples.”, International Conference on Neural Information Processing (ICONIP), (2022). (CORE A, 140 MEiN p.)

Contribution: In Chapter 4, we addressed the problem that standard conditional GANs, although promising for detector simulations, often collapsed to a narrow set of modes. This mode collapse prevented them from reproducing the full variability of particle collision data and thereby reduced their utility for high-energy physics experiments at CERN. Existing diversity-enhancing techniques attempted to mitigate this problem but typically produced physically implausible samples that undermined simulation fidelity.

To solve this issue, we proposed *Selective Diversity GAN (SDI-GAN)*, which introduced a regularization term to encourage diversity only in ways consistent with variability in the training data. This ensured that the generator reproduced multimodal calorimeter responses while remaining faithful to real measurements. As shown in Figure 10.2.1, SDI-GAN generated responses that were both diverse and physically consistent.

Through extensive experiments on synthetic datasets and simulations of the Zero Degree Calorimeter at CERN, we demonstrated that SDI-GAN significantly improved both fidelity and diversity compared to baseline conditional GANs. The PhD candidate, as first author, devised, implemented, and evaluated the method, applied

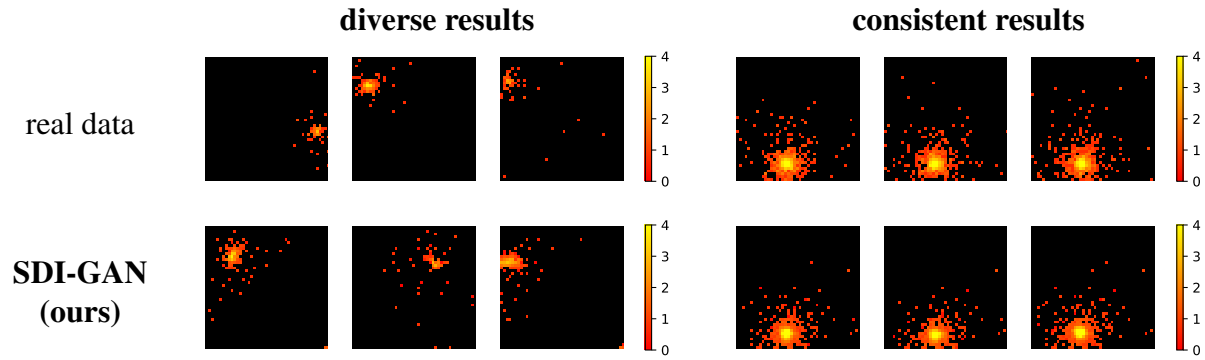


Figure 10.2.1. Examples of calorimeter response simulations. We propose a regularization of the GAN training loss that encourages the model to generate diverse outputs while maintaining consistent responses for the appropriate particles.

it to detector simulations in collaboration with CERN physicists, and prepared the manuscript for publication.

10.2.2. Generative Mixture-of-Experts Model for High-Energy Physics Simulation

Publication: Patryk Będkowski, **Jan Dubiński**, Filip Szatkowski, Kamil Deja, Przemysław Rokita, and Tomasz Trzciński. "ExpertSim: Fast Particle Detector Simulation Using Mixture-of-Generative-Experts.", European Conference on Artificial Intelligence (ECAI), (2025). (CORE A, 140 MEiN p.)

Contribution: In Chapter 5, we built on SDI-GAN to address a further problem: even with improved diversity, a single generative model remained insufficient to capture the intrinsically multimodal structure of calorimeter response data. Particle collisions produced complex distributions that exceeded the representational capacity of a single generator.

We solved this by proposing *ExpertSim*, a Mixture-of-Generative-Experts framework in which multiple specialized generators, each based on SDI-GAN, were coordinated by a gating network. This router assigned input conditions to the most suitable expert, enabling the model as a whole to capture the full multimodality of calorimeter responses. As illustrated in Figure 10.2.2, this modular design achieved high fidelity and robustness while being far more computationally efficient than traditional Monte Carlo simulations.

Our experiments confirmed that ExpertSim consistently outperformed single-model baselines in both accuracy and simulation speed. The PhD candidate, as co-author,

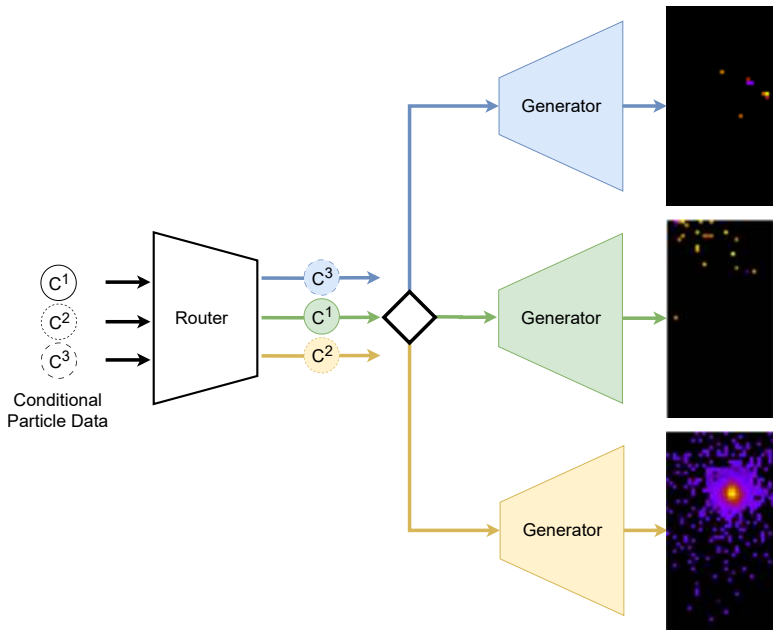


Figure 10.2.2. We introduce a generative mixture-of-experts framework for modeling multimodal data, where a trained gating network assigns particles to specialized expert models. This design enables a more faithful reproduction of the complexity inherent in particle physics data.

proposed the design of the framework, contributed the expert models, analyzed results, and participated in refining the approach and co-authoring the manuscript.

10.2.3. Active Defense Method against Stealing Encoder Models

Publication: **Jan Dubiński**, S. Pawlak, F. Boenisch, T. Trzciński, and A. Dziedzic. "Bucks for Buckets (B4B): Active Defenses Against Stealing Encoders.", Conference on Neural Information Processing Systems (NeurIPS), (2023). (CORE A*, 200 MEiN p.)

Contribution: In Chapter 6, we shifted focus from building reliable models to protecting them against misuse. A pressing problem in machine learning deployment was the rise of *model stealing attacks*, in which adversaries repeatedly queried an encoder model via an API and reconstructed a high-quality substitute. Existing defenses were largely passive and failed to stop determined attackers.

We solved this by introducing *Bucks for Buckets (B4B)*, the first active defense specifically against encoder stealing. B4B detected suspicious behavior by monitoring the coverage of user queries in the embedding space. Once excessive probing was detected, it adaptively injected noise into responses to degrade stolen models.

To counter Sybil-style attacks using multiple accounts, it also applied per-user transformations, ensuring that attackers could not combine outputs across accounts. As shown in Figure 10.2.3, this approach preserved high utility for legitimate users while making encoder replication infeasible.

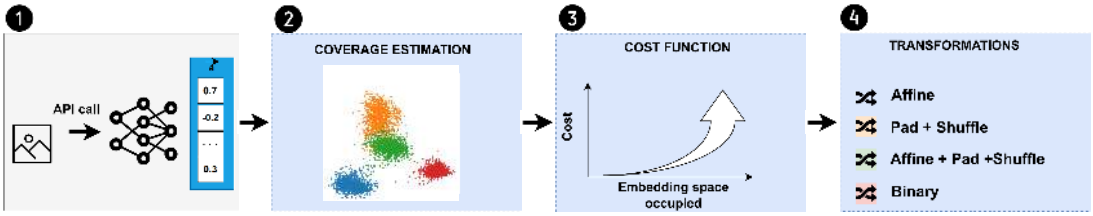


Figure 10.2.3. We present *B4B*, the first active defense against encoder stealing. *B4B* monitors how much of the embedding space user queries occupy, with large coverage indicating potential stealing, and adaptively injects noise to degrade the returned representations. To prevent attackers from bypassing this defense with multiple fake accounts, B4B applies per-user transformations. Legitimate users retain high-quality outputs, while adversaries are prevented from stealing a high-utility model.

We validated B4B across several encoder architectures and showed that it reliably detected and disrupted state-of-the-art stealing strategies without degrading performance for benign use. The PhD candidate, as equally contributing first author, developed the latent-space exploration metric, designed the defense pipeline, implemented the experiments, analyzed outcomes, and led the manuscript preparation.

10.2.4. Realistic Evaluation of Training Data Identification Methods for Large Diffusion Models

Publication: **Jan Dubiński**, A. Kowalczuk, S. Pawlak, P. Rokita, T. Trzciński, and P. Morawiecki. "Towards More Realistic Membership Inference Attacks on Large Diffusion Models.", Winter Conference on Computer Vision (WACV), (2024). (CORE A, 140 MEiN p.)

Contribution: In Chapter 7, we extended the safeguarding of intellectual value from models to training data. The core problem was whether large diffusion models, such as Stable Diffusion, disclosed training data through *membership inference attacks (MIAs)*. While MIAs had shown success on small benchmarks, prior evaluations were unrealistic for billion-scale models, giving a misleading impression of effectiveness.

We addressed this by designing a new evaluation protocol and dataset, enabling fair assessment of MIAs on diffusion models. Applying this framework, we showed

that prior setups overstated attack performance and that, in practice, detecting single training samples in large diffusion models remained extremely difficult. This revealed that privacy and copyright concerns persisted despite optimistic claims.

The PhD candidate, as equally contributing first author, identified these evaluation flaws, designed the new protocol, conducted experiments, analyzed results, and co-authored the manuscript.

10.2.5. Method for Identifying Copyrighted Data Used in Training of Large Diffusion Models

Publication: **Jan Dubiński**, A. Kowalczuk, F. Boenisch, and A. Dziedzic. "CDI: Copyrighted Data Identification in Diffusion Models.", Conference on Computer Vision and Pattern Recognition (CVPR), (2025). (CORE A*, 200 MEiN p.)

Contribution: In Chapter 8, we addressed a key limitation of MIAs: their focus on single samples made them too weak for copyright enforcement. The practical problem for data owners was verifying whether their *datasets*—rather than individual examples—had been used in training.

We proposed *Copyrighted Data Identification (CDI)*, a dataset-level auditing framework. CDI aggregated multiple membership signals across many candidate samples and applied statistical testing to determine whether a dataset had been used in training. As shown in Figure 10.2.4, this method enabled reliable detection of copyrighted datasets even in large diffusion models.

We validated CDI across multiple architectures and datasets, showing that it significantly outperformed prior approaches in both robustness and accuracy. The PhD candidate, as equally contributing first author, co-devised the method, implemented the framework, ran experiments, analyzed results, prepared figures, and co-authored the manuscript.

10.2.6. Analysis of Privacy Risks for Emerging Image Autoregressive Models

Publication: A. Kowalczuk, **Jan Dubiński**, F. Boenisch, and A. Dziedzic. "Privacy Attacks on Image Autoregressive Models." International Conference on Machine Learning (ICML), (2025). (CORE A*, 200 MEiN p.)

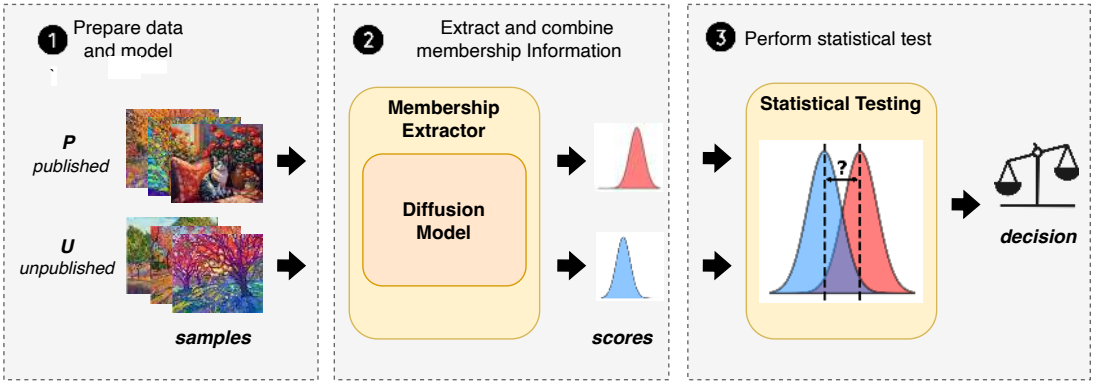


Figure 10.2.4. We introduce *CDI*, a framework that allows data owners to verify whether a given dataset was used to train a diffusion model. The process begins by collecting a *suspect dataset* (data suspected of being used in training) and a *control dataset* (similar data known not to have been used). The central question is: *Was this dataset part of the model's training data?* To answer, CDI extracts multiple membership signals from each sample, aggregates them into a unified score, and compares the score distributions of the two datasets via a statistical test. The outcome of this test determines whether the suspect dataset was used in the model's training set.

Contribution: In Chapter 9, we extended dataset auditing beyond diffusion architectures to the emerging class of *image autoregressive models (IARs)*. The open problem was whether these sequential generation architectures exposed training data differently than diffusion models.

We investigated this by adapting privacy attacks to the token-by-token generation process of IARs. Building on CDI, we developed dataset-level inference strategies tailored to autoregressive settings and demonstrated that these models were especially vulnerable. As illustrated in Figure 10.2.5, IARs not only leaked dataset-level information but could also reconstruct verbatim training samples, confirming severe privacy risks.

The PhD candidate, as equally contributing first author, co-designed the adapted auditing framework, implemented parts of the experimental pipeline, performed evaluations, analyzed results, prepared figures, and contributed to the manuscript.

10.3. Future Outlook of Generative Artificial Intelligence

This thesis was written at a moment when generative modelling is diversifying rather than converging. Diffusion models currently dominate visual domains, while autoregressive approaches are regaining attention in image generation, and large language models show the scalability of token-based architectures. Each family

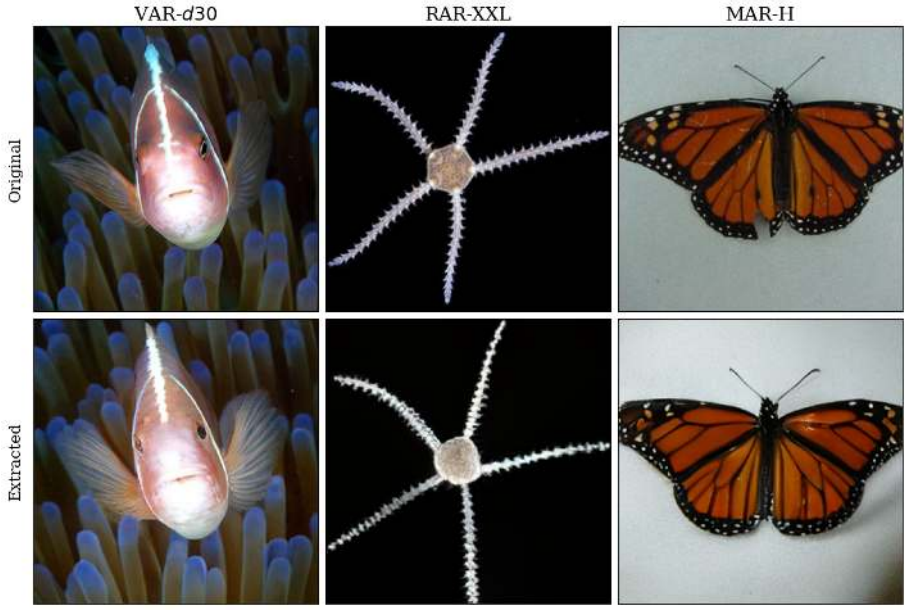


Figure 10.2.5. We extract training samples from image autoregressive models (IARs), demonstrating that they can reconstruct verbatim images from their training data. The first row shows original samples, while the second row shows the extracted images.

of methods has clear strengths and limitations, and their relative importance will likely continue to shift.

From the perspective of this work, the progress of image autoregressive models is of particular interest. Their similarity to language models opens the possibility of transferring alignment techniques that have already been developed in natural language processing. Reinforcement learning with human feedback or constraint-based decoding could in principle be adapted to enforce domain-specific requirements in visual tasks. Such methods would be especially relevant if autoregressive models are to complement or replace diffusion models in applications such as high-energy physics, where generated outputs must remain consistent with physical laws.

At the same time, the discussion around training data ownership is becoming more pressing. Legal disputes and regulation point to the need for systematic auditing of models and datasets. Methods such as membership inference, dataset inference, and copyrighted data identification, studied in this thesis, provide initial steps in this direction. They show that it is possible to test for training data usage, but also highlight the limitations of current techniques, which often reduce to binary signals. Moving beyond these first tools will be necessary if generative models are to be used responsibly in sensitive or regulated domains.

10.4. Open Questions

Although the contributions of this dissertation address several aspects of reliability and safety in generative modelling, they also leave a number of questions open.

Can image autoregressive models be applied to HEP simulation? Earlier chapters of this work analysed the potential of generative models for calorimeter simulation. So far, GANs and diffusion models have been the main candidates. However, the recent emergence of competitive image autoregressive models raises the question of whether their sequential token-based generation can also be applied in this setting. Their resemblance to large language models suggests that alignment strategies developed for text could be used to ensure consistency with physics-based constraints. Exploring this possibility would extend the set of tools available for scientific simulation and may provide benefits in controllability and interpretability.

How can we move from binary dataset detection to quantifying dataset usage? The methods developed in this thesis for membership and dataset inference provide only binary conclusions: a point or dataset was, or was not, used in training. For many applications, particularly copyright enforcement, it would be more useful to estimate the fraction of a dataset that contributed to training. This requires moving beyond simple thresholding and towards calibrated estimators that can aggregate weak per-sample evidence into quantitative measures of dataset usage. Initial ideas for this problem were outlined in our follow-up project proposal, but the development of reliable statistical tools for this task remains an open challenge.

10.5. Conclusion

This thesis analysed generative models from the perspective of both reliability and safety. In the first part, we proposed modifications to GANs and a mixture-of-experts approach for diffusion models, improving the diversity and fidelity of calorimeter simulations in high-energy physics. In the second part, we turned to the problem of protecting models and data. We introduced an active defense against model stealing, proposed more realistic evaluations of membership inference for diffusion models, developed a test for copyrighted data identification, and analysed privacy risks in autoregressive models. Taken together, these studies show that generative modelling must be considered not only as a tool for data synthesis but also as a system whose reliability and security are essential for its safe use.

The open questions identified above suggest that generative modelling will continue to raise both technical and practical challenges. Reliable and safe systems are not a finished product but an ongoing process, requiring adaptation as architectures evolve and as expectations of transparency and accountability increase. The methods presented here contribute to this process by combining advances in model design with tools for auditing and protection, pointing towards a research agenda at the intersection of generative modelling, scientific application, and data governance.

Publications Included in This Dissertation

1. **Jan Dubiński**, K. Deja, S. Wenzel, P. Rokita, T. Trzciński. *Selectively increasing the diversity of GAN-generated samples*. International Conference on Neural Information Processing (ICONIP), 2022. (CORE A, 140 MEiN p.)
2. P. Będkowski, **Jan Dubiński**, F. Szatkowski, K. Deja, P. Rokita, T. Trzciński. *ExpertSim: Fast Particle Detector Simulation Using Mixture of Generative Experts*. European Conference on Artificial Intelligence (ECAI), 2025. (CORE A, 140 MEiN p.)
3. **Jan Dubiński***, S. Pawlak*, F. Boenisch*, T. Trzciński, A. Dziedzic. *Bucks for Buckets (B4B): Active Defenses Against Stealing Encoders*. Conference on Neural Information Processing Systems (NeurIPS), 2023. (**CORE A***, 200 MEiN p.)
4. **Jan Dubiński***, A. Kowalczuk*, S. Pawlak, P. Rokita, T. Trzciński, P. Morawiecki. *Towards More Realistic Membership Inference Attacks on Large Diffusion Models*. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024. (CORE A, 140 MEiN p.)
5. **Jan Dubiński***, A. Kowalczuk*, F. Boenisch, A. Dziedzic. *CDI: Copyrighted Data Identification in Diffusion Models*. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025. (**CORE A***, 200 MEiN p.)
6. A. Kowalczuk*, **Jan Dubiński***, F. Boenisch, A. Dziedzic. *Privacy Attacks on Image AutoRegressive Models*. International Conference on Machine Learning (ICML), 2025. (**CORE A***, 200 MEiN p.)

Publications Not Included in This Dissertation

1. M. Podhajski, **Jan Dubiński**, F. Boenisch, A. Dziedzic, A. Pręgowska, T. P. Michalak. *On Stealing Graph Neural Network Models*. The 40th Annual AAAI Conference on Artificial Intelligence (AAAI), 2025. (**CORE A***, 200 MEiN p.)
2. Y. Djenouri, N. Belmecheri, T. Michalak, **Jan Dubiński**, A. N. Belbachir, A. Yazidi. *Learning Graph Representation of Agent Diffuser*. Accepted for International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2025. (**CORE A***, 200 MEiN p.)
3. M. Podhajski, **Jan Dubiński**, F. Boenisch, A. Dziedzic, A. Pręgowska, T. Michalak. *Efficient Model-Stealing Attacks Against Inductive Graph Neural Networks*. European Conference on Artificial Intelligence (ECAI), 2024. (CORE A, 140 MEiN p.)
4. S. Pawlak, F. Szatkowski, M. Bortkiewicz, **Jan Dubiński**, T. Trzciński. *Progressive Latent Replay for Efficient Generative Rehearsal*. International Conference on Neural Information Processing (ICONIP), 2022. (CORE A, 140 MEiN p.)
5. J. Kaleta, P. Skierś, **Jan Dubiński**, P. Korzeniowski, T. Trzciński, J. M. Tomczak, K. Deja. *JointDiffusion: Joint representation learning for generative, predictive, and self-explainable AI in healthcare*. Computerized Medical Imaging and Graphics, 2025.
6. **Jan Dubiński**, K. Deja, S. Wenzel, P. Rokita, T. Trzciński. *Machine learning methods for simulating particle response in the zero degree calorimeter at the ALICE experiment, CERN*. AIP Conference Proceedings, 2024.
7. M. Kita, **Jan Dubiński**, P. Rokita, K. Deja. *Generative diffusion models for fast simulations of particle collisions at CERN*. Polish Conference on Artificial Intelligence, 2024.
8. P. Będkowski, **Jan Dubiński**, K. Deja, P. Rokita. *Deep generative models for proton zero degree calorimeter simulations in ALICE, CERN*. Polish Conference on Artificial Intelligence, 2024.
9. K. Rogoziński, **Jan Dubiński**, P. Rokita, K. Deja. *Particle physics DL-simulation with control over generated data properties*. Polish Conference on Artificial Intelligence, 2024.

Additional 130 co-authored articles published as a member of the ALICE Collaboration.

Bibliography

[1] Clarifai, https://www.clarifai.com/. URL https://www.clarifai.com/.

[2] Cohere, https://cohere.ai. URL https://cohere.ai/.

[3] Openai, https://openai.com. URL https://openai.com/.

[4] Poet2, https://labs.hyperledger.org/labs/archived/sawtooth-poet2.html. URL https://labs.hyperledger.org/labs/archived/sawtooth-poet2.html.

[5] Google, https://google.github.io/snappy/. URL https://google.github.io/snappy/.

[6] Facebook zstd, https://github.com/facebook/zstd. URL https://github.com/facebook/zstd.

[7] Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997. ISSN 0098-3500. doi: 10.1145/279232.279236.

[8] Presented at the 8th International Conference on Calorimetry in High Energy Physics, pp. 362-369, Apr 1999.

[9] Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern recognition letters*, 31(11):1348–1358, 2010.

[10] Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[11] In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[12] In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[13] Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[14] Generative models for fast cluster simulations in the tpc for the alice experiment. In *Conference on Information Technology, Systems Research and Computational Physics*, pages 267–280. Springer, 2018.

[15] Bingan: Learning compact binary descriptors with a regularized gan. *NeurIPS*, 2018.

[16] White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pages 5558–5567. PMLR, 2019.

[17] Large scale image completion via co-modulated generative adversarial net-

works. *CoRR*, abs/2103.10428, 2021. URL https://arxiv.org/abs/2103.10428.

[18] Code repository for latent diffusion models., 2021. URL https://github.com/CompVis/latent-diffusion.

[19] Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[20] Code repository for torchprofile python library., 2021. URL https://github.com/zhijian-liu/torchprofile.

[21] Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

[22] Code repository for dit models, 2022. URL https://github.com/facebookresearch/DiT.

[23] Laion translated: 3b captions translated to english from laion5b, 2022. URL https://laion.ai/blog/laion-translated/. [Online].

[24] High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[25] Stability diffusion vae checkpoint, 2022. URL https://huggingface.co/stabilityai/sd-vae-ft-ema-original.

[26] Code repository for u-vit models ., 2023. URL https://github.com/baofff/U-ViT.

[27] Bucks for buckets (b4b): Active defenses against stealing encoders. *Advances in Neural Information Processing Systems*, 36:55237–55259, 2023.

[28] In *1:23-cv-00135-JLH*, 2023.

[29] Tight auditing of differentially private machine learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1631–1648, 2023.

[30] Zero time waste in pre-trained early exit neural networks. *Neural Networks*, 2023.

[31] Deep generative models for proton zero degree calorimeter simulations in alice, cern. *arXiv preprint arXiv:2406.03263*, 2024.

[32] Blind baselines beat membership inference attacks for foundation models. *arXiv preprint arXiv:2406.16201*, 2024.

[33] Towards more realistic membership inference attacks on large diffusion models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4860–4869, 2024.

[34] Minimal data requirement for realistic endoscopic image generation with stable diffusion. *International journal of computer assisted radiology and*

surgery, 19(3):531–539, 2024.

[35] Membership inference attacks cannot prove that a model was trained on your data. *arXiv preprint arXiv:2409.19798*, 2024.

[36] Privacy attacks on image autoregressive models. In *Forty-second International Conference on Machine Learning*, 2025.

[37] Jointdiffusion: Joint representation learning for generative, predictive, and self-explainable ai in healthcare. *Computerized Medical Imaging and Graphics*, page 102619, 2025.

[38] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. *arXiv preprint arXiv:1906.00830*, pages 1615–1631, 2018. URL https://www.usenix.org/conference/usenixsecurity18/presentation/adi.

[39] Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021.

[40] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22669–22679, 2023.

[41] Romain Beaumont. Clip retrieval: Easily compute clip embeddings and build a clip retrieval system with them, 2022.

[42] Martin Bertran, Shuai Tang, Aaron Roth, Michael Kearns, Jamie H Morgenstern, and Steven Z Wu. Scalable membership inference attacks via quantile regression. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[44] Patryk Będkowski, Jan Dubiński, Filip Szatkowski, Kamil Deja, Przemysław Rokita, and Tomasz Trzciński. Expertsim: Fast particle detector simulation using mixture-of-generative-experts, 2025. URL https://arxiv.org/abs/2508.20991.

[45] Bochuan Cao, Changjiang Li, Ting Wang, Jinyuan Jia, Bo Li, and Jinghui Chen. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=RRSltzPc7w.

[46] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[47] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914, 2022. doi: 10.1109/SP46214.2022.9833649.

[48] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. 2023.

[49] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, October 2021.

[50] CERN. Remote access (16.01.2024), 2024.

[51] Hongyan Chang, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Reza Shokri. Context-aware membership inference attacks against pre-trained large language models. *arXiv preprint arXiv:2409.13745*, 2024.

[52] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

[53] Sanyuan Chen, Shujie Liu, Long Zhou, Yanqing Liu, Xu Tan, Jinyu Li, Sheng Zhao, Yao Qian, and Furu Wei. Vall-e 2: Neural codec language models are human parity zero-shot text to speech synthesizers, 2024. URL `https://arxiv.org/abs/2406.05370`.

[54] Ting Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 2020.

[55] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. 2020.

[56] Yan Chen, Xueru Wang, Xiaobin Deng, Yilun Liu, Xi Chen, Yunwei Zhang, Lei Wang, and Hang Xiao. Mattergpt: A generative transformer for multi-property inverse design of solid-state materials, 2024.

[57] Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik G. Learned-Miller, and Chuang Gan. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

[58] Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela

Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, 2022.

[59] Adam Coates, Andrew Ng, and Honglak Lee. In *AISTATS*, 2011.

[60] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition, 1988. ISBN 978-0805802832.

[61] Tianshuo Cong, Xinlei He, and Yang Zhang. *CoRR*, abs/2201.11692, 2022. URL https://arxiv.org/abs/2201.11692.

[62] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Defossez. Simple and controllable music generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 47704–47720. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/94b472a1842cd7c56dcb125fb2765fbd-Paper-Conference.pdf.

[63] Jesse C Cresswell, Brendan Leigh Ross, Gabriel Loaiza-Ganem, Humberto Reyes-Gonzalez, Marco Letizia, and Anthony L Caterini. Caloman: Fast generation of calorimeter showers with density estimation on learned manifolds. *arXiv preprint arXiv:2211.15380*.

[64] Damai Dai, Chengqi Deng, Chenggang Zhao, R. Xu, Huazuo Gao, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *Annual Meeting of the Association for Computational Linguistics*, 2024.

[65] Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

[66] Erik Daxberger, Floris Weers, Bowen Zhang, Gunter, et al. Mobile v-moes: Scaling down vision transformers via sparse mixture-of-experts. *arXiv preprint arXiv:2309.04354*, 2023.

[67] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2018.

[68] Kamil Deja, Jan Dubiński, Piotr Nowak, Sandro Wenzel, Przemysław Spurek, and Tomasz Trzcinski. End-to-end sinkhorn autoencoder with noise generator. *IEEE Access*, 9:7211–7219, 2021. doi: 10.1109/ACCESS.2020.3048622.

[69] G Dellacasa, X Zhu, M Wahn, FM Staley, V Danielian, TL Karavicheva, DP Mikhalev, N Carrer, M Gheata, G Stefanek, et al. Alice technical design report of the zero degree calorimeter (zdc). Technical report, ALICE, 1999.

[70] G. et al. Dellacasa. 1999.

[71] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[72] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

[73] Prafulla Dhariwal and Alexander Nichol. *Advances in Neural Information Processing Systems*, 2021.

[74] Riccardo Di Sipio, Michele Faucci Giannelli, Sana Ketabchi Haghighat, and Serena Palazzo. Dijetgan: a generative-adversarial network approach for the simulation of qcd dijet events at the lhc. *Journal of high energy physics*.

[75] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

[76] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, 2022.

[77] Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. Flocks of stochastic parrots: Differentially private prompt learning for large language models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[78] Haonan Duan, Adam Dziedzic, Mohammad Yaghini, Nicolas Papernot, and Franziska Boenisch. On the privacy risk of in-context learning. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[79] Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8717–8730. PMLR, 23–29 Jul 2023.

[80] Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*, 2024.

[81] Jan Dubiński, Kamil Deja, Sandro Wenzel, Przemysław Rokita, and Tomasz Trzciński. Selectively increasing the diversity of gan-generated samples. In

Neural Information Processing, 2023. doi: 10.1007/978-3-031-30105-6_22.

[82] Jan Dubiński, Kamil Deja, Sandro Wenzel, et al. Machine learning methods for simulating particle response in the zero degree calorimeter at the alice experiment, cern. abs/2306.13606, 2023.

[83] Jan Dubiński, Antoni Kowalczuk, Franziska Boenisch, and Adam Dziedzic. In *The IEEE CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2025.

[84] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[85] Adam Dziedzic, Nikita Dhawan, Muhammad Ahmad Kaleem, Jonas Guan, and Nicolas Papernot. On the difficulty of defending self-supervised learning against model extraction. In *International Conference on Machine Learning*, 2022.

[86] Adam Dziedzic, Haonan Duan, Muhammad Ahmad Kaleem, Nikita Dhawan, Jonas Guan, Yannis Cattan, Franziska Boenisch, and Nicolas Papernot. Dataset inference for self-supervised models. *Advances in Neural Information Processing Systems*, 35:12058–12070, 2022.

[87] Adam Dziedzic, Muhammad Ahmad Kaleem, Yu Shen Lu, and Nicolas Papernot. Increasing the cost of model extraction with calibrated proof of work. In *International Conference on Learning Representations*, 2022. URL `https://arxiv.org/abs/2201.09243`.

[88] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 585–605. Springer, 2015.

[89] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2019.

[90] Martin Erdmann, Jonas Glombitza, and Thorben Quast. Precise simulation of electromagnetic calorimeter showers using a wasserstein generative adversarial network. *Computing and Software for Big Science*, 3(1), Jan 2019. ISSN 2510-2044.

[91] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020.

[92] Zach Evans, Julian D. Parker, CJ Carr, Zack Zukowski, Josiah Taylor, and Jordi Pons. Long-form music generation with latent diffusion, 2024. URL `https://arxiv.org/abs/2404.10301`.

[93] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav

Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. Beyond english-centric multilingual machine translation, 2020.

[94] Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens, 2024. URL https://arxiv.org/abs/2410.13863.

[95] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 2022.

[96] Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. Scaling diffusion transformers to 16 billion parameters, 2024. URL https://arxiv.org/abs/2407.11633.

[97] Vitaly Feldman. In *ACM SIGACT Symposium on Theory of Computing*, 2020.

[98] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22466–22477, October 2023.

[99] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[100] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2012–2020. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/frosst19a.html.

[101] Xiaomeng Fu, Xi Wang, Qiao Li, Jin Liu, Jiao Dai, and Jizhong Han. Model will tell: Training membership inference for diffusion models. *arXiv preprint arXiv:2403.08487*, 2024.

[102] Jean-loup Gailly and Mark Adler. zlib compression library. 2004. URL http://www.dspace.cam.ac.uk/handle/1810/3486.

[103] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23164–23173, October 2023.

[104] Aditya Golatkar, Alessandro Achille, Ashwin Swaminathan, and Stefano

Soatto. Training data protection with compositional diffusion models, 2024.

[105] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[106] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[107] Jean-Bastien Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning. *Computer Vision and Pattern Recognition*, 2020.

[108] Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. Demix layers: Disentangling domains for modular language modeling, 2021.

[109] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis, 2024. URL `https://arxiv.org/abs/2412.04431`.

[110] Vincent Hanke, Tom Blanchard, Franziska Boenisch, Iyiola Emmanuel Olatunji, Michael Backes, and Adam Dziedzic. Open llms are necessary for current private adaptations and outperform their closed alternatives. In *Thirty-Eighth Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[111] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models, 2018. URL `https://arxiv.org/abs/1705.07663`.

[112] Jamie Hayes, Ilia Shumailov, Christopher A. Choquette-Choo, Matthew Jagielski, George Kaissis, Katherine Lee, Milad Nasr, Sahra Ghalebikesabi, Niloofar Mireshghallah, Meenatchi Sundaram Mutu Selva Annamalai, Igor Shilov, Matthieu Meeus, Yves-Alexandre de Montjoye, Franziska Boenisch, Adam Dziedzic, and A. Feder Cooper. Strong membership inference attacks on massive datasets and (moderately) large language models. 2025.

[113] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[114] Charles Herrmann, Richard Strong Bowen, and Ramin Zabih. Channel selection using gumbel softmax. In *European Conference on Computer Vision*.

Springer, 2020.

[115] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[116] Dominik Hintersdorf, Lukas Struppek, Kristian Kersting, Adam Dziedzic, and Franziska Boenisch. Finding nemo: Localizing neurons responsible for memorization in diffusion models. In *Thirty-Eighth Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[117] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[118] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[119] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.

[120] Jessica N Howard, Stephan Mandt, Daniel Whiteson, and Yibo Yang. Learning to simulate high energy particle collisions from unlabeled data. *Scientific Reports*, 2022.

[121] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[122] Jing Huang, Diyi Yang, and Christopher Potts. Demystifying verbatim memorization in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10711–10732, 2024.

[123] Huggingface. Fine tuning stable diffusion, 2023. [Online].

[124] Sebastien Incerti, Ioanna Kyriakou, MA Bernal, MC Bordage, Z Francis, Susanna Guatelli, V Ivanchenko, M Karamitros, N Lampe, Sang Bae Lee, et al. Geant4-dna example applications for track structure simulations in liquid water: A report from the geant4-dna project. *Medical physics*, 45(8): e722–e739, 2018.

[125] Matthew Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot. High accuracy and high fidelity extraction of neural networks. *USENIX Security Symposium*, 2020.

[126] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1937–1954, 2021.

[127] B. D. Jovanovic and P. S. Levy. A look at the rule of three. *The American*

Statistician, 51(2):137–139, 1997. ISSN 00031305. URL http://www.jstor.org/stable/2685405.

[128] Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Yanqing Liu, Yichong Leng, Kaitao Song, Siliang Tang, Zhizheng Wu, Tao Qin, Xiang-Yang Li, Wei Ye, Shikun Zhang, Jiang Bian, Lei He, Jinyu Li, and Sheng Zhao. Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models, 2024. URL https://arxiv.org/abs/2403.03100.

[129] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

[130] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[131] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[132] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[133] Sanjay Kariyappa, Atul Prakash, and Moinuddin K Qureshi. Maze: Data-free model stealing attack using zeroth-order gradient estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13814–13823, 2021.

[134] Gulrukh Khattak, Sofia Vallecorsa, and Federico Carminati. Three dimensional energy parametrized generative adversarial networks for electromagnetic shower simulation. pages 3913–3917, 10 2018.

[135] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[136] Mikołaj Kita, Jan Dubiński, Przemysław Rokita, and Kamil Deja. Generative diffusion models for fast simulations of particle collisions at cern, 2024. URL https://arxiv.org/abs/2406.03233.

[137] Fei Kong, Jinhao Duan, RuiPeng Ma, Heng Tao Shen, Xiaoshuang Shi, Xiaofeng Zhu, and Kaidi Xu. An efficient membership inference attack for the diffusion model by proximal initialization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rpH9FcCEV6.

[138] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng

Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. Hunyuanvideo: A systematic framework for large video generative models, 2025. URL https://arxiv.org/abs/2412.03603.

[139] James T Kost and Michael P McDermott. Combining dependent p-values. *Statistics & Probability Letters*, 60(2):183–190, 2002.

[140] Yamuna Krishnamurthy, Chris Watkins, and Thomas Gaertner. Improving expert specialization in mixture of experts, 2023. URL https://arxiv.org/abs/2302.14703.

[141] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[142] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision*, 128(7):1956–1981, 2020.

[143] Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion, 2025. URL https://arxiv.org/abs/2501.06158.

[144] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2020.

[145] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization, 2024. URL https://arxiv.org/abs/2406.11838.

[146] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 880–895, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384544. URL https://doi.org/10.1145/3460120.3484575.

[147] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang.

Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 2021.

[148] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems*.

[149] Chuanjian Liu, Yunhe Wang, Kai Han, Chunjing Xu, and Chang Xu. Learning instance-wise sparsity for accelerating deep models. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[150] Guan-Horng Liu, Tianrong Chen, Evangelos Theodorou, and Molei Tao. Mirror diffusion models for constrained and watermarked generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=XPWEtXzlLy.

[151] Qihao Liu, Zhanpeng Zeng, Ju He, Qihang Yu, Xiaohui Shen, and Liang-Chieh Chen. Alleviating distortion in image generation via multi-resolution diffusion models. *arXiv preprint arXiv:2406.09416*, 2024.

[152] Rui Liu, Yixiao Ge, Ching Lam Choi, Xiaogang Wang, and Hongsheng Li. Divco: Diverse conditional image synthesis via contrastive generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[153] Yupei Liu, Jinyuan Jia, Hongbin Liu, and Neil Zhenqiang Gong. Stolenencoder: Stealing pre-trained encoders in self-supervised learning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2115–2128, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394505. doi: 10.1145/3548606.3560586. URL https://doi.org/10.1145/3548606.3560586.

[154] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[155] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[156] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. In *Proceedings of the 18th European Conference on Computer Vision (ECCV)*, volume 15135 of *Lecture Notes in Computer Science*, pages 23–40. Springer, 2024.

[157] Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *Proceedings of ICLR 2021: 9th International Conference on Learning Representationsn*, 2021.

[158] Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. Llm dataset inference: Did you train on my dataset?, 2024. URL https://arxiv.org/abs/2406.06443.

[159] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[160] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 2022.

[161] Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. Membership inference attacks against language models via neighbourhood comparison. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11330–11343, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.719. URL https://aclanthology.org/2023.findings-acl.719.

[162] Vinicius Mikuni, Benjamin Nachman, and Mariel Pettee. Fast point cloud generation with diffusion models in high energy physics. *Physical Review D*, 108(3), 2023. ISSN 2470-0029. doi: 10.1103/physrevd.108.036025. URL http://dx.doi.org/10.1103/PhysRevD.108.036025.

[163] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014. URL https://arxiv.org/abs/1411.1784.

[164] Saman Motamed and Farzad Khalvati. Inception augmentation generative adversarial network. *CoRR*, abs/2006.03622, 2020. URL https://arxiv.org/abs/2006.03622.

[165] Yuki Nagai, Y. Uchida, S. Sakazawa, and Shin'ichi Satoh. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval, 7:3–16*, 2018.

[166] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

[167] Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. Evomoe: An evolutional mixture-of-experts training framework via dense-to-sparse gate. *arXiv preprint arXiv:2112.14397*, 2021.

[168] OpenReview.net. Review of, 2023. [Online].

[169] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 4954–4963, 2019.

[170] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction poisoning: Towards defenses against dnn model stealing attacks, 2020. URL `https://arxiv.org/abs/1906.10908`.

[171] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Calogan: Simulating 3d high energy particle showers in multilayer electromagnetic calorimeters. *Physical Review D*.

[172] Michela Paganini, Luke de Oliveira, and Benjamin Nachman. Calogan: Simulating 3d high energy particle showers in multi-layer electromagnetic calorimeters with generative adversarial networks. *Physical Review D*, 97, 2017.

[173] David Park, Seungjoo Yoo, Hyojin Bahng, Jaegul Choo, and Noseong Park. Megan: Mixture of experts of generative adversarial networks for multimodal image generation. 07 2018.

[174] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[175] Justin N. M. Pinkney. Pokemon blip captions, 2022.

[176] Ed Pizzi, Sreya Dutta Roy, Sugosh Nagavara Ravindra, Priya Goyal, and Matthijs Douze. A self-supervised descriptor for image copy detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14532–14542, 2022.

[177] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

[178] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[179] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[180] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, 2021.

[181] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. *arXiv preprint arXiv:2204.06125*, 2022.

[182] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural*

Information Processing Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf.

[183] Reuters. Lawsuits accuse ai content creators of misusing copyrighted work, 2023.

[184] Reuters. Artists take new shot at stability, midjourney in updated copyright lawsuit, 2023.

[185] Reuters, 2023.

[186] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, et al. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34, 2021.

[187] Andres C. Rodríguez, Tomasz Kacprzak, Aurelien Lucchi, Adam Amara, Raphaël Sgier, Janis Fluri, Thomas Hofmann, and Alexandre Réfrégier. Fast cosmic web simulations with generative adversarial networks. *Computational Astrophysics and Cosmology*, 5(1), Nov 2018. ISSN 2197-7909. URL http://dx.doi.org/10.1186/s40668-018-0026-4.

[188] Karol Rogoziński, Jan Dubiński, Przemysław Rokita, and Kamil Deja. Particle physics dl-simulation with control over generated data properties, 2024. URL https://arxiv.org/abs/2405.14049.

[189] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 2021.

[190] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bj Ommer. Compvis stable diffusion v1-4 model card, 2022. URL https://huggingface.co/CompVis/stable-diffusion-v1-4. [Online].

[191] Ruichen Rong, Shuang Jiang, Lin Xu, Guanghua Xiao, Yang Xie, Dajiang J Liu, Qiwei Li, and Xiaowei Zhan. *GigaScience*, 10(2), 02 2021. ISSN 2047-217X. URL https://doi.org/10.1093/gigascience/giab005.

[192] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[193] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[194] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014.

[195] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019.

[196] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL https://arxiv.org/abs/2202.00512.

[197] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. *Advances in Neural Information Processing Systems*, 2016.

[198] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.

[199] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 30105–30118. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/sauer23a.html.

[200] Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 2020.

[201] Christoph Schuhmann. Clip+mlp aesthetic score predictor, 2022. URL https://github.com/christophschuhmann/improved-aesthetic-predictor. [Online].

[202] Christoph Schuhmann. Laion-aesthetics, 2022. URL https://laion.ai/blog/laion-aesthetics/. [Online].

[203] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models,

2022.

[204] Maximilian Seitzer, August 2020. Version 0.3.0.

[205] Zeyang Sha, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. Can't steal? cont-steal! contrastive stealing attacks against image encoders. 2022. URL https://arxiv.org/abs/2201.07513.

[206] Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, and Ben Y. Zhao. Tree-rings watermarks: Invisible fingerprints for diffusion images. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *32nd USENIX Security Symposium (USENIX Security 23)*, volume 36, pages 2187–2204, Anaheim, CA, 2023. USENIX Association. ISBN 978-1-939133-37-3. URL https://www.usenix.org/conference/usenixsecurity23/presentation/shan.

[207] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2016.

[208] Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. Model stealing attacks against inductive graph neural networks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1175–1192. IEEE, 2022.

[209] Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zWqr3MQuNs.

[210] Yuge Shi, Siddharth N, Brooks Paige, and Philip Torr. Variational mixture-of-experts autoencoders for multi-modal deep generative models. In *Advances in Neural Information Processing Systems*, 2019.

[211] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, Los Alamitos, CA, USA, may 2017. IEEE Computer Society. doi: 10.1109/SP.2017.41. URL https://doi.ieeecomputersociety.org/10.1109/SP.2017.41.

[212] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 69–75, 2020. doi: 10.1109/SPW50608.2020.00028.

[213] Malcolm Slaney and Michael Casey. Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *IEEE Signal processing magazine*, 25(2):

128–131, 2008.

[214] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

[215] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=St1giarCHLP.

[216] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92c3b916311a5517d9290576e3ea37ad-Paper.pdf.

[217] Florian Strub, Romaric Gaudel, and Jérémie Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 11–16. ACM, September 2016. doi: 10.1145/2988450.2988456. URL http://dx.doi.org/10.1145/2988450.2988456.

[218] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

[219] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 2019.

[220] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. Hart: Efficient visual generation with hybrid autoregressive transformer, 2024. URL https://arxiv.org/abs/2410.10812.

[221] Midjourney Team. https://www.midjourney.com/, 2022.

[222] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders, 2017. URL https://arxiv.org/abs/1703.00395.

[223] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction, 2024. URL https://arxiv.org/abs/2404.02905.

[224] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders, 2017.

[225] Florian Tramèr, F. Zhang, A. Juels, M. Reiter, and T. Ristenpart. Stealing machine learning models via prediction apis. *USENIX Security Symposium*,

2016.

[226] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE transactions on services computing*, 14(6):2073–2089, 2019.

[227] Jean-Baptiste Truong, Pratyush Maini, Robert J. Walls, and Nicolas Papernot. Data-free model extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[228] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017.

[229] Thanh Van Le, Hao Phung, Thuan Hoang Nguyen, Quan Dao, Ngoc N. Tran, and Anh Tran. Anti-dreambooth: Protecting users from personalized text-to-image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2116–2127, October 2023.

[230] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008, 2017. URL `https://arxiv.org/abs/1706.03762`.

[231] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

[232] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020.

[233] Duy Tin Vo and Richard Khoury. Language identification on massive datasets of short message using an attention mechanism cnn, 2019.

[234] Vladimir Vovk and Ruodu Wang. Combining p-values via averaging. *Biometrika*, 107(4):791–808, 2020.

[235] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang,

Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL https://arxiv.org/abs/2503.20314.

[236] Feifei Wang, Zhentao Tan, Tianyi Wei, Yue Wu, and Qidong Huang. Simac: A simple anti-customization method for protecting face privacy against text-to-image synthesis of diffusion models, 2024.

[237] Wenhao Wang, Adam Dziedzic, Michael Backes, and Franziska Boenisch. Localizing memorization in ssl vision encoders. In *Thirty-Eighth Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[238] Wenhao Wang, Muhammad Ahmad Kaleem, Adam Dziedzic, Michael Backes, Nicolas Papernot, and Franziska Boenisch. Memorization in self-supervised learning improves downstream generalization. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[239] Wenhao Wang, Adam Dziedzic, Grace C. Kim, Michael Backes, and Franziska Boenisch. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[240] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[241] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[242] Zhenting Wang, Chen Chen, Lingjuan Lyu, Dimitris N. Metaxas, and Shiqing Ma. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=f8S3aLm0Vp.

[243] Ryan Webster, Julien Rabin, Loic Simon, and Frederic Jurie. On the de-duplication of laion-2b, 2023.

[244] Yuxin Wen, Yuchen Liu, Chen Chen, and Lingjuan Lyu. Detecting, explaining, and mitigating memorization in diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

[245] Florian Wiesner, Matthias Wessling, and Stephen Baek. Towards a physics foundation model, 2025. URL https://arxiv.org/abs/2509.13805.

[246] Maksymilian Wojnar. Applying generative neural networks for fast simulations of the alice (cern) experiment. *arXiv preprint arXiv:2407.16704*, 2024.

[247] Maksymilian Wojnar. Even faster simulations with flow matching: A study of zero degree calorimeter responses, 2025.

[248] Maksymilian Wojnar, Emilia Majerz, and Witold Dzwinel. Fast simulation of the zero degree calorimeter responses with generative neural networks. *Computing and Software for Big Science*, 9(1):1, 2025.

[249] Yutong Wu, Han Qiu, Tianwei Zhang, Lin Jiwei, and Meikang Qiu. Water-

marking pre-trained encoders in contrastive learning. *ArXiv*, abs/2201.08217, 2022.

[250] Weixin Xie, Jianhang Zhang, Qin Xie, Chaojun Gong, Youjun Xu, Luhua Lai, and Jianfeng Pei. Accelerating discovery of novel and bioactive ligands with pharmacophore-informed generative models, 2024.

[251] Haotian Xue, Chumeng Liang, Xiaoyu Wu, and Yongxin Chen. Toward effective protection against diffusion-based mimicry through score distillation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NzxCMe88HX.

[252] Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tianchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2019.

[253] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. 2021.

[254] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, 2018.

[255] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[256] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation, 2024. URL https://arxiv.org/abs/2411.00776.

[257] Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. In *Forty-first International Conference on Machine Learning*, 2024.

[258] Shengfang Zhai, Huanran Chen, Yinpeng Dong, Jiajun Li, Qingni Shen, Yansong Gao, Hang Su, and Yang Liu. Membership inference on text-to-image diffusion models via conditional likelihood discrepancy. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 74122–74146. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/874411a224a1934b80d499068384808b-Paper-Conference.pdf.

[259] Anqi Zhang and Chaofeng Wu. Adaptive pre-training data detection for large language models via surprising tokens. *arXiv preprint arXiv:2407.21248*, 2024.

[260] Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. Min-k%++: Improved baseline for detecting pre-training data from large language models. *arXiv preprint arXiv:2404.02936*, 2024.

[261] Qian Zhang, Xiangzi Dai, Ninghua Yang, Xiang An, Ziyong Feng, and Xingyu Ren. Var-clip: Text-to-image generator with visual auto-regressive modeling, 2024. URL `https://arxiv.org/abs/2408.01181`.

[262] Yixuan Zhang, Teng Long, and Hongbin Zhang. Stable diffusion for the inverse design of microstructures, 2024.

[263] Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, 2022.

[264] Bihe Zhao, Pratyush Maini, Franziska Boenisch, and Adam Dziedzic. Unlocking post-hoc dataset inference with synthetic data. 2025.

[265] Xiaosen Zheng, Tianyu Pang, Chao Du, Jing Jiang, and Min Lin. Intriguing properties of data attribution on diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024.

[266] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 2022.

[267] Simiao Zuo, Qingru Zhang, Chen Liang, Pengcheng He, Tuo Zhao, and Weizhu Chen. In *NAACL 2022*, 2022.

A. Supplement for Bucks for Buckets (B4B): Active Defenses Against Stealing Encoders

A.1. Broader Impact

The goal of our work is to actively defend self-supervised encoders against model stealing attacks. Since we are directly defending encoders, any negative societal impacts of our work are minimal. One potentially negative impact could be the degradation of performance for legitimate users. However, as shown in our experimental results, we are able to preserve high utility for standard users.

A.2. Limitations

We show how our defense method is tuned for SimSiam and DINO. There are more types of SSL encoders that can be tested with our method. The B4B defense method requires tuning the parameters, such as the number of occupied buckets that is allowed without any penalty for the cost function, or the selection of the transformations. These steps are rather difficult to automate but can be replaced with more data-driven approaches. For example, instead of designing a cost function from scratch, one could create an ML model to obtain a cost for a given occupation of the representation space. We explain more details in the Section A.3.2.

A.3. Alternative Building Blocks to Instantiate B4B

While we present a reference implementation of B4B in our work that instantiates the three building blocks with (1) Local Sensitive Hashing, (2) Utility of the Representations, and (3) a set of concrete transformations, there exists a multitude of alternatives to concretely implement our B4B framework. In the following, we present these alternatives, grouped by building block.

A.3.1. Alternative Estimation of the Coverage of Embedding Space

We also explore alternative methods to measure the distances between representations for queries sent to an API. One of them is to apply the cosine distance (where for two representations a and b, it is defined as: $1 - \frac{a^T b}{||a||_2 \cdot ||b||_2}$) since it can be measured between individual data points in a pair-wise fashion. If the total pair-wise cosine distance between representations for a given user is small, then the user queries presumably come from a single downstream task distribution. Otherwise, a user might be malicious and would like to cover a large part of the representation space, then the total pair-wise cosine distance for the user's representations would be high. Note that in this case, the cosine distance can be replaced with any other distance measure, such as the Minkowski distance. We opt for the LSH in our reference implementation, since it is much less expensive to compute than cosine distance. LSH requires only $2^{12} = 4096$ buckets that can be expressed as a binary table with the same number of elements, which requires in the worst case iterating over all of them to count how many are occupied. With more than 4096 queries sent by a given user, the computation on the LSH is sublinear $< O(n)$ with respect to the number of user queries. For the cosine distance approach, the required computation grows quadratically $O(n^2)$ with the number of queries.

A.3.2. Alternative Cost Functions

The cost functions can be designed from scratch manually or learned, for example, via an ML model, such as a neural network or SVM. In our initial version, the function was designed manually, where the underlying premise is that once a specified number of buckets is occupied, the cost should grow exponentially. Instead of defining such a function or providing the high-level parameters for functions that we contributed, one could learn an ML model that for a given number of buckets occupied, it should output an estimated cost, or even directly, the desired σ (standard deviation) of the noise added to representations. This method requires a relatively large number of data points to be provided for training the model, however, lowers the burden on a defender to either decide on the specific function or adjust its parameters. Thus, it could be more user-friendly, for example, not necessitating any mathematical background, but can be precise enough to obtain the desired behavior.

Note that instead of adding the calibrated noise (proportional to the estimated cost) to the representations, we could rather require a given user to pay a higher monetary cost for queries that cover a large fraction of the representation space, or force a user to solve a puzzle in a form of the proof-of-work [87], wait a specified amount of time via proof-of-elapsed time (PoET) [4], or prove that a specified amount

of disk space was reserved [88, 88]. For example, consider the approach with PoET. A user sends queries to the API, which we cost based on their occupation of the embedding space. The user is sent a waiting time. The users' resource (e.g., a CPU) has to be occupied for this specific waiting time without performing any work. At the end of the specified amount of time, the user sends proof of the elapsed time, which can be easily verified by the server. PoET requires access to specialized hardware, for example, secure CPU instructions that are becoming widely available in consumer and enterprise processors. If a user does not own such hardware, proof of elapsed time could be produced using a service exposed by a cloud provider (e.g., Azure VMs that feature TEE 2). Note that if a server sends the time based on the calculated cost, the adversary might learn the cost function. Instead, the exact waiting time should be split in random *subwaiting* times and sent to the user one by one. Thus, a server should rather have a few rounds of exchange with the client to incur the additional cost.

A.3.3. Alternative to Transformations

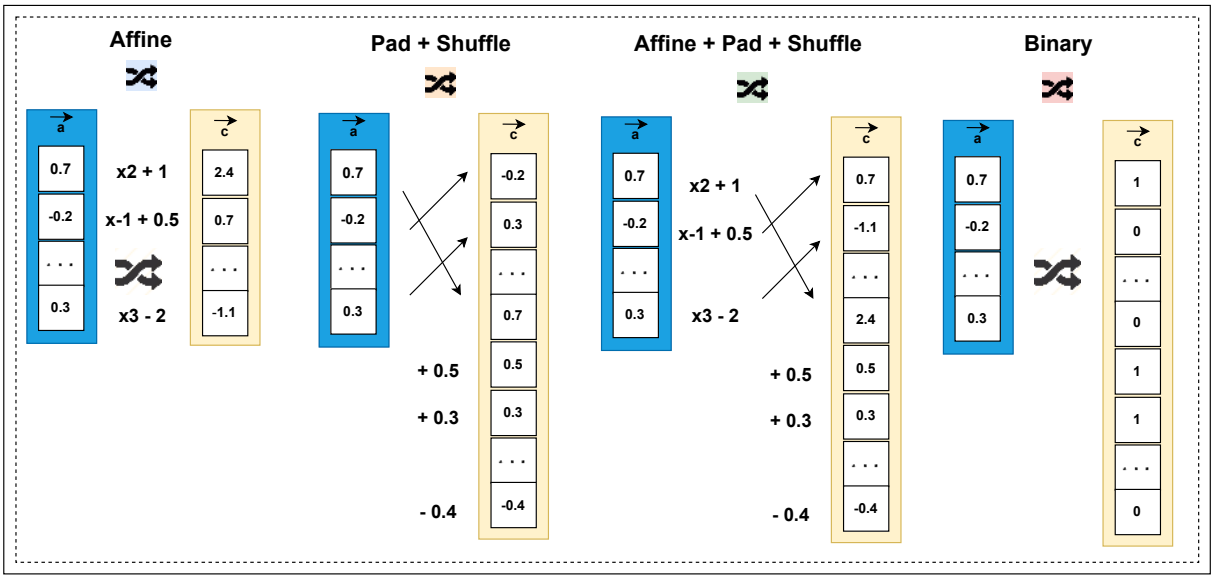


Figure A.3.1. Overview on Transformations. We depict the inner-workings of the transformations considered in this work.

As an alternative to the transformations used within this work (see Figure A.3.1), one could use a different set of transformations or combinations thereof. The padding can be done with different constant values and combined with adding constant values within the representations. The padding and adding the constant values can be followed by shuffling the elements within the representations. We can apply the

affine of binary transformations on top of the padding and shuffling. Additionally, we can also use other pre-defined linear transformations like rotations or shearing.

The representations could also be compressed to smaller vectors and the compression rate would depend on the occupation of the representation space, for example, the higher the number of occupied buckets in our hash table, the more compressed the output representations could be. Such representations could be compressed via FFT, a cosine transform, or standard compression techniques such as snappy [5]. If the information from the representations should not be lost, then the lossless compression techniques can be applied, for instance, zstd [6]. The only requirement of the compression techniques is to ensure that they do not decrease the accuracy on downstream tasks for legitimate users.

Another alternative is to incorporate an additional neural network layer for transforming the returned representations. The training of this supplementary layer should primarily focus on preserving the usability of the representations for legitimate users. This approach grants the API provider with additional capabilities, as it allows for the utilization of customized training objectives. For instance, if the API provider employs LSH (Locality-Sensitive Hashing) to estimate the coverage of the representation space, they can leverage buckets and train the additional layer to maintain high-quality representations exclusively for frequently-used buckets and their surrounding areas, while not prioritizing the rest of the representation space. This approach safeguards legitimate users from any adverse effects, as their coverage of the representation space is minimal. Simultaneously, it ensures that adversaries are unable to exploit representations from the entire representation space.

A.4. Sybil Attacks

We consider an adversary who generates n sybil accounts to steal the encoder from the API. For each of the accounts, the representations are transformed in a different way. Therefore, to replicate the victim model using all the obtained representations, the adversary has to map these representations into one single space. This can be done, for example, by training a neural network to perform the mapping.

We assume the adversary obtains $\{N_1, N_2, \ldots, N_n\}$ many representations from the victim for each of the n sybil accounts. Without loss of generality, we assume the adversary maps them back to the embedding space of the first sybil account. To learn the mapping, the adversary can apply different strategies.

A.4.1. Sybil Strategies

We present three potential approaches that Sybils might want to apply to circumvent our defense. Consider three users: A, B, and C, with their respective datasets D_A, D_B, and D_C, each with different distributions to maximize extraction effectiveness. First, user A is selected to unify representations from other users B and C. User A would have to query from at least two different datasets D_B and D_C, while other users would act legitimately. Sybil attackers want to deploy as many users as possible but with more fake accounts, user A incurs high coverage of the representation space, and this is prevented by our single-user defense. In all other cases neither of the sybil users can act legitimately, thus they are already affected by the single-user defense. Second, user A would query from their own dataset D_A and partially from dataset D_B. Then user B would query from their own dataset D_B and partially from dataset D_C, and so on. This method is the most inconspicuous but requires a number of remappings that scales super-exponentially with the number of fake accounts, which is impractical. Finally, each user would query from their respective dataset, for example, user B would query from dataset D_B and additionally from a remapping dataset, *e.g.,* D_A. Representations could be unified by mapping them to A's representations. The last approach as well as all other cases reduce to the minimum of remapping between representations of a pair of users. We show that our defense cuts such attempts short by ensuring that the remapping between representations is prohibitive even for a pair of users.

A.5. Additional Related Work

One of the main workhorse techniques used in the encoders is contrastive learning, where the representations are trained so that the positive pairs (two augmented versions of the same image) have similar representations while negative pairs (augmentations of two different images) have representations which are far apart.

SimSiam utilizes Siamese networks (two encoders with shared weights) but with a simplified training process and architecture. In contrast to the previous frameworks, such as SimCLR [54], SimSiam's authors show that negative samples are unnecessary and collapsing solutions can be avoided by applying the projection head to of one of the encoders, and a stop-gradient operation to the other. SimSiam minimizes the negative cosine similarity between two randomly augmented views of the same image from the Siamese encoders, which is expressed via a symmetrized loss [107]. This creates a simple yet highly effective representation learning method.

DINO is another popular representation learning framework. While SimSiam

uses CNNs, DINO employs vision transformers (ViTs). It trains a student and teacher encoder with the same architecture, updating the teacher with an (exponential moving) average of the student. Different random transformations of the same image are passed through both encoders. The student receives smaller image crops, forcing it to generate representations restoring parts of the original image. The training objective is minimizing cross-entropy loss between teacher and student representations.

A.6. Additional Experimental Results

A.6.1. Details on Experimental Setup

The end-to-end experiments on stealing SimSiam and ViT DINO encoders were done using 3 A100 GPUs. Detailed experiments including mapping, transformations and the evaluation was performed using a single computer equipped with two Nvidia RTX 2080 Ti GPUs.

A.6.2. Datasets Used

CIFAR10 [141]: The CIFAR10 dataset consists of 32x32 colored images with 10 classes. There are 50000 training images and 10000 test images.

SVHN [166]: The SVHN dataset contains 32x32 coloured images with 10 classes. There are roughly 73000 training images, 26000 test images and 530000 "extra" images.

ImageNet[71]: Larger sized coloured images with 1000 classes. As is commonly done, we resize all images to be of size 224x224. There are approximately 1 million training images and 50000 test images.

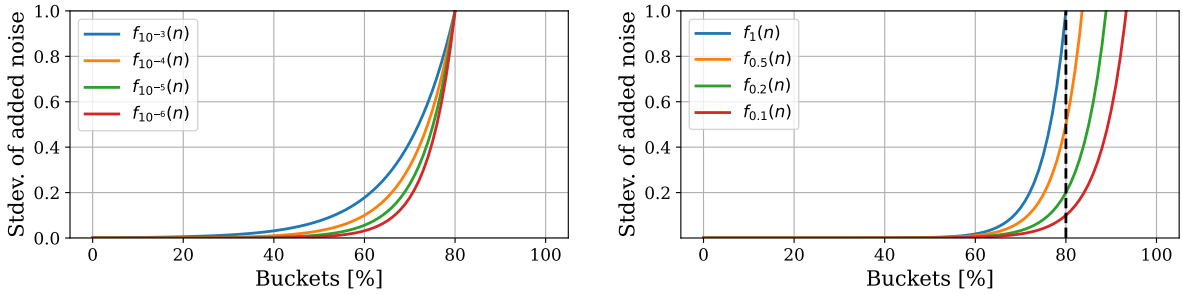
STL10 [59]: The STL10 dataset contains 96x96 coloured images with 10 classes. There are 5000 training images, 8000 test images, and 100000 unlabeled images.

LAION-5B [203] The LAION-5B dataset consists of 5,85 billion CLIP-filtered image-text pairs. The dataset was crawled from publically available internet.

A.6.3. More Results for the End2End Empirical Evaluation

We consider fine-tuning parameters β, λ, and α for our cost function and the intuitive meaning behind these parameters. In general, our recommendation is to adjust the parameter β that specifies how many buckets are allowed to be filled by users' downstream tasks. On the other hand, when parameter λ is increased, this causes a higher amount of added noise before we reach the number of buckets

specified by β, which lowers the performance of a given downstream task relatively early. For example, a higher value of λ in Figure 6.4.2, would cause an increase in the amount of added noise much earlier than for the target value of β. Finally, parameter α controls the amount of noise once the number of buckets specified by β is reached. Thus, in Figure 6.4.2, we set $\alpha = 1$ and the maximum standard deviation of the added Gaussian noise is 1.



(a) Cost Function for different λ parameter values.

(b) Cost Function for different α parameter values.

Figure A.6.1. Effects of λ and α parameters on the Cost Function. We present the Cost Function for α=1, β=80 and different values of λ (*left*) and $\lambda = 10^{-6}$, β=80 and different values of α (*right*).

Table A.6.1. Stealing and Using Encoders With and Without our Defense. The model used in the experiments is Simsiam, with the following parameters for the cost function $\lambda = 10^{-4}$, $\alpha = 1$, and $\beta = 80\%$, and the number of buckets equal to 2^{12}. Due to the higher value of the parameter λ, we observe lower performance on downstream tasks for the attackers since the magnitude of noise added to the representations is higher. However, for more complicated tasks than CIFAR10, this change might cause a potential drop in accuracy for the legitimate users.

User	Defense	# Queries	Dataset	Type	CIFAR10	STL10	SVHN	F-MNIST
Legit	None	All	Task	Query	90.41 ±0.02	95.08 ±0.13	75.47 ±0.04	91.22 ±0.11
Legit	B4B	ALL	TASK	Query	90.02 ±0.1	94.88 ±0.17	74.72 ±0.13	91.76 ±0.09
Attack	None	50K	ImgNet	Steal	65.2 ±0.03	64.9 ±0.01	62.1 ±0.01	88.5 ±0.01
Attack	B4B	50K	ImgNet	Steal	28.22 ± 0.04	26.62 ± 0.02	19.62 ± 0.02	78.41 ±0.01
Attack	None	100K	ImgNet	Steal	68.1 ±0.03	63.1 ±0.01	61.5 ±0.01	89.0 ±0.07
Attack	B4B	100K	ImgNet	Steal	17.73 ± 0.18	15.59 ± 0.61	19.53 ± 0.01	55.11 ± 0.05
Sybil	B4B	50K+50K	ImgNet	Steal	33.43 ± 0.03	31.18 ± 0.12	22.91 ± 0.01	75.35 ± 0.05

Table A.6.2. Stealing and Using Encoders With and Without our Defense. The model used in the experiments is Simsiam, with the following parameters for the cost function $\lambda = 10^{-6}$, $\alpha = 1$, and $\beta = 50\%$, and the number of buckets equal to 2^{12}. This experiment corresponds to considering 50% of buckets filled as a too-large coverage of the embedding space. This improves the defense but again might potentially harm the performance of more complicated tasks than CIFAR10 since they could occupy more buckets than 50%.

User	Defense	# Queries	Dataset	Type	CIFAR10	STL10	SVHN	F-MNIST
Legit	None	All	Task	Query	90.41 ±0.02	95.08 ±0.13	75.47 ±0.04	91.22 ±0.11
Legit	B4B	ALL	TASK	Query	90.27 ±0.07	95.12 ±0.13	74.94 ±0.16	91.66 ±0.05
Attack	None	50K	ImgNet	Steal	65.2 ±0.03	64.9 ±0.01	62.1 ±0.01	88.5 ±0.01
Attack	B4B	50k	ImgNet	Steal	15.52 ± 0.37	12.57 ± 0.23	19.53 ± 0.01	23.17 ± 0.01
Attack	None	100K	ImgNet	Steal	68.1 ±0.03	63.1 ±0.01	61.5 ±0.01	89.0 ±0.07
Attack	B4B	100k	ImgNet	Steal	16.27 ± 0.04	13.93 ± 0.35	19.54 ± 0.02	54.69 ± 0.02
Sybil	B4B	50k+50k	ImgNet	Steal	30.14 ± 0.01	29.57 ± 0.08	19.99 ± 0.03	71.72 ± 0.01

Table A.6.3. Stealing and Using Encoders With and Without our Defense. The model used in the experiments is Simsiam, with the following parameters for the cost function $\lambda = 10^{-6}$, $\alpha = 1$, and $\beta = 30\%$, and the number of buckets equal to 2^{12}. Since the value of parameter β is decreased substantially to 30%, we observe a drop in accuracy for legitimate users. For example, more than 1% for CIFAR10. In the next Table A.6.4, we show that by also decreasing the parameter α, we can attenuate this harmful effect and retain higher accuracy for legitimate users. In case of an attack, for 100k stealing queries, we observe much lower accuracy levels than for $\beta = 50\%$ shown in Table A.6.2.

User	Defense	# Queries	Dataset	Type	CIFAR10	STL10	SVHN	F-MNIST
Legit	None	All	Task	Query	90.41 ±0.02	95.08 ±0.13	75.47 ±0.04	91.22 ±0.11
Legit	B4B	ALL	TASK	Query	88.1 ±0.11	94.92 ±0.11	74.37 ±0.02	91.67 ±0.07
Attack	None	50K	ImgNet	Steal	65.2 ±0.03	64.9 ±0.01	62.1 ±0.01	88.5 ±0.01
Attack	B4B	50k	ImgNet	Steal	30.82 ± 0.09	26.37 ± 0.07	21.87 ± 0.03	66.0 ± 0.02
Attack	None	100K	ImgNet	Steal	68.1 ±0.03	63.1 ±0.01	61.5 ±0.01	89.0 ±0.07
Attack	B4B	100k	ImgNet	Steal	9.57 ± 0.17	9.83 ± 0.09	19.57 ± 0.01	27.06 ± 0.46
Sybil	B4B	50k+50k	ImgNet	Steal	29.15 ± 0.02	28.67 ± 0.06	19.98 ± 0.03	70.62 ± 0.03

Table A.6.4. Stealing and Using Encoders With and Without our Defense. The model used in the experiments is Simsiam, with the following parameters for the cost function $\lambda = 10^{-6}$, $\alpha = 0.1$, and $\beta = 30\%$, and the number of buckets equal to 2^{12}. Due to the lower performance on downstream tasks observed in Table A.6.3 while keeping the parameter β fixed to 30% and λ fixed to 10^{-6}, we decrease the value of parameter α to 0.1, which increases the performance of legitimate users on their downstream tasks. In this experiment, we also carry out a sybil attack with more accounts (4 instead of 2), but observe that this modification does not improve the performance of the attacker. With more accounts, a sybil has to sacrifice more queries for the remappings between the representations from different accounts. Additionally, note that each account introduces a different remapping error by the dint of different transformations applied to each account by B4B.

User	Defense	# Queries	Dataset	Type	CIFAR10	STL10	SVHN	F-MNIST
Legit	None	All	Task	Query	90.41±0.02	95.08±0.13	75.47±0.04	91.22±0.11
Legit	B4B	50K	CIFAR10	Query	90.17 ±0.1	94.92 ±0.09	74.97 ±0.13	91.71 ±0.08
Attack	None	50K	ImgNet	Steal	65.2 ±0.03	64.9 ±0.01	62.1 ±0.01	88.5 ±0.01
Attack	B4B	50K	ImgNet	Steal	19.95 ±0.19	15.54 ± 0.34	19.57 ± 0.01	23.50 ± 0.19
Attack	None	100K	ImgNet	Steal	68.1 ±0.03	63.1 ±0.01	61.5 ±0.01	89.0 ±0.07
Attack	B4B	100K	ImgNet	Steal	10.35 ± 0.19	12.37 ± 0.69	19.34 ± 0.01	68.93 ± 0.17
Sybil	B4B	4×25K	ImgNet	Steal	33.15 ± 0.04	30.23 ± 0.07	20.87 ± 0.01	72.19 ± 0.02

A.6.4. B4B vs Static Noise Addition Defenses

We compare our B4B against the current state-of-the-art baseline defense, namely adding a static addition of noise to all the returned representations (as proposed in [85] (Section A.4),[153, 205]). For the Table A.6.5, we use the same setup as in Table 6.4.1 (with an ImageNet pre-trained encoder).

Our results show the following insights:

1. If the amount of noise is small ($\sigma = 0.1$) then the performance drop is negligible but for both a legitimate user (row 2) and an adversary (row 6). In this case, the defense does not affect the adversary at all (compare rows 5 & 6).
2. If the amount of noise is large ($\sigma = 10$) then the performance drop is large for both a legitimate user (row 3) and an adversary (row 7). In this case, the encoder is worthless for legitimate users since the performance is too low.

A.6.5. Additional Embedding Space Coverage Experiments

We present additional experiments on measuring the coverage of the representation space.

First, we use the same set-up as from Table 6.4.1 - SimSiam with ResNet50 pretrained on ImageNet. When querying the encoder with ImageNet-Full (includes all 1000 classes) and LAION-5B datasets, they both occupy a large fraction of the representation space of the victim encoder, as shown on Figure A.6.2. In contrast,

CIFAR10 covers the smallest portion of the representation space as the simplest dataset tested. ImageNet-Dogs (with only 118 classes for dog breeds) falls in the middle, occupying more space than CIFAR10 but less than ImageNet-Full and LAION-5B. Its intermediate coverage aligns with its mid-level difficulty compared to the other datasets. As indicated by representation space coverage, stealing the encoder is similarly effective with ImageNet-Full and LAION-5B datasets, as both datasets cover a large fraction of the representation space. Overall, Figure A.6.2 demonstrates that: 1) our B4B can successfully protect the encoder model even from attackers stealing with data that was not used to train the model (LAION-5B in this case) and 2) while providing clean representation for users querying from downstream tasks that are part of more complicated datasets (ImageNet-Dogs).

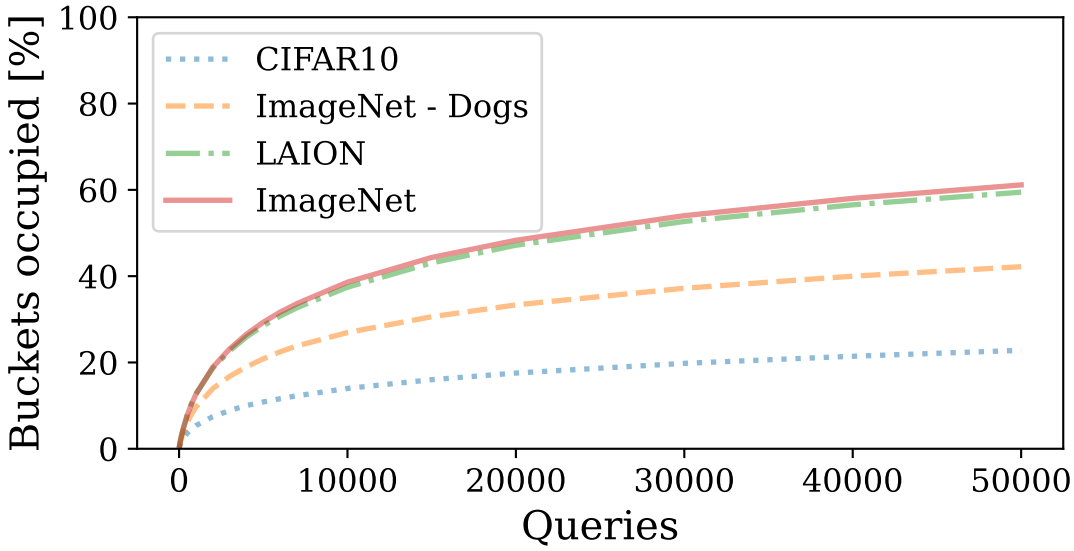


Figure A.6.2. Fraction of Occupied Buckets (Embedding Space Coverage) for the ImageNet encoder. Representations for the downstream datasets (CIFAR10, ImageNet - Dogs) occupy a smaller fraction of buckets than representations from the complex ImageNet or LAION-5B datasets. The underlying encoder is SimSiam pre-trained on ImageNet with ResNet50.

Our method of measuring the embedding space coverage is not limited to a particular encoder or dataset used for pretraining. We demonstrate this in Figure A.6.3, showing the fraction of occupied buckets for a SimCLR [54] Resnet34 encoder pretrained on CIFAR10.

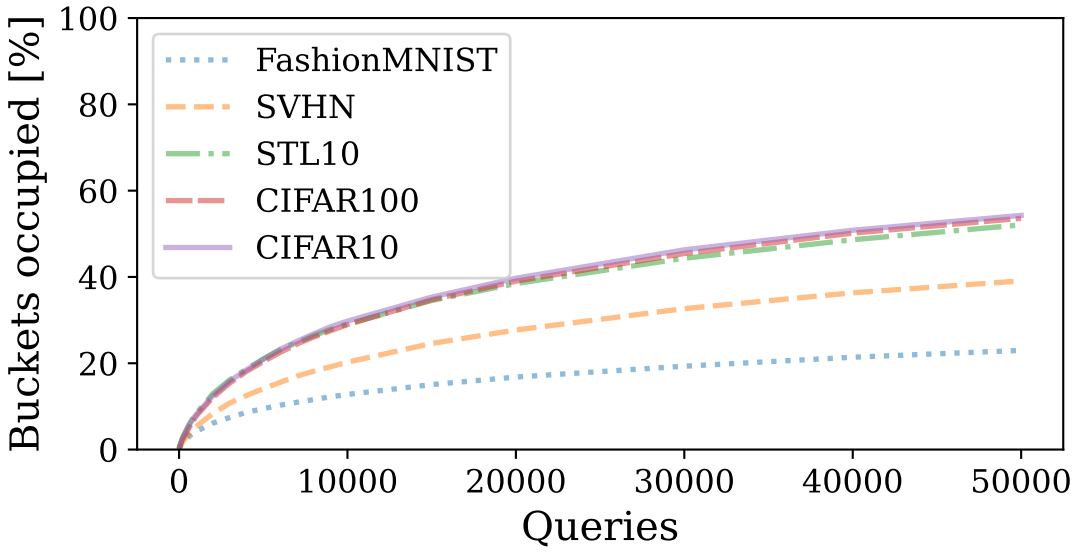


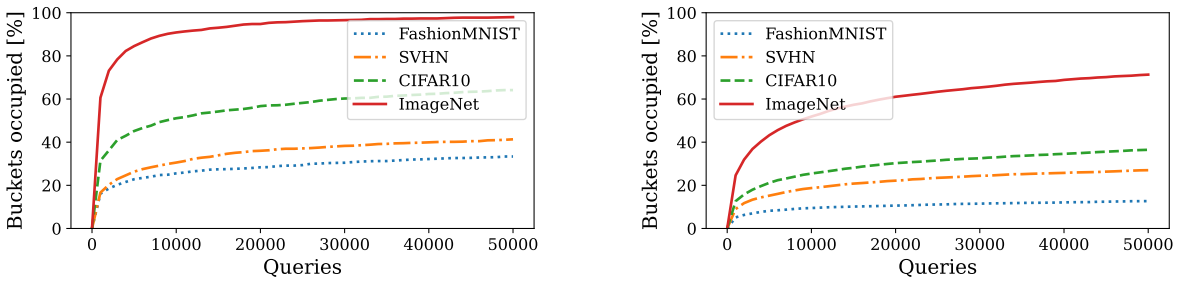
Figure A.6.3. Fraction of Occupied Buckets (Embedding Space Coverage) for the CIFAR10 encoder. B4B can be applied to an encoder trained on CIFAR10. Representations for the downstream datasets (FashionMNIST, SVHN) occupy a smaller fraction of buckets than representations from CIFAR10, CIFAR100, and STL10 datasets. The underlying encoder is SimCLR pre-trained on CIFAR10 with ResNet34.

Table A.6.5. Stealing and Using Encoders with Static Noise Addition Defenses vs. Our B4B Defense. Adding a small amount of noise results in negligible drop in performance for both legitimate user (row 2) and an adversary (row 6). Adding a large amount of noise defend stealing (row 7), but significantly harm legitimate users at the same time (row 3). Our B4B defense solves the above problem and provides high performance for legitimate users (row 4) while effectively defending the encoder against stealing attacks (row 8).

User	Defense	# Queries	Dataset	Type	CIFAR10	STL10	SVHN	F-MNIST
Legit	None	ALL	TASK	Query	90.41 $\pm$0.02	95.08$\pm$0.13	75.47$\pm$0.04	91.22$\pm$0.11
Legit	Noise σ=0.1	ALL	TASK	Query	90.20$\pm$0.03	95.15$\pm$0.13	75.29$\pm$0.09	91.24$\pm$0.02
Legit	Noise σ=10	ALL	TASK	Query	65.11$\pm$0.45	76.37$\pm$0.14	33.23$\pm$0.09	65.83$\pm$0.13
Legit	B4B	ALL	TASK	Query	90.24$\pm$0.11	95.05$\pm$0.1	74.96$\pm$0.13	91.7$\pm$0.01
Attack	None	50K	ImgNet	Steal	65.2$\pm$0.03	64.9$\pm$0.01	63.1$\pm$0.01	88.5 $\pm$0.01
Attack	Noise σ=0.1	50K	ImgNet	Steal	64.92$\pm$0.04	64.61$\pm$0.02	62.35$\pm$0.01	88.41$\pm$0.01
Attack	Noise σ=10	50K	ImgNet	Steal	36.32$\pm$0.2	32.59$\pm$0.06	20.59$\pm$0.01	74.94$\pm$0.02
Attack	B4B	50K	ImgNet	Steal	35.72$\pm$0.04	31.54$\pm$0.02	19.74$\pm$0.02	70.01$\pm$0.01

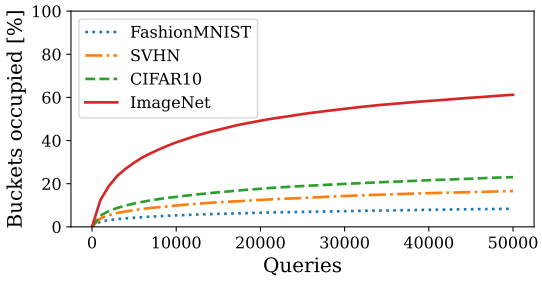
A.6.6. Setting the Number of Buckets

We present our procedure to find an optimal number of buckets in Figure A.6.4.

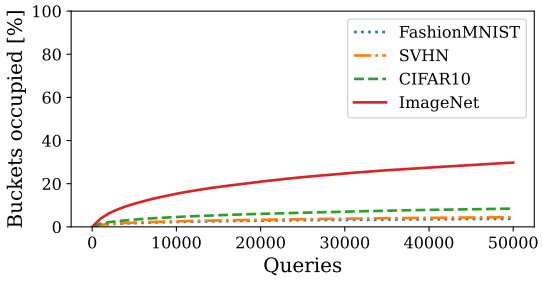


(a) Number of buckets = 2^8

(b) Number of buckets = 2^{10}



(c) Number of buckets = 2^{12}



(d) Number of buckets = 2^{14}

Figure A.6.4. Estimating Embedding Space Coverage through LSH on the SimSiam Encoder. We extend the results from Figure 6.4.1(a) and present the fraction of buckets occupied by representations of different datasets as a function of the number of queries posed to the encoder. We consider different number of buckets in the LSH table. We observe that 2^8 buckets is to small since queries from the ImageNet dataset saturate all the buckets after around 50k queries, while the number 2^{14} of buckets is too large since it is never occupied more than 40%. Thus, the number 2^{12} buckets is a good middle ground. Subfigure (c) corresponds to Figure 6.4.1 from the main paper. We also use the same notation and carry out our experiments in the same way as in Figure 6.4.1.

A.6.7. Results for DINO

We show that our defense is also applicable to the DINO encoder. The occupation of the representations space is presented visually in Figure A.6.5. We also show that the number of buckets 2^{12} is optimal for DINO in Figure A.6.6. The impact of transformation on the representations from DINO is shown Table A.6.7. Finally, the end to end experiment for DINO is presented in Table A.6.6.

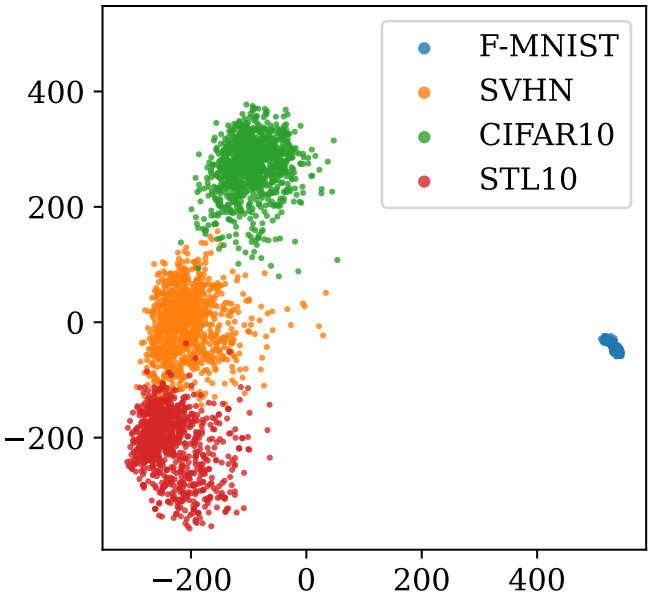


Figure A.6.5. Representations from Different Tasks Occupy Different Sub-Spaces of the Embedding Space. Presented for FashionMNIST, SVHN, CIFAR10, and STL10. In this plot, we used the DINO ViT Small encoder trained on ImageNet.

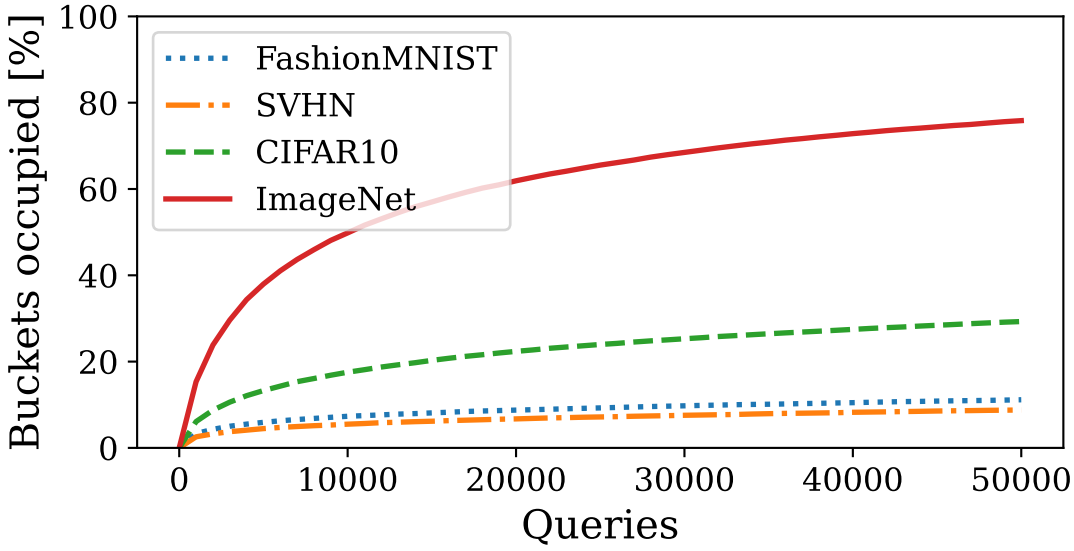


Figure A.6.6. Estimating Embedding Space Coverage through LSH on the DINO Encoder. The number of buckets is set to 2^{12}. We also use the same notation and carry out our experiments in the same way as in Figure 6.4.1.

Table A.6.6. Stealing and Using Encoders With and Without our Defense. The model used in the experiments is DINO, with the following parameters for the cost function $\lambda = 10^{-6}$, $\alpha = 1000$, and $\beta = 60\%$, and the number of buckets equal to 2^{12}. We have to increase the value of parameter α by $\times 1000$ since the norms of the DINO representations are also around 10^3 higher than for SimSiam. We observe that B4B performs similarly on DINO as for SimSiam.

USER	DEFENSE	# QUERIES	DATASET	TYPE	CIFAR10	STL10	SVHN	F-MNIST
LEGIT	NONE	ALL	TASK	QUERY	94.51 ±0.08	97.98 ±0.04	70.66 ±0.16	89.98 ±0.03
LEGIT	B4B	ALL	TASK	QUERY	94.25 ±0.11	98.05 ±0.04	69.66 ±0.14	89.68 ±0.01
ATTACK	NONE	50K	IMGNET	STEAL	67.92 ± 0.04	66.02 ± 0.22	61.30 ± 0.01	89.46 ± 0.01
ATTACK	B4B	50K	IMGNET	STEAL	42.02±0.05	38.91±0.06	19.94±0.02	73.33±0.04
ATTACK	NONE	100K	IMGNET	STEAL	75.07 ± 0.01	76.32 ± 0.02	71.79 ± 0.06	89.76 ± 0.01
ATTACK	B4B	100K	IMGNET	STEAL	19.27±0.03	21.24±0.03	19.84±0.01	71.01±0.03
SYBIL	B4B	50K+50K	IMGNET	STEAL	45.56± 0.06	42.50±0.02	24.25±0.03	78.01± 0.08

A.6.8. Additional evaluation of transformations

Additionally, we show the impact of transformations on the performance of legitimate users in Table A.6.7 (for both SimSiam and DINO).

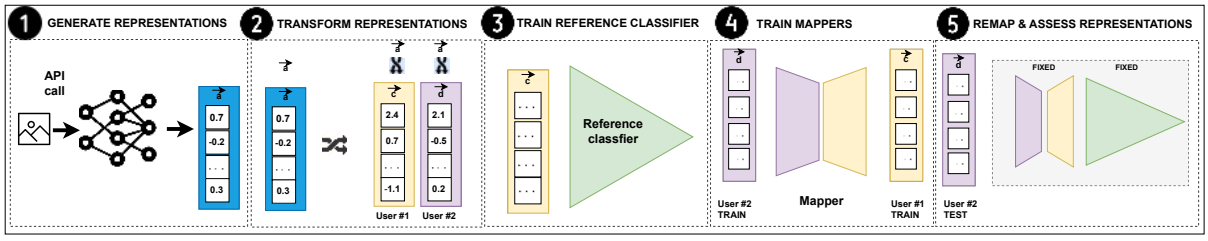


Figure A.6.7. Protocol to Evaluate the Mapping Between Representations. We present the protocol of evaluating remappings for two sybil accounts. ❶ API receives inputs from two sybil accounts and generates corresponding representations. ❷ Representations are transformed on a per-user basis and returned. ❸ Adversary trains a reference classifier on representations from account one. ❹ Adversary trains a linear model to find mapping from representations of account two to representations of account one. ❺ To check the quality of obtained mapping representations from test set of account two are mapped using the fixed mapper (from step 4) to representation space of account one. This enables the calculation of cosine distance between representations from account one and their counterparts from account two shown in Figure 6.4.3. Additionally, the fixed reference classifier (from step 3) can be used to measure the accuracy drop caused by remapping .

Table A.6.7. Impact of Transformations on the Performance for Legitimate Users. We show that the transformations applied per-account do not harm the performance of legitimate users on their downstream tasks. The victim encoders was trained on the ImageNet dataset using SimSiam and DINO frameworks.

TRANSFORMATION	ENCODER	CIFAR10	STL10	SVHN	F-MNIST
NONE	*Victim SimSiam*	90.41 ±0.02	95.08 ±0.13	75.47 ±0.04	91.22 ±0.11
AFFINE	SIMSIAM	90.24 ±0.11	95.05 ±0.1	74.96 ±0.18	91.42 ±0.15
PAD+SHUFFLE	SIMSIAM	90.4 ±0.05	95.34 ±0.06	75.47 ±0.01	91.38 ±0.15
AFFINE+PAD+SHUFFLE	SIMSIAM	90.18 ±0.06	95.03 ±0.05	74.86 ±0.1	91.35 ±0.1
BINARY	SIMSIAM	88.78 ±0.2	94.72 ±0.02	68.42 ±0.16	88.91 ±0.34
NONE	*Victim DINO*	94.51 ±0.08	97.98 ±0.04	70.66 ±0.16	89.98 ±0.03
AFFINE	DINO	94.25 ±0.11	98.05 ±0.04	69.77 ±0.11	89.68 ±0.01
PAD+SHUFFLE	DINO	94.72 ±0.02	98.07 ±0.03	70.44 ±0.1	89.91 ±0.08
AFFINE+PAD+SHUFFLE	DINO	94.26 ±0.06	98.02 ±0.01	69.49 ±0.2	89.70 ±0.1
BINARY	DINO	92.96 ±0.1	98.03 ±0.03	59.53 ±0.27	88.26 ±0.04

B. Supplement for Towards More Realistic Membership Inference Attacks on Large Diffusion Models

B.1. Broader Impact

In this work, we treat the membership inference attacks as a potential tool for privacy protection. By applying these attacks to a model, we can highlight the instances where images have been used in training without the appropriate consent. As more individuals and organizations become aware of this possibility, we might see a push towards more stringent data usage policies and ethical guidelines for machine learning practices. The potential of these attacks to expose privacy violations could serve as a catalyst for a broader dialogue on privacy rights and data ownership in the digital age.

However, these benefits come with a significant caveat. The same mechanism that can be used to protect user privacy can also be used maliciously. If a membership inference attack is successful and achieves high accuracy, it could potentially lead to personal data leakage. Thus, the dual implications of membership inference attacks for diffusion models offer both a warning and a beacon. They highlight the need for robust privacy protection measures while simultaneously alerting us to the potential risks of personal data leakage.

B.2. Limitations

LAION-5B dataset, the source of data used both to train the Stable Diffusion model and to create LAION-mi dataset does not contain the images, just URLs to them. Therefore, to prepare a training set, one needs to first download the images using the URLs, and then train the model. Since the content, that a URL points to is out of our control, it might be a subject of edition or deletion, which we are unable to handle. Because of that, we cannot guarantee with 100% certainty that every member in LAION-mi is a real member of the Stable Diffusion model's train set.

In fact, during our experiments, we find out that approximately 10% of links are dead, i.e. we cannot download images from them. This limitation is inherent to the LAION-5B dataset, and therefore cannot be alleviated. Additionally, a URL to an image from LAION-mi has to be alive to use it for the attack evaluation. We argue that it is unlikely that a URL from the LAION-mi members subset that was dead during training of the Stable Diffusion is now alive, however, we cannot rule out such scenario. Fortunately, since it affects only the members set, it makes the membership inference task harder, in effect saving us from the pitfalls described in Section 7.4.

B.3. Stable Diffusion-v1.4 Training

Datasets Datasets used to train this model are as follows

- LAION-2B EN: a subset of the LAION-5B[203] with images' descriptions in English.
- LAION-Aesthetics V2 5+[202]: a subset of the LAION-2B EN dataset, in which each image is scored using the LAION-Aesthetics_Predictor V2[201], and only images with the Aesthetic Score above 5 are a part of it.

Stable Diffusion-v1.4 This version of the Stable Diffusion is first trained for 431k steps using samples from LAION-2B EN, then fine-tuned for 515k steps using LAION-Aesthetics V2 5+, and then fine-tuned for 225k steps on LAION-Aesthetics V2 5+ again [190].

B.4. Loss Attacks

In this section, we discuss different approaches to perform the diffusion denoising process in the white-box scenario. In each of the following setups, at every timestep, we collect the Model loss, Latent error, and Pixel error described in Section 7.6.2. These values are calculated based on the output of the UNet SD-v1.4 backbone using the methods described below.

B.4.1. All Attack Methods

Here we describe all the methods used in our experiments. For all methods, the classifier-free guidance is applied with scale 7.5 unless otherwise stated.

1. **Partial denoising** Following findings in [48] in this method, we start our denoising process at the timestep 300 for the steps 26 up to the timestep 50. Latent image representation is noised once, with scale α_{300}, and then for each step calculated from the UNet noise prediction.
2. **Partial denoising non-iterative** is similar to the *Partial denoising* method, but at each timestep we pass a newly noised latent representation of the image to the UNet, instead of the output of the previous timestep's inference through UNet. The main intuition behind this is to see whether the information about membership gets lost or accumulates during the iterative denoising process.
3. **Partial denoising latent shift** follows the *Partial denoising* attack method, but at the middle timestep 170 we shift the latent representation of the image by a scaled random noise. The intuition behind this method is that the member samples should return to the correct trajectory after the shift better than the nonmember samples enabling us to better identify the member samples.
4. **Partial denoising text shift** is similar to the *Partial denoising* latent shift, but instead of shifting the latent representation of the image, we shift the text embedding by a random noise $N(0, I) \times 0.1$ before passing it to the UNet. The intuition here is the same as in the partial denoising latent shift method, but we want to see if the text embedding is more important than the latent representation of the image.
5. **Partial denoising bigger text shift** is identical to the *Partial denoising text shift*, but the noise added to the text embedding is scaled by 0.5 instead of 0.1. We want to test if adding more noise to the text embedding benefits the performance of the attacks.
6. **Partial denoising wrong start** In this method we follow the *Partial denoising*, but the starting latent representation of the image is noised using α_t from the wrong timestep, $t = 100$ instead of $t = 300$. We want to test whether we can extract more information about the membership if we apply lower noise to the latent representations at the beginning of the process than UNet expects.
7. **Full denoising start 100** is similar to the *Partial denoising wrong start* method, but we perform an almost full denoising process, starting on timestep and 900 ending at timestep 0, denoising every 100 and starting from the latent noised with α_{100} noise scale.
8. **Full denoising start 100 text shift** is close to the *Full denoising start 100*, but we shift the text embedding by a random noise $N(0, \mathbf{I}) * 0.1$ before passing it to the UNet.
9. **Full denoising start 50** is similar to the Full denoising start 100, but the starting latent is noised using α_{50} instead of α_{100}.

10. **Full denoising start 300** is like the Full denoising start 100, but the starting latent is noised using α_{300} instead of α_{100}.
11. **Short denoising start 300** In this method, we perform a short denoising process, starting from timestep 200 ending at timestep 0 every 100 steps instead of the full one. The latent representation of the image is noised using α_{300}.
12. **Reversed noising** We perform 10 denoising steps for timesteps from 900 to 0. The image latent representation we get from the VAE encoder is noised using noise scales α_t in reverse order, e.g. at timestep 800 we pass the latent noised with α_{100} to the UNet. Additionally, during inference, we use classifier-free guidance on the guidance scale 100. The text embedding passed to UNet remains unchanged. The intuition behind the attack is that member samples will behave more robustly than nonmember samples under such conditions.
13. **Full denoising start 300 no cfg** is similar to the *Full denoising start 300*, but we do not use the classifier-free guidance. We want to see if the classifier-free guidance is beneficial for the attacks performance.
14. **Full denoising start 100 non-iterative**. This method is identical with the *Full denoising start 100*, but at each timestep we pass newly noised latent representation with scale α_{100} of the image to the UNet, instead of the output of the previous timestep.
15. **Reversed noising regular cfg** resembles the *Full denoising start 100 non-iterative* attack method. The latent representations we pass to the UNet are noised using α_t from the timesteps in the reversed order, i.e. from 0 to 900, so the first latent passed to the UNet is noised using α_0, while UNet receives timestep 900 as input. The classifier-free guidance is applied with scale 7.5.
16. **Reversed denoising** In this method, we reverse the order of timesteps at which we perform denoising using UNet. We start from the timestep 0 and then go up to the timestep 900, for 10 steps in total. Similarly as in the baseline loss threshold method, we apply noise to the latent image representation once, but this time using noise scale α_{100}. At each timestep we measure all losses described in Section 7.6.2 and use them to evaluate the attack. The intuition here is similar as in the *Reversed noising* method, but here we test if the iterative nature of the diffusion denoising process will magnify this effect.

B.4.2. Classifier Attack

In addition to the *threshold* attack type described in 7.6.2 we also introduce the *classifier* attack type. *Classifier* attack builds on top of the *threshold* attack. It uses a binary classifier C to predict the membership of an image x based on the losses collected during inference through the diffusion model. The classifier is trained on a

set of images with known membership and returns a probability of membership for a given image. We then perform a *threshold* attack on the classifier's output.

We train four model classes: Logistic Regression (LR), Decision Tree Classifier (DTC), Random Forest Classifier (RFC), and Neural Network binary classifier (NN). The classifiers are trained in the binary classification task, with the member samples as positive examples and the nonmember samples as negative examples. The input for every classifier consists of the vector of the loss values we collect at every timestep. For the Logistic Regression (LR), Decision Tree Classifier (DTC) and Random Forest Classifier (RFC) we perform Grid Search with k-Cross Validation, $k = 5$ for each fit. The hyperparameters used for the classifiers are described in Table B.4.1. Note that we perform the Grid Search for every fit separately, therefore we do not report the best parameters in our table, since they turn out to be different for different fits.

For the Neural Network classifier (NN) we use the following architecture: 3 fully connected layers, with the input size of $3 * timesteps$ (since for different methods we have a different amount of data), hidden size of 10 and a single output for the binary classification. We use ReLU activation function for the hidden layers and Sigmoid for the output layer. We use Adam optimizer with a learning rate 0.001 and binary cross entropy loss. We train the classifier for 100 epochs with batch size 32 and early stopping. We use the best model from the early stopping for the evaluation.

Following 7.6.4 we fit our models 100 times on 100 different training sets sampled from the whole attack set and evaluate them on the remaining evaluation sets. We report the mean and standard deviation of the metrics for the 100 fits. The results are presented in Table B.4.2.

Table B.4.1. Hyperparameters used for the classifiers: Logistic Regression (LR), Decision Tree Classifier (DTC) and Random Forest Classifier (RFC).

Classifier	Hyperparameter	Values
LR	C	[0.01, 0.1, 1, 10, 100]
DTC	max_depth	[2, 8, 16]
	min_samples_split	[2, 8, 16]
RFC	n_estimators	[10, 100, 1000]
	max_depth	[2, 8, 16]

B.4.3. Results

Following our evaluation method described in Section 7.6.4 we report our results for each method in Table B.4.2 for *threshold* attacks, and in Table B.4.3 for *classifier* attacks.

Table B.4.2. *Threshold* attack results for each method. We see that for each attack method Model loss gives out better performance of *threshold* attacks compared to Latent or Pixel error. It is also more robust for the denoising procedure modification than Latent Error, dropping to at most 2%, while Latent error performance can drop to 1.1%. Pixel error seems also more stable than Latent Error, but we cannot achieve as high results as for the Model loss or Latent error. The high discrepancy between different attack methods points that we can indeed influence the amount of membership information extracted during influence, with some methods performing better on Model loss and some on other losses, Pixel and Latent error.

LOSS / METHOD	MODEL LOSS	LATENT ERROR	PIXEL ERROR
PARTIAL DENOISING	2.3%±0.61	**2.4**%±0.62	1.7%±0.68
PARTIAL DENOISING NON-ITERATIVE	2.3%±0.68	1.1%±0.46	1.39%±0.59
PARTIAL DENOISING LATENT SHIFT	2.3%±0.62	2.24%±0.9	1.6%±0.62
PARTIAL DENOISING TEXT SHIFT	2.2%±0.57	2.28%±0.57	1.7%±0.64
PARTIAL DENOISING BIGGER TEXT SHIFT	2.3%±0.62	2.22%±0.75	1.74%±0.6
PARTIAL DENOISING WRONG START	2.2%±0.69	2.13%±0.85	1.99%±0.56
FULL DENOISING START 100	2.08%±0.64	1.1%±0.41	1.84%±0.6
FULL DENOISING START 100 TEXT SHIFT	2.01%±0.6	1.1%±0.43	1.8%±0.6
FULL DENOISING START 50	2.0%±0.6	1.15%±0.38	1.95%±0.55
FULL DENOISING START 300	1.99%±0.61	1.34%±0.49	1.72%±0.64
SHORT DENOISING START 300	2.32%±0.67	2.06%±0.72	1.57%±0.67
REVERSED DENOISING	2.25%±0.64	2.17%±0.64	**2.03**%±0.55
FULL DENOISING 300 NO CFG	2.07%±0.62	1.45%±0.52	1.7%±0.6
FULL DENOISING 100 NON-ITERATIVE	2.2%±0.55	2.18%±0.75	1.91%±0.53
REVERSED NOISING REGULAR CFG	**2.5**%±0.74	2.03%±0.6	1.92%±0.5
REVERSED NOISING	**2.51**%±0.73	1.26%±0.52	1.9%±0.51

We conclude that extraction of the membership information from the attacked model can be improved by modifying the diffusion denoising process by altering the timesteps, latent representations, and text embeddings. We also see that the information about membership can be better extracted by using *classifier* attack, but for most methods, a simple *threshold* attack outperforms the *classifier* attack.

We also observe that different methods achieve visibly different performance on different timesteps, which points out that the timing of loss measurement is also really important when performing the *threshold* attack, see Figure B.4.1. We see the importance of to collecting all three losses on each timestep, with the biggest difference visible for the *Reversed noising* method, with the worst result around 0.5% and the best around 2.5% for Latent error at timestep 0 and Model loss at timestep 500. We derive the following insights from these plots. Applying the denoising process iteratively instead of passing the newly noised latent to the UNet greatly benefits the Latent and Pixel error *threshold* attacks, while slightly reducing the effectiveness of *threshold* attacks on the Model loss. Shifting the latent by a random noise mid-inference affects the results negatively, with the biggest drop

Table B.4.3. *Classifier attack* results for each method. Same as in Tab. B.4.2 we observe that for different attack methods the same model classes achieve visibly different performance. This again confirms that we are able to obtain more information about the membership of samples by influencing the whole denoising process of the large diffusion model. Unsurprisingly, we are able to outperform the simple *threshold* attacks, because the membership information ends up spread out on different timesteps and losses. Using *classifier* attack we can combine the information from the whole inference procedure and make better predictions. Surprisingly, it seems to be harder to perform such attacks in the TPR@FPR=1% regime. For almost all methods and model classes we are not able to outperform the simple *threshold* attacks, especially on the Model loss. For almost all methods the Random Forest Classifier model outperforms all other model types, with the exception of *Reversed noising regular cfg* method data fitted using Logistic Regression. Decision Tree Classifier fails to be better than a random guess for almost all methods, and the Neural Network classifier also fails to deliver satisfying results.

Classifier class method	LR	DTC	RFC	NN
Partial denoising	1.83%±0.87	0.67%±1.09	2.27%±0.76	1.50%±0.92
Partial denoising non-iterative	2.24%±0.86	**1.10**%±1.12	2.35%±0.88	1.52%±1.03
Partial denoising latent shift	1.94%±0.92	0.86%±0.94	2.35%±0.86	1.40%±0.91
Partial denoising text shift	1.71%±0.87	0.49%±0.84	2.43%±0.93	1.40%±1.00
Partial denoising bigger text shift	2.14%±1.01	0.80%±1.13	2.30%±0.73	1.49%±0.90
Partial denoising wrong start	1.50%±0.70	0.74%±1.08	2.26%±0.87	1.51%±0.92
Full denoising start 100	1.09%±0.56	0.54%±0.74	1.89%±0.85	1.30%±0.76
Full denoising start 100 text shift	1.10%±0.62	0.62%±0.76	1.88%±0.80	1.24%±0.69
Full denoising start 50	1.16%±0.68	0.70%±0.68	1.89%±0.78	1.31%±0.78
Full denoising start 300	1.21%±0.58	0.42%±0.85	1.87%±0.87	1.39%±0.80
Short denoising start 300	1.94%±0.91	0.54%±0.83	2.31%±0.91	1.49%±0.85
Reversed denoising	1.96%±0.96	0.80%±1.17	2.37%±0.98	1.26%±0.78
Full denoising 300 no cfg	1.31%±0.73	0.31%±0.65	1.78%±0.78	1.37%±0.82
Full denoising 100 non-iterative	1.60%±0.77	0.68%±1.05	2.19%±0.85	1.60%±0.73
Reversed noising regular cfg	**2.41**%±1.09	0.47%±1.03	2.17%±0.84	1.40%±0.82
Reversed noising	1.98%±0.97	0.41%±1.02	**2.75**%±1.03	**1.62**%±0.86

in performance for the Model loss. Shifting text embedding by a random noise for different noise scales (0.1 and 0.5) does not make any significant difference in terms of the final results, generally being slightly harmful to the performance of the *threshold* attack. The mismatch between the actual and latent noise timesteps is by far the most impactful modification we implemented. We see improvements on the Pixel error *threshold* attacks for *Partial denoising wrong start* method, where we start from the latent noised with the noise scale of $\alpha_t = 100$ and denoising from the timestep 300 to 50. However, for the *Full denoising* attack methods we see that different noise scales applied to the starting latent representations do not change the performance. Applying the classifier-free guidance seems to be insignificant for the final results (see *Full denoising 300* vs *Full denoising 300 no cfg*), but increasing the guidance scale (from the default 7.5 to 100) reduces the effectiveness of the *threshold* attacks on Pixel and Latent error. The best results are obtained when we reverse

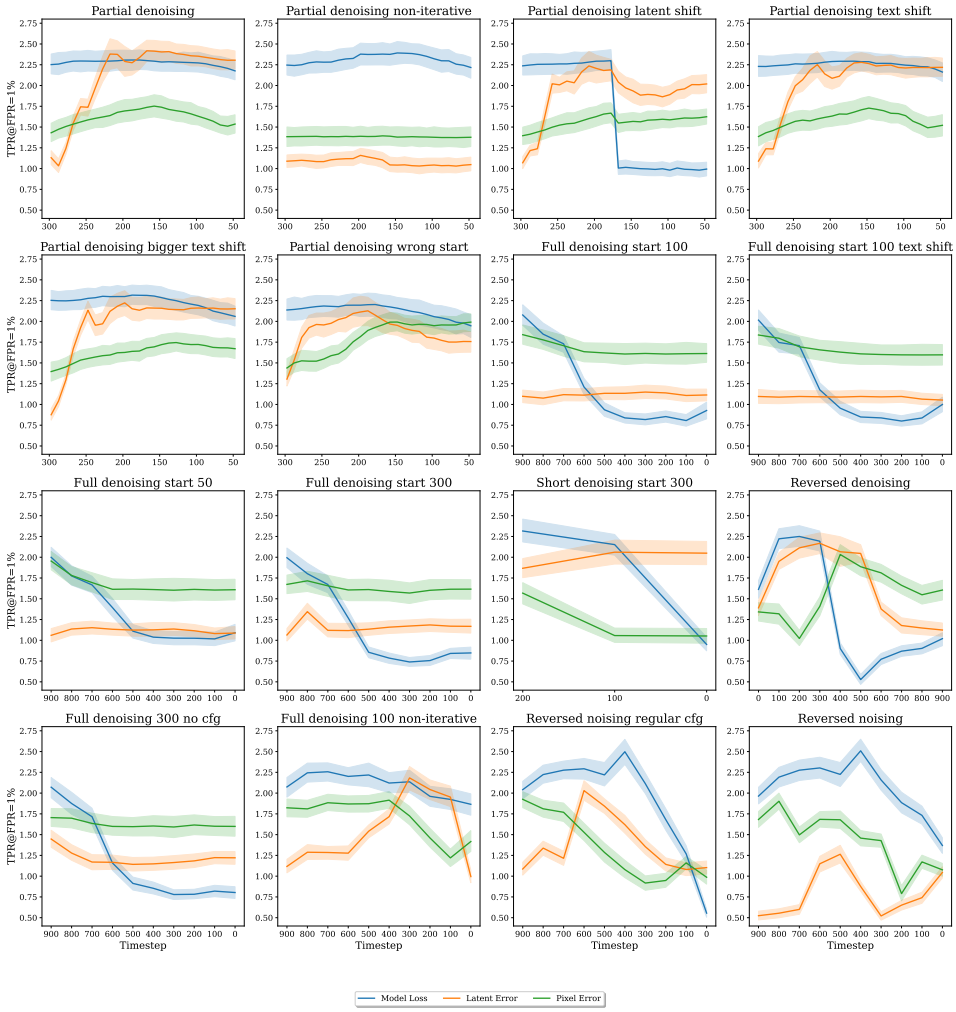


Figure B.4.1. Attack performances on different losses in different timesteps for each attack method for the *threshold* attack type. The solid line indicates the mean value of the TPR@FPR=1% metric from 100 experiments, and the shaded area is the 95% confidence interval.

the order of noise scales used to apply noising on the latent representation of the input image, while the input timesteps we pass to UNet remain in the normal order. We also see in Tab. B.4.3 that these methods allow the *classifier* attacks to achieve the best results, suggesting that this approach extracts the most data about the membership from the model.

B.5. Experiments Randomization

Here we highlight the need to perform the randomization described in Section 7.6.4. For each of the methods described in Appendix B.4 we show the differences between the best, mean and worst results for each attack from the separate runs. We also visualise these differences for the classification attack type. We can observe

a mismatch between mean and best results for all of the attacks, some of them being even three times worse than the best ones (*Partial denoising* method). When the available size of the nonmembers set is relatively small, the randomization is crucial to obtain reliable results, especially when proposing attacks based on more sophisticated methods that require training a classifier, which is the case in the *classifier* attack type. In this case, we suggest splitting the whole attack set randomly in the way described in Section 7.6.4 into training and evaluation sets, then training the classifier, and repeating the whole procedure for at least 100 times. The reported results should be the mean of the results for each attack from each run. In this way, we can mitigate the influence of the potential outliers and obtain more reliable results.

Visualisation of the mismatch between the best, mean, and worst results can be found in Figure B.5.1 for the *threshold* attacks and Figure B.5.2 for the *classifier* attacks.

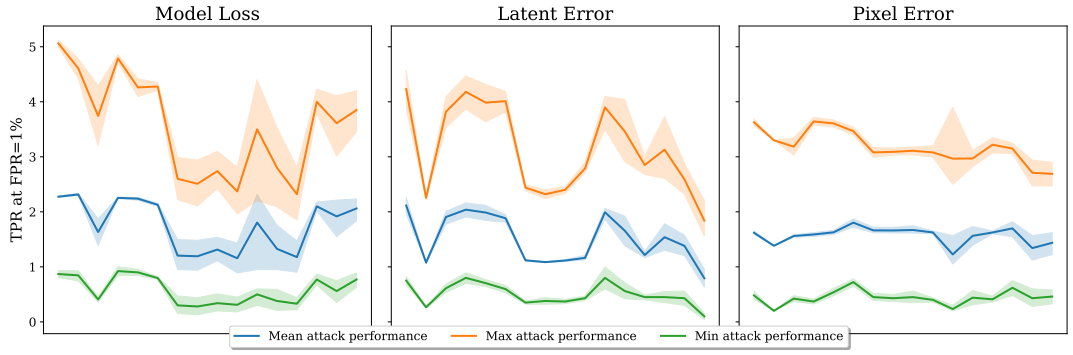


Figure B.5.1. Min, mean, max TPR@FPR=1% for different methods and losses, *threshold* attack type. Solid line indicates the mean value of the TPR@FPR=1% metric from 100 experiments, and the shaded area is the 95% confidence interval.

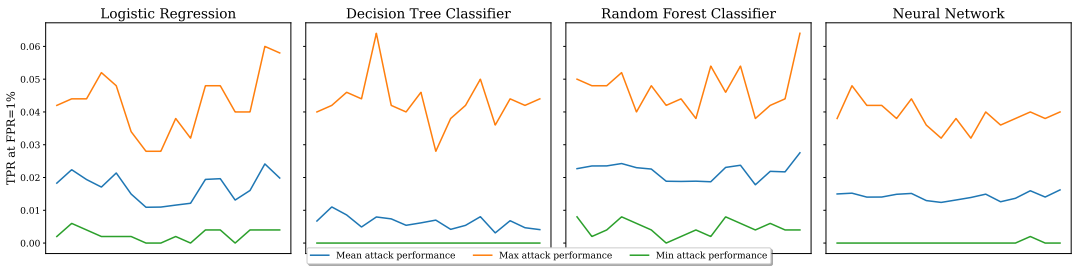
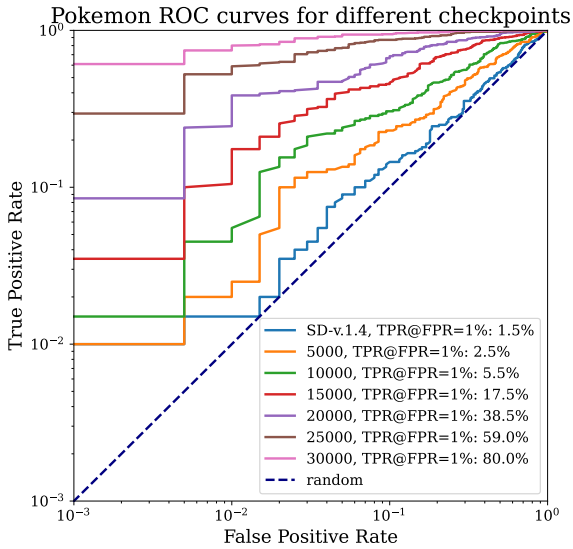


Figure B.5.2. Min, mean, max TPR@FPR=1% for different methods, classifier classes and losses, *classifier attack* type.

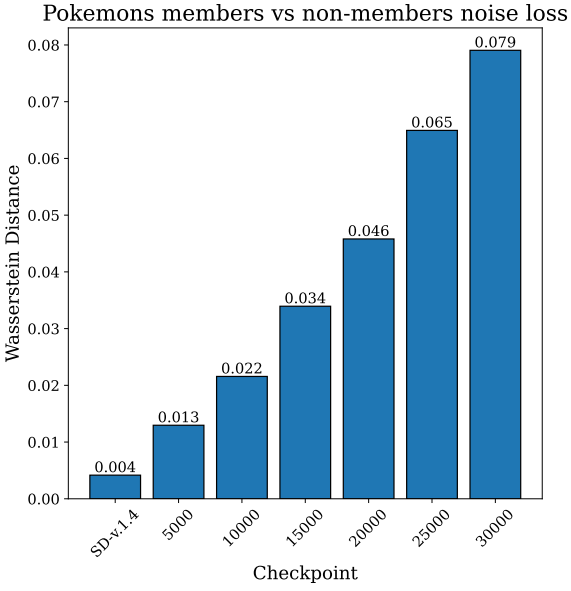
B.6. Overfitting Impact on Membership Inference Attacks

In this section, we present the results of our experiments on overfitting on the POKEMON dataset from Section 7.6.4. We finetune the original StableDiffusion-v1.4 using the default method from [123] for 30000 train steps with the learning rate 0.00001 and gradient accumulation of 4. Every 5000 steps we save the partially finetuned model.

The attack method we use to perform for this section is the Baseline loss threshold method, described in Section 7.6.3. The ROC curves and TPR@FPR=1% results for different checkpoints can be found in Figure B.6.1a. We can observe that the model is overfitting to the training set, as the TPR@FPR=1% results are significantly better for the checkpoints from the later stages of the training. This is in line with the findings in [47], which also show that model loss based membership inference attacks give better results when the model is overfitted. This issue seems to be not addressed in some of the previous contributions, e.g. in [79] authors claim really good performance of their attacks on the finetuned version of the SD-v1.4 model on the same dataset.



(a) ROC curves and TPR@FPR=1% results for different checkpoints.



(b) Wasserstein Distance for different checkpoints.

Figure B.6.1. Figure B.6.1a shows how overfitting correlates with the attacks performance. Figure B.6.1b shows the Wasserstein Distance calculated between the losses of members and nonmembers sets for each checkpoint.

B.7. Fine-tuned Shadow Models for Large Diffusion Models

Training a new Stable Diffusion model from scratch multiple times is in practice infeasible. Thus, it is impossible to directly apply the existing shadow model attack in this case. To overcome this limitation we focus our analysis on model fine-tuning, rather than training from the start. Motivated by the state-of-the-art results achieved by shadow model membership inference, we draw inspiration for our approach from the offline *LIRA* attack introduced in [47]. Intuitively, we aim to answer the question "*is there a difference between finetuning the model on member and nonmember samples?*". To overcome the memory requirements for storing multiple versions of the Stable Diffusion models we apply fine-tuning with *LoRA* [121].

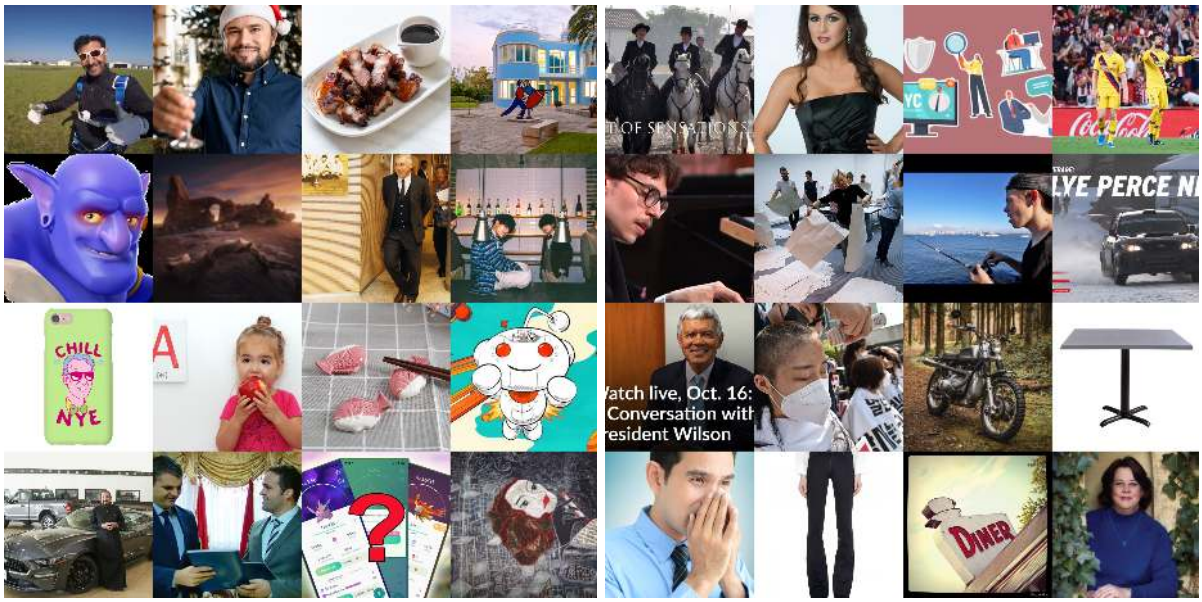
First, we sample n nonmember samples from LAION-mi. For each single sample, we finetune the Stable Diffusion v1.4 model on a single training step at $t = 100$ using *LoRA*. We measure the ratio of the training loss after and before the finetuning. We repeat that procedure for n member samples from LAION-mi. Thus we obtain loss ratio distributions for shadow models finetuned on member (D_{mem}) and nonmember (D_{nonmem}) samples. For inference, we fine-tune the model on the new sample using the same procedure, obtaining the corresponding loss ratio $l\star$. If $\frac{Pr[l\star|D_{mem}]}{Pr[l\star|Dnonmmem]} > \tau$ we classify the sample as a member.

This attack can only be performed in the white-box scenario. The attacker needs to have access to the model weights to perform the fine-tuning.

We evaluate our method on 100 subsets of 1000 member and 1000 nonmember samples randomly selected from 4000 samples from each set. The approach based on shadow models achieves TPR@FPR=1% equal to 2.21% ± 1.11. The performance of the attack is thus comparable with the attacks described in Section 7.6.3. Although shifting the focus of shadow models from full training to finetuning using *LORA* enabled us to apply this kind of approach, the resources required to execute this attack are still significantly larger than for the methods described in Sec. 7.6.3.

B.8. LAION-mi Samples

In this section we provide a random sample of the LAION-mi dataset, 16 images from the nonmembers set and 16 images from the members set, see Figure B.8.1.



(a) LAION-mi nonmember samples. **(b)** LAION-mi member samples.

Figure B.8.1. Visualization of the random subset of LAION-mi dataset

B.9. GPU Cost

To conduct the experiments for this paper we utilized 4000 hours of Nvidia A100 GPU compute.

C. Supplement for CDI: Copyrighted Data Identification in Diffusion Models

C.1. Broader Impact

Our research addresses the pressing issue of verifying the ownership of data used for training large image generation models, as highlighted by recent legal disputes [28]. By introducing `CDI`, we aim to enable data owners to verify if their data was (illegitimately) used for training DMs. Our work contributes to a more transparent and accountable ML ecosystem, aligning with broader societal values of fairness and respect for data ownership. We anticipate `CDI` will have a positive impact on both the ML community and society, promoting responsible and fair development of ML models. We make our code available at `https://github.com/sprintml/copyrighted_data_identification`.

C.2. Limitations

In this paper, we assess the effectiveness of `CDI` for image generation DMs. Although we believe that our methodology extends well to other modalities such as text or video, we do not perform an experimental evaluation in these areas. Our research focuses on diffusion models as the current state-of-the-art image generators widely employed in commercial applications. While acknowledging the existence of alternative generative image models, which have recently been shown to demonstrate comparable capabilities [130, 199], we choose to focus on DMs within the scope of our research. We note that CDI requires at least gray-box access, i.e. DM's predictions at an arbitrary timestep t to be effective. This limitation stems from the lack of reliable, strictly black-box MIA methods for DMs, *i.e.,* methods that leverage only the final generated image, while avoiding the pitfalls described in App. C.4.

C.3. Additional Background

C.3.1. Previous Membership Inference Attacks against DMs

We present the previous MIAs against DM that provide the initial set of features we use for our `CDI`.

Denoising Loss by Carlini et al. [48]. The common intuition for utilizing the loss function of the model is that its values should be lower for the training set (members) than validation or test set (non-members). Formally, we follow LiRA [47], and its extension to DMs [19], and for each sample, we compute $\|\epsilon - f_\theta(z_t,t)\|_2^2$ at $t = 100$. Note, that z_t is obtained by adding $\epsilon \sim \mathcal{N}(0,I)$ to the original z, a process which, by its stochasticity, introduces noise to obtained scores. car [19] suggest to address this issue by computing the loss for five z_t noised using different ϵ. The final feature is the mean of these five measurements.

SecMI$_{\text{stat}}$ by Duan et al. [79]. The feature is based on the assumption that the effect of the denoising process should restore member samples better than non-member samples. Duan et al. [79] formalizes this idea by introducing *t-error*, a metric that aims to approximate the estimation error of f_θ on a given z. More specifically, *t-error* is defined as $\|\psi_\theta(\phi_\theta(\tilde{z}_t,t),t) - \Phi_\theta(z_0,t)\|_2^2$, where $\psi_\theta(z_t,t) = z_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\hat{f}_\theta(z_t,t) + \sqrt{1-\bar{\alpha}_{t-1}}f_\theta(z_t)$ is the DDIM [215] denoising step, $\phi_\theta(z_t,t) = z_{t+1} = \sqrt{\bar{\alpha}_{t+1}}\hat{f}_\theta(z_t,t) + \sqrt{1-\bar{\alpha}_{t+1}}f_\theta(z_t,t)$, is the DDIM sampling inverse step, $\hat{f}_\theta(z_t,t) = \frac{z_t - \sqrt{1-\bar{\alpha}_t}f_\theta(z_t,t)}{\sqrt{\bar{\alpha}_t}}$, $\bar{\alpha}_t = \prod_{k=0}^{t}\alpha_k$, and $\Phi_\theta(z_s,t) = \phi_\theta(\cdots\phi_\theta(\phi_\theta(z_s,s),s+1),\cdots,t-1)$ is the deterministic reverse. *t-error*, intuitively, should hold lower values for member samples.

PIA by Kong et al. [137]. PIA builds on the notion that, under DDIM sampling settings, given z and any z_t, one can determine the *ground truth trajectory* consisting of intermediate z_s, $s \in (0,t)$. Subsequently, f_θ learns to reflect this trajectory and is more competent in it for members. Capturing that difference can act as a membership signal, defined as $\|f_\theta(z,0) - f_\theta(\sqrt{\bar{\alpha}_t}z + \sqrt{1-\bar{\alpha}_t}f_\theta(z,0),t)\|_5$, where $\bar{\alpha}_t = \prod_{k=0}^{t}\alpha_k$, and $t = 200$. The feature should be lower for members.

PIAN by Kong et al. [137]. An important consideration in PIA is that $f_\theta(z,0)$ follows a Gaussian, *i.e.,* $f_\theta(z,0) \sim \mathcal{N}(0,I)$, as it should make the attack more performant. To assure that, Kong et al. [137] propose PIAN (normalized PIA), as an extension of their method. Formally, $\hat{f}_\theta(z,0) = C\cdot H\cdot W\sqrt{\frac{\pi}{2}}\frac{f_\theta(z,0)}{\|f_\theta(z,0)\|_1}$ with C,H,W denoting the channels, height, and width of the input sample, respectively. However, our experiments

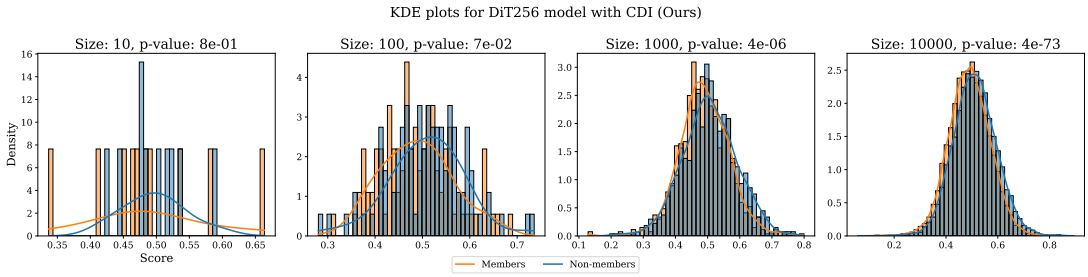


Figure C.3.1. KDE plots for varying $|\mathbf{P}_{\text{test}}|$. We use the DIT256 model.

in App. C.21 and section 8.5.1 indicate that this intuition does not translate to latent DMs, which is in line with findings from the original paper [137].

C.4. On the Mismatch in MIAs Results

Contrary to the promising results of $\text{SecMI}_{\text{stat}}$, PIA, and PIAN, our evaluation in App. C.21 shows that these attacks fail to reach performance significantly higher than random guessing. The following is a methodological analysis of issues in the experimental setup proposed in [79, 101, 137]. We identify three pitfalls in their experimental settings: (1) usage of toy models, (2) overfitting to the evaluation set, and (3) distribution mismatch between members and non-members sets.

C.4.1. Pitfall 1: Toy Models

MIAs performance is directly correlated with the level of overfitting in the attacked model [195, 211]. Unfortunately, it is a common approach to evaluate the MIAs against DMs on very small toy models, trained or fine-tuned on small-scale datasets like CIFAR10 [141], as it speeds up the inference. This in turn can lead to elevated performance which is further reported in published works [79, 101, 137], *e.g.,* the **TPR@FPR=1%** at the level of 10% for $\text{SecMI}_{\text{stat}}$ and 30% for PIAN. We argue that evaluating any type of MIA in such setting is flawed, and provides incorrect insight on the performance of the proposed MIA.

In contrast to previous works, we focus only on evaluating success of `CDI` on non-overfitted large DMs from official repositories, trained on high-scale datasets like ImageNet.

C.4.2. Pitfall 2: Overfitting to the Evaluation Set

$\text{SecMI}_{\text{NNs}}$ [79] in its official implementation uses the evaluation set for selection of the best-performing classifier[1].

[1] `https://github.com/jinhaoduan/SecMI/blob/main/mia_evals/secmia.py#L358`

C.4.3. Pitfall 3: Mismatch in Distribution of Member and Non-Member Sets

dub [33] highlight the importance of members and non-members sets being indistinguishable from each other in order for the results of MIAs being reliable. Indeed, the mismatch between these can be enough for any classification-based method to succeed, without the context of attacked model. More importantly, a simple Loss attack [43] also benefits from this pitfall, as out-of-distribution (incorrect non-members) samples usually achieve significantly higher values of model loss objective than in-distribution samples (members, or correct non-members, *e.g.,* test set). Unfortunately, Duan et al. [79] for evaluation of their SecMI, as well as Kong et al. [137] for PIA on SOTA Stable Diffusion [24] utilize an external dataset, namely COCO [231], as their non-members set. In effect, it casts doubt on the reported success of their methods.

In our work we avoid this problem by using validation sets of the DMs as the source of non-members samples.

C.5. On Reporting p-Values

To ensure the reliability of the statistical test results reported for `CDI`, we adopt the following approach. We repeat the t-tests 1000 times on features obtained from randomly sampled subsets of $\mathbf{P}$ and $\mathbf{U}$, aggregating the resulting p-values. We consider various p-value aggregation methods as reviewed by [139, 234], and make our final decision based on the specific context in which we apply `CDI`.

In our framework, we assume that `CDI` is executed by an arbitrator approached by a victim whose private data might have been used in the training of a DM. Each execution of the t-test represents data verification for a *single* data owner, with each p-value corresponding to the test outcome for an *individual case*. To appropriately represent the performance of our method in this setting, we report p-values aggregated using an arithmetic mean over the success for all data owners.

This aggregation method is a conservative approach for reporting our results (comparing to *e.g.,* harmonic mean). This is due to the vulnerability of the arithmetic mean to outliers as the p-values are strictly positive, and the mean can be saturated easily by a single large value. In contrast, harmonic mean can be too easily brought down to almost zero by a single good result which would overstate the success of our method.

C.6. Impact of the Size of the $\mathbf{P}_{\text{test}}$ on CDI Confidence

We present a visualization of the impact of the size of the $\mathbf{P}_{\text{test}}$ on the confidence of CDI in fig. C.3.1. We sample $n = 10, 100, 1000, 10000$ samples randomly from $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$, and apply CDI. We observe that when we increase the size of $\mathbf{P}_{\text{test}}$ the method becomes more stable and more confident. This is an inherent feature of statistical testing, a core element of CDI.

C.7. Additional Features

As noted previously, Latent Diffusion Models represent the state-of-the-art in high-resolution image generation for DMs. Such models are two-stage architectures, where the diffusion process occurs in the latent space of an autoencoder. This observation introduces another potential angle for MIA on latent DMs. We note that CDI can be extended by incorporating signal from features specifically tailored for the autoencoder part of the model. However, it is important to recognize that the diffusion backbone and the autoencoder are separate models, which can be trained on different datasets. This necessitates caution when deciding whether to incorporate features extracted from the encoder. Nonetheless, in cases where both the autoencoder and the DM are trained on the same dataset, we claim that the performance of CDI can be further boosted by incorporating membership signals from the autoencoder. To this end, we propose an additional feature that can be utilized in such scenarios.

Autoencoder Reconstruction Loss (ARL). The goal of this feature is to extract differences in the autoencoder reconstruction errors between members and non-members, where members should exhibit significantly lower errors. To obtain the features, we first note that the current state-of-the-art DMs consist of the pixel and latent spaces [24]. For our ARL feature, we leverage the two-stage structure of DMs and extract the membership signal directly in the pixel space, which contains an autoencoder with the encoder part $\mathcal{E}$ and decoder $\mathcal{D}$. Given an input image x, the encoder $\mathcal{E}$ encodes x into a latent representation $z = \mathcal{E}(x)$. The decoder $\mathcal{D}$ reconstruct the image from the latent z, yielding $\tilde{x} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x))$. The autoencoder reconstruction loss serves as the ARL feature computed as $||x - \mathcal{D}(\mathcal{E}(x))||_2^2$.

However, for all the models on which we perform our experiments, the autoencoder was trained on different dataset than the diffusion backbone. The LDM model utilizes VQ-VAE[182] trained by rom [24] on the Open Images Dataset V4 [142] dataset. All other models use KL autoencoder [24, 135] provided by Stablility AI [25]. This model was first trained on the Open Images Dataset V4 and then finetuned on

subsets of LAION-Aesthetics [203] and LAION-Humans [203] datasets. To keep compatibility with existing models trained by Stability AI, only the decoder part was finetuned. Accounting for the difference between the underlying autoencoder and diffuser training datasets present in all models on which we evaluate CDI we do not employ the ARL feature in our framework in this paper. However, we note that it constitutes another source of membership signal applicable in cases when both stages of the latent DM were trained on the same dataset.

C.8. Extending CDI

CDI provides an inherently flexible framework for identifying data collections used in the training of DMs. In this section, we discuss how CDI can be extended with additional feature extraction methods and applied to a broader range of DM architectures and data modalities.

C.8.1. Extending CDI with Features from Novel MIAs

As the field advances, CDI can be extended with additional feature extraction methods based on novel MIAs. We demonstrate this by incorporating CLiD [258] as a feature extraction method within CDI. First, in Table C.8.1, we show that the performance of CLiD MIA remains limited for models trained on ImageNet (first five columns). While it significantly outperforms previous MIAs, as compared to table C.21.1, it still does not allow reliable membership identification for ImageNet-trained models. Note that we do not evaluate CLiD on U-ViT256-Uncond, as it requires both conditional and unconditional inputs to the DM.

Table C.8.1. CLiD [258] MIA results at a TPR@FPR=1%. Values in the table are in %.

Model	LDM	U-ViT256	DiT-256	U-ViT512	DiT512	U-ViT256-T2I	U-ViT256-T2I-Deep
TPR@FPR=1%	1.52	2.18	2.74	2.03	2.00	10.91	24.11

Next, we employ features from CLiD in CDI and present in Table C.8.2 that it further improves our method. This confirms our premise from Section 8.4.2, that more meaningful features improve our method. It also demonstrates how CDI benefits from advancements in MIAs. When we extend CDI with CLiD, the method needs as few as 30 samples to identify data collections used in DM training.

C.8.2. Extending CDI to Novel DMs

With new DM architectures introduced, CDI can be applied to a broader range of DMs. In Table C.8.3, we evaluate the performance of CDI on SiT [156], MDT [103],

Table C.8.2. Effect of extending CDI with features extracted using CLiD. We report the minimal sample size of **P** needed to confidently reject the null hypothesis with CDI.

Model	LDM	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-T2I	U-ViT256-T2I-Deep
Base CDI	6000	3000	2000	2000	1000	200	70
With CLiD	4000	700	400	2000	700	50	30

and DiMR [151], and show that it successfully identifies the data collections used in DM training.

Table C.8.3. Performance of CDI on additional DMs. We report the minimal sample size of **P** needed to confidently reject the null hypothesis with CDI.

Model	SiT-XL/2	MDTv2-XL/2	MDTv1-XL/2	DiMR-XL/2R	DiMR-G/2R
$\|\mathbf{P}\|$	300	300	200	2000	2000

C.8.3. Extending CDI to Other Data Modalities

CDI framework is inherently flexible and can be applied to data modalities beyond images, such as text or audio. This extension is straightforward due to the use of feature extraction methods that operate in the latent space of DMs.

Table C.8.4. Model details. We report the training details for the models used in this paper in the context of the minimal sample size of **P** needed to confidently reject the null hypothesis with CDI.

	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Model parameters	395M	500M	675M	500M	676M	44M	45M	58M
Training steps	178k	500k	400k	500k	400k	1M	1M	1M
Batch size	1200	1024	256	1024	256	256	256	256
Dataset	ImageNet	ImageNet	ImageNet	ImageNet	ImageNet	COCO	COCO	COCO
Dataset size	1.2M	1.2M	1.2M	1.2M	1.2M	83k	83k	83k
Min. P size	6000	3000	2000	2000	1000	300	200	70

C.9. Experimental Setup

C.9.1. Models

We evaluate the effectiveness of CDI on publicly available state-of-the-art DMs. This section provides an overview of the models used in our experiments. For more detailed information about these models and their training procedure, readers are encouraged to consult the original papers. Additionally, we make the model checkpoints readily accessible for download to facilitate the replication of our results.

Table C.9.1. Feature extraction time for different features. Time in seconds is given for processing 1 batch of 64 samples on an A100 GPU.

	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Denoising Loss	4.06	10.55	12.87	9.64	40.77	2.01	2.28	2.69
SecMI$_{stat}$	8.41	23.22	22.16	26.35	96.02	3.27	4.09	4.90
PIA	5.21	6.45	6.87	8.77	26.04	1.72	2.24	2.56
PIAN	5.58	6.93	7.22	9.21	28.04	1.94	2.42	2.78
Gradient Masking (GM)	31.90	81.67	72.42	90.33	110.57	9.14	11.56	15.49
Multiple Loss (ML)	7.11	20.28	18.42	22.55	70.28	2.92	3.45	4.26
Noise Optim (NO)	94.64	64.12	64.17	78.14	181.33	182.23	205.78	120.26

All models, with the exception of LDM256 utilize ViT[75] as the diffusion backbone. LDM256 uses the UNet architecture [192] instead, being prior work in the area of latent DMs.

LDM256 - a class-conditioned LDM checkpoint provided by rom [24], trained on ImageNet dataset in 256x256 resolution. The diffuser backbone of this model is a UNet architecture with 395M parameters.

DiT256, *DiT512* - class-conditioned DiT-XL/2 checkpoints provided by Peebles and Xie [174], trained on ImageNet dataset in 256x256 and 512x512 resolutions respectively. DiT256 has 675M parameters and DiT512 has 676M parameters.

U-ViT256, *U-ViT512* - class-conditioned U-ViT-Huge/4 checkpoints provided by Bao et al. [40], trained on ImageNet dataset in 256x256 and 512x512 resolutions respectively. U-ViT256 has 500.8M parameters and U-ViT512 has 500.9M parameters.

U-ViT256-T2I - a text-conditioned U-ViT-Small/4 checkpoint provided by Bao et al. [40], trained on COCO dataset in 256x256 resolution.

U-ViT256-T2I-Deep - a text-conditioned U-ViT-Small/4-Deep checkpoint provided by Bao et al. [40], trained on COCO dataset in 256x256 resolution. The model architecture differs from U-ViT256-T2I by having a larger number of transformer blocks (16 instead of 12). U-ViT256-T2I and U-ViT256-T2I-Deep have 45M and 58M parameters respectively.

U-ViT256-Uncond - an unconditioned U-ViT-Small/4 checkpoint trained on COCO dataset in 256x256 resolution. We train the model for 1.000.000 training steps, following the configuration used by Bao et al. [40] for *U-ViT256-T2I* checkpoint. Namely, we use *AdamW*[154] optimizer (lr 2×10^{-5}, weight decay 0.03, betas (0.99,0.99)) and batch size 256. U-ViT256-Uncond has 44M parameters.

C.9.2. Compute Resources and Feature Extraction Time

We compute our experiments on A100 80GB NVIDIA GPUs on an internal cluster, amounting to 150 GPU-hours. Training the U-ViT256-Uncond model requires

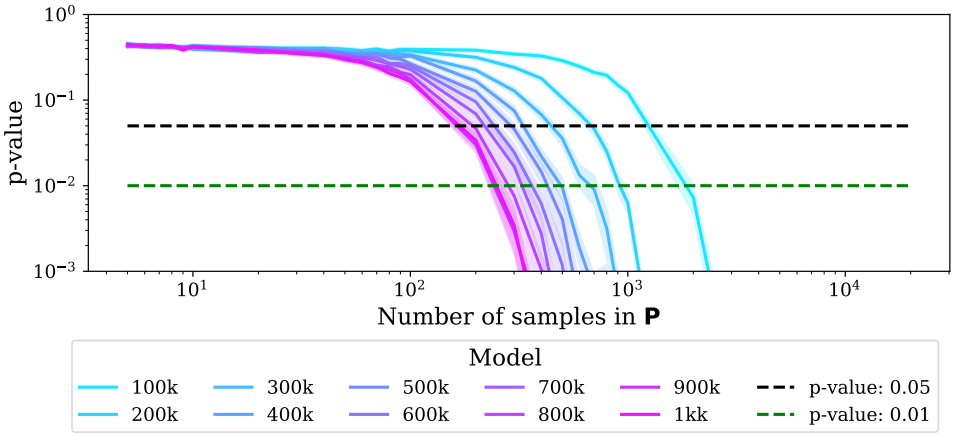


Figure C.9.1. Results of `CDI` for U-ViT256-Uncond for different number of model's training steps Solid lines indicate aggregated p-values aggregated over 1000 randomized trials for each size of **P**, shaded areas around the lines are 95% confidence intervals. The higher the number of the model's training steps the fewer suspect samples are required from the data owner to confidently reject H_0.

additional 80 GPU-hours. We provide the time needed to extract features from a batch of 64 samples in table C.9.1. Fitting the scoring model s of `CDI` does not require GPU utilization and can be executed in a negligible time (<10 seconds) on a CPU.

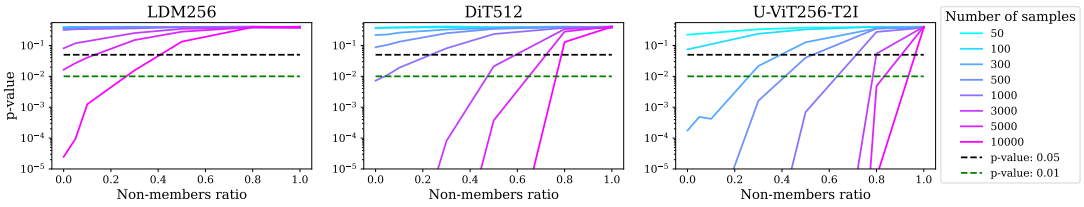


Figure C.9.2. Impact of non-members ratio in P on `CDI`, and resilience against false positives. The lines represent p-values for a given non-member ratio while varying sizes of **P**. Note that for the non-members ratio of 1 (all samples in **P** are non-members), the p-values are always significantly above the significance level ($\alpha = 0.01$), which means `CDI` does not return false positive answers.

C.10. Model Details and `CDI` Effectiveness

Based on Table C.8.4 we make the following observations. (1) For a given DM architecture, trained on a given dataset, the number of samples required for the confident identification of the training data decreases with increasing input resolution. This phenomenon is clearly visible when comparing the required number of samples for U-ViT256 and U-ViT512 and for DiT256 and DiT512 which differ only in the input image resolution (2) Models trained on smaller datasets exhibit stronger signal for identifying training data, as evident when contrasting the results

on models trained with ImageNet (LDM, U-ViT256, U-ViT512, DiT256, DiT512) and COCO dataset (U-ViT256-T2I, U-ViT256-T2I-Deep, U-ViT256-Uncond).

C.11. Number of Model Training Steps and CDI Effectiveness

To assess the effect of the number of DM's training steps on CDI, we conduct the following experiment. We measure the effectiveness of CDI for U-ViT256-Uncond model after every 100,000 training steps. The results in Fig. C.9.1 confirm that CDI is effective even for models trained for a limited number of steps. With enough samples provided by the data owner $|\mathbf{P}|$ = 2,000, CDI confidently rejects H_0 for U-ViT256-Uncond after just 100,000 training steps.

C.12. More Results for the Non-members in P and False Positives

We present the additional results on CDI's robustness in cases when $\mathbf{P}$ contains (some) non-member samples in Fig. C.9.2. We expand Fig. 8.5.3 for DiT512 with experimental results for LDM256 and U-ViT256-T2I. CDI remains effective even when not all data samples are used as training data. *i.e.,*, $\mathbf{P}$ contains a certain ratio of non-members, while the remaining samples in $\mathbf{P}$ are members. We observe that CDI demonstrates greater robustness in cases when part of $\mathbf{P}$ contains non-members if the data owner supplies larger $\mathbf{P}$. This is evident in the decreasing p-value at a given non-members ratio as the size of $\mathbf{P}$ increases. Importantly, for the non-members ratio of 1, the p-values are significantly above the significance level (α = 0.01), which means CDI does not return false positive answers.

Table C.12.1. Performance of CDI under $|\mathbf{U}|$ and $|\mathbf{P}|$ imbalance. We report the minimal sample size of $\mathbf{P}$ needed to confidently reject the null hypothesis with CDI.

$\lvert\mathbf{U}\rvert/\lvert\mathbf{P}\rvert$ ratio	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-T2I	U-ViT256-T2I-Deep
1.00	6000	3000	2000	2000	1000	200	70
0.90	6000	3000	2000	2000	2000	200	70
0.75	6000	3000	2000	3000	2000	200	70
0.50	6000	3000	2000	3000	2000	300	90
0.25	7000	4000	2000	4000	2000	400	200
0.10	8000	5000	4000	6000	5000	800	300

Table C.12.2. Performance of CDI in gray- vs white-box model access. We depict the number of samples required from the data owner in $\mathbf{Q}_{\text{test}}$ to reject the null hypothesis for different model access scenarios. Our CDI framework remains effective when assuming the more restrictive gray-box model access, provided with larger $\mathbf{P}$.

Access Type	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Gray-Box Model	8000	3000	2000	4000	2000	400	200	70
White-Box Model	6000	3000	2000	2000	1000	300	200	70

C.13. Results under |P| and |U| Imbalance

In practice, the training-to-test set ratio ($|U|/|P|$) is often much smaller than one. We analyze CDI's performance in such scenarios and find it remains effective even when the victim has access to fewer test samples ($|U| < |P|$). We report the number of samples in P needed to reject H_0 for given ratios ($|U| = \text{ratio} \times |P|$). Notably, an extreme set imbalance can be mitigated by increasing P.

C.14. MIAs and their Model Access Types

We group features, which we use in CDI, into two categories, based on their respective access type. Because Gradient Masking and Noise Optimization utilize gradient calculations they require white-box access to the DM's network weights. To execute the remaining feature extraction method CDI needs only gray-box access, *i.e.,* the ability to predict the noise added to a clean sample at an arbitrary timestep t. We specify the model access type per each feature in table C.14.1.

Table C.14.1. Each feature with its Model Access Type.

MIA	Model Access Type
Denoising Loss [47]	gray-box
SecMI$_{stat}$ [79]	gray-box
PIA [137]	gray-box
PIAN [137]	gray-box
Multiple Loss (ML)	gray-box
Gradient Masking (GM)	white-box
Noise Optimization (NO)	white-box

C.15. Gray-box vs. White-box Comparison

CDI remains effective even in a gray-box model access scenario. We assess the effectiveness of CDI within a gray-box model access scenario, as specified in our threat model (section 8.4). In this setup, only the original MIA features and Multiple Loss are included, whereas the white-box model access setup leverages the full set

of features. Our results in table C.12.2 show that CDI maintains effectiveness under gray-box access, successfully rejecting the null hypothesis, though it requires a larger sample size in $\mathbf{P}$.

C.16. Analysis of the Scoring Function

Fig. 8.5.2 demonstrates the varying impact of features on CDI performance. We further aid our analysis of the scoring function by SHAPley [155] summary plots in fig. C.16.1. The results illustrate the model-specific nature of feature importance. This highlights the necessity of a scoring function that can agnostically learn the optimal feature utilization for each dataset and DM, leading to more confident and adaptable membership estimations, ultimately enhancing CDI's effectiveness.

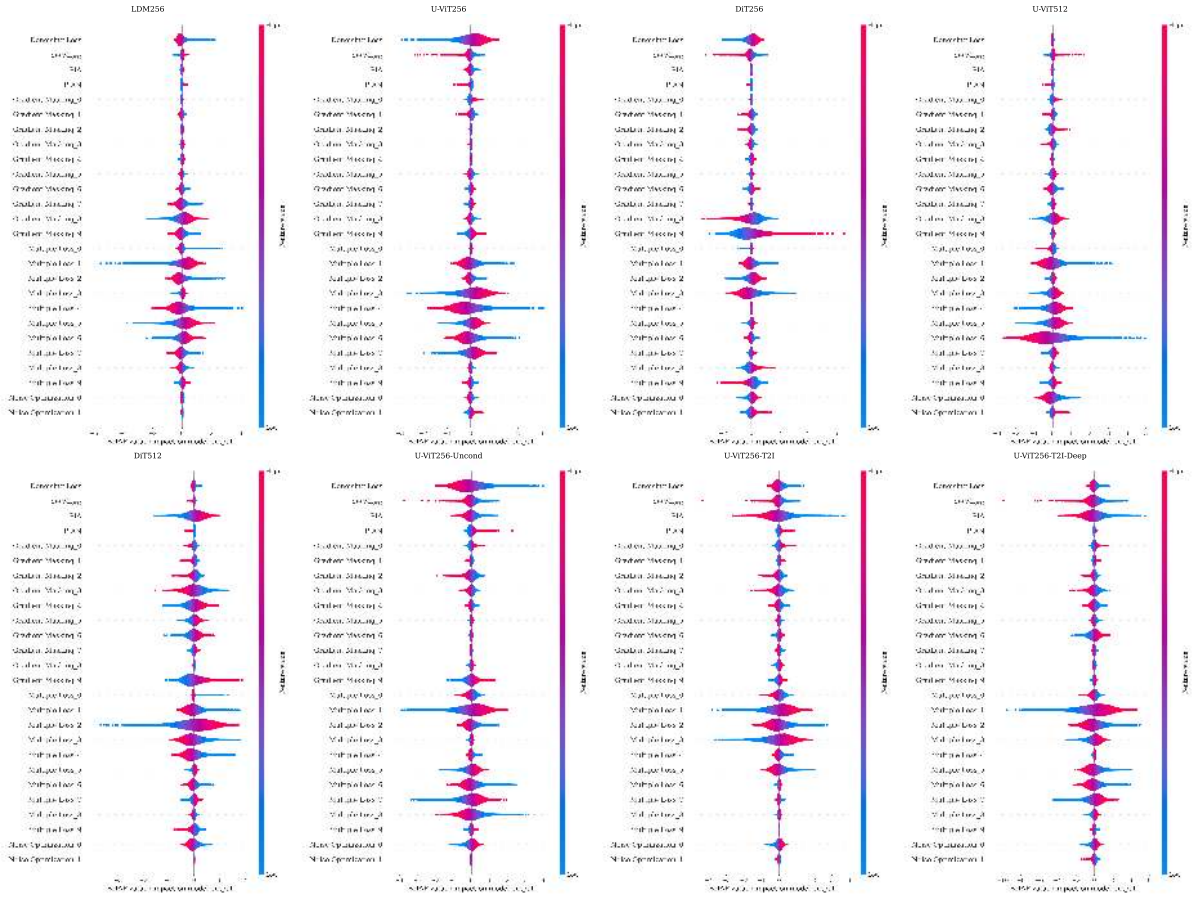


Figure C.16.1. SHAPley plots for all models and features. The scoring functions have been trained using 5,000 members as $\mathbf{P}_{\text{ctrl}}$ and 5,000 non-members in $\mathbf{U}_{\text{ctrl}}$, The resulting plots are evaluated on 20,000 members in $\mathbf{P}_{\text{test}}$ and 20,000 non-members in $\mathbf{U}_{\text{test}}$ to provide the most accurate results.

C.17. From p-Value to TPR@FPR=1%.

The ablation study on the importance of the statistical testing in CDI we conduct in section 8.5.2 requires us to transform p-values returned by CDI to **TPR@FPR=1%** to directly compare CDI with set-level MIA in table 8.5.1. We note that the TPR of a statistical test is equivalent to its power. We estimate the power by computing the effect size, using Cohen's d [60], which is defined as $d = \frac{\bar{x_1} - \bar{x_2}}{s}$. x_1 and x_2 are the observed scores, and $s = \sqrt{\frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2}}$. n_1 and n_2 are the sizes of the population (here: the number of samples in **P** and **U**, respectively), and $s_j^2 = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} (x_{1,j} - \bar{x_j})^2$ for $j = 1, 2$. After we obtain the effect size, we compute the power of the t-test using a solver, setting $\alpha = 0.01$ to get **TPR@FPR=1%**.

C.18. ROC curves of CDI

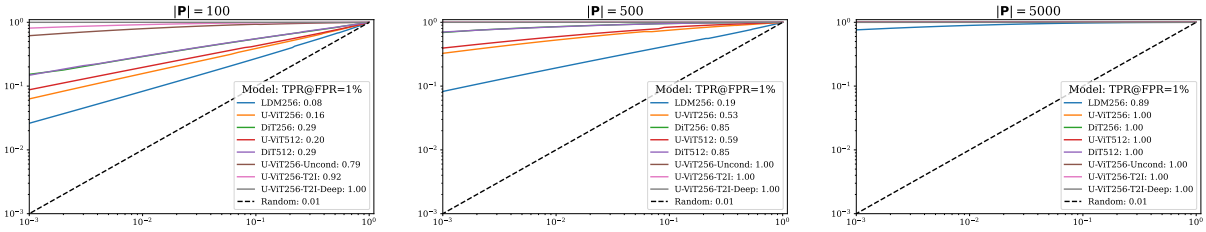


Figure C.18.1. ROC curves of CDI. CDI achieves perfect performance with sufficiently large |**P**|. Resulting **TPR@FPR=1%** align with p-values we report in Fig. 8.5.1.

In section 8.5.1, we evaluate CDI following the evaluation methodology proposed for DI by [157]. In this section, we additionally extend this evaluation by analyzing CDI through ROC curves and true positive rates (TPR).

We build on section 8.5.2 and section C.17, and obtain TPR at varying FPR by sweeping over α values from 0 to 1. To ensure stability of our results, for each size of **P** we sample 1000 subsets from **P** and **U**, compute the TPR, and finally average the TPR to get the final value. We visualize our results in fig. C.18.1. Key takeaways from the figure are: (1) CDI achieves perfect performance given large enough |**P**|, even at **FPR=0%**. (2) CDI is not over-confident. P-values we report in Fig. 8.5.1 align with the resulting **TPR@FPR=1%**, thanks to the careful choice of α (ref. section 8.4.3). Intuitively, low p-values correspond to high **TPR@FPR=1%**. For example, LDM256 for |**P** | = 5000 achieves p-values close to 0.01, and **TPR@FPR=1%** of 0.89. (3) In effect, CDI is not susceptible to p-value hacking, *i.e.,* the improvement in CDI's performance observed in CDI's higher confidence (lower p-value) signifies higher TPR at lower FPR.

C.19. MIAs Fail on DI Task

In this experiment, we compare CDI to MIAs on the task of DI. Similarly to set-level MIA we introduce in section 8.5.2, we follow this procedure: We apply a MIA to each sample in the suspect set, returning Positive prediction if the score for any sample exceeds a certain threshold. Repeating this process for every suspect set and varying the threshold yields a ROC curve for each MIA. Finally, we compare these ROC curves against the ROC curve for CDI, obtained as in App. C.18.

We sample 1000 **P** containing only member samples (Positive) and 1000 **P** containing only non-member samples (Negative). **U** remains unchanged, *i.e.,* contains only non-members. We vary the size of P for a more thorough analysis and compute ROC curves.

We demonstrate in Fig. C.21.1 that CDI achieves TPR orders of magnitude higher than MIAs on the DI task. Applying single-sample MIAs to DI is challenging. Our experiments show MIAs are unstable, with high FPR due to set-wise confidence swayed by a single high score. Notably, increasing the size of **P** does not improve the performance. CDI's statistical testing is robust by comparing distributions of membership scores and capturing subtle differences. Then, the application of the t-test in CDI quantifies these differences, with the p-value serving as a reliable confidence measure.

C.20. Further Applicability of Our Novel Features

One of our contributions is the new features introduced in section 8.4.1. As we use these features to fit our *scoring function*, we can also use them to perform the default *threshold* MIA. We introduce the detailed results of that experiment in App. C.21. Here, we analyze the characteristics of our features.

C.20.1. Gradient Masking

Recall, to obtain this feature we: (1) distort 20% of the input image's latent based on the absolute gradient values. (2) Use the DM to reconstruct the distorted part by performing a single denoising step. (3) Compute L2 reconstruction loss over the distorted region.

Visualization. We visualize the reconstruction effect in fig. C.20.6. We observe that: (1) the reconstructed members resemble semantics of the original images better than the reconstructed non-members do. (2) For members and non-members there is a notable decrease in the level of detail in the images after reconstruction,

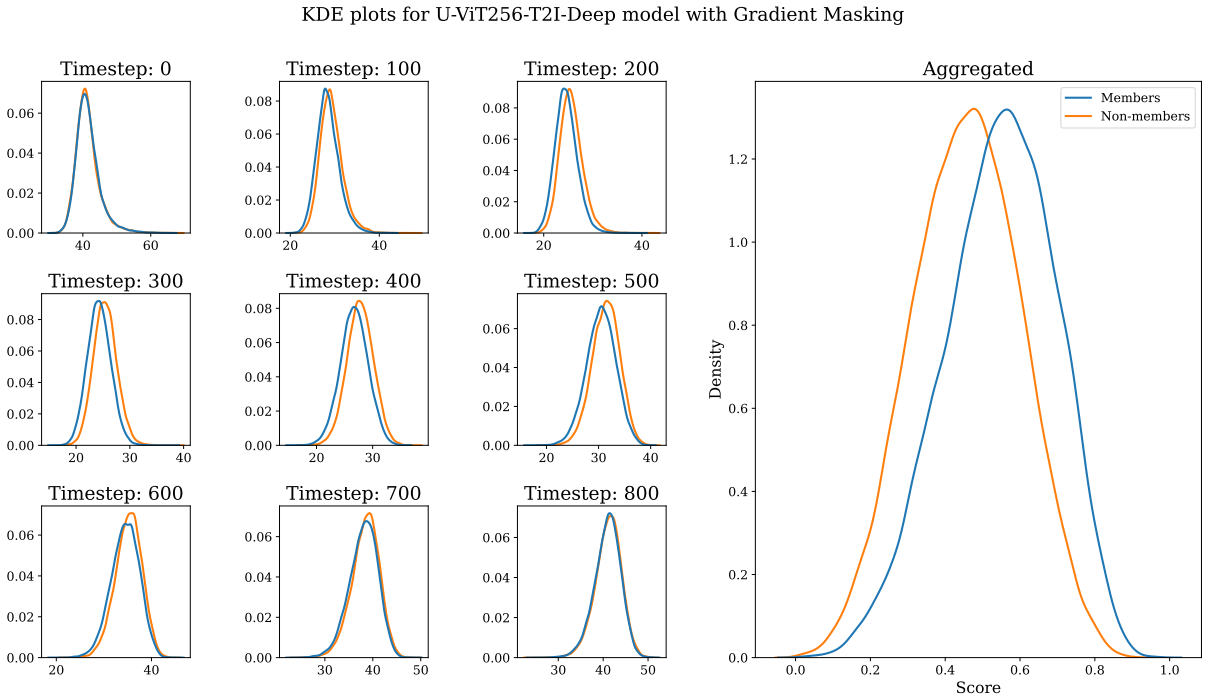


Figure C.20.1. KDEplots of the Gradient Masking features for timesteps, and after aggregation. The model used is U-ViT256-T2I-Deep.

e.g., for U-ViT256-T2I-Deep we observe that for the member sample the details of the painting and the table are gone. (3) The crude elements of the reconstructed images stay unchanged, *e.g.,* for ImageNet models we can see that the reconstructed image contains a dog (for the member sample), or a turtle in the grass (the non-member). (4) Distortion is not uniform between the models. For the ViT-based models we observe distortions that are spread out throughout the image, while for the LDM (UNet-based model) the distortions are more local.

Features. In fig. C.20.1 we analyze the distributions of the features (reconstruction errors) throughout various timesteps, and then we compare them with the distribution obtained by aggregating these features with our *scoring function*. We note that the difference between members and non-members is negligible for singular features. However, after we aggregate them we observe a significant difference.

C.20.2. Noise Optim and Multiple Loss

For these two features our findings are in line with the findings for Gradient Masking. In fig. C.20.2 and fig. C.20.3 we visualize the distributions of singular features and the distributions after aggregation.

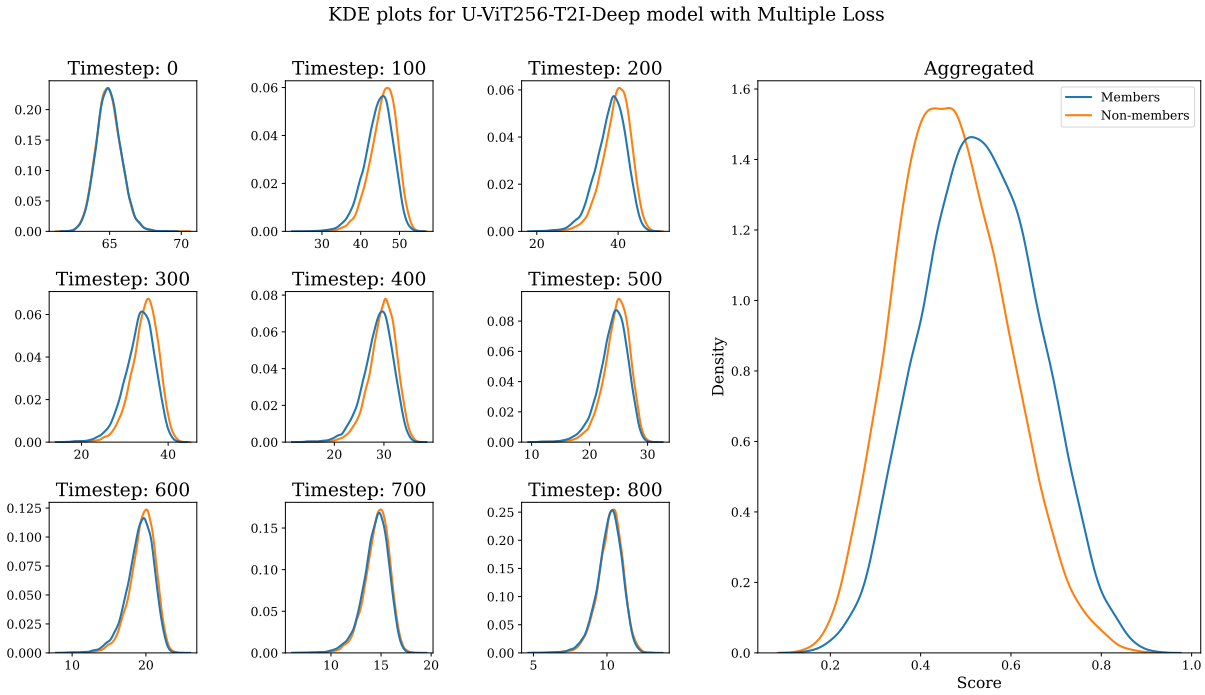


Figure C.20.2. KDEplots of the Multiple Loss features for timesteps, and after aggregation. The model used is U-ViT256-T2I-Deep.

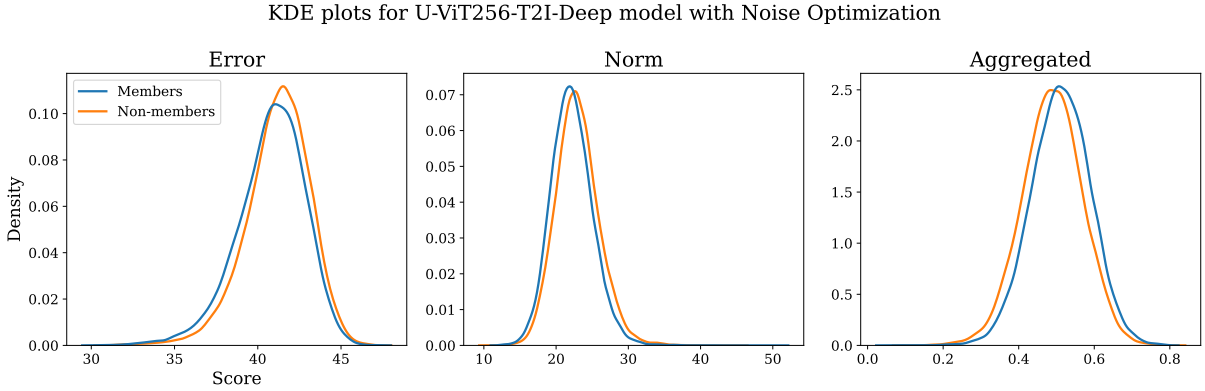


Figure C.20.3. KDEplots of the Noise Optim features, and after aggregation. The model used is U-ViT256-T2I-Deep. Error refers to the reconstruction error obtained after optimization, and Norm is the L2 norm of added perturbation.

C.20.3. Effect of aggregation on MIA and `CDI` performance

We compare the metrics for MIA and `CDI` when we use a singular feature vs when we aggregate the features. In fig. C.20.4 we observe that while some singular features barely cross the threshold of random guessing (TPR=1%), the aggregate provides better results (although the effectiveness is limited). For `CDI`, in fig. C.20.5 we observe that aggregation allows us to lower the p-value by orders of magnitude lower than the best singular feature can.

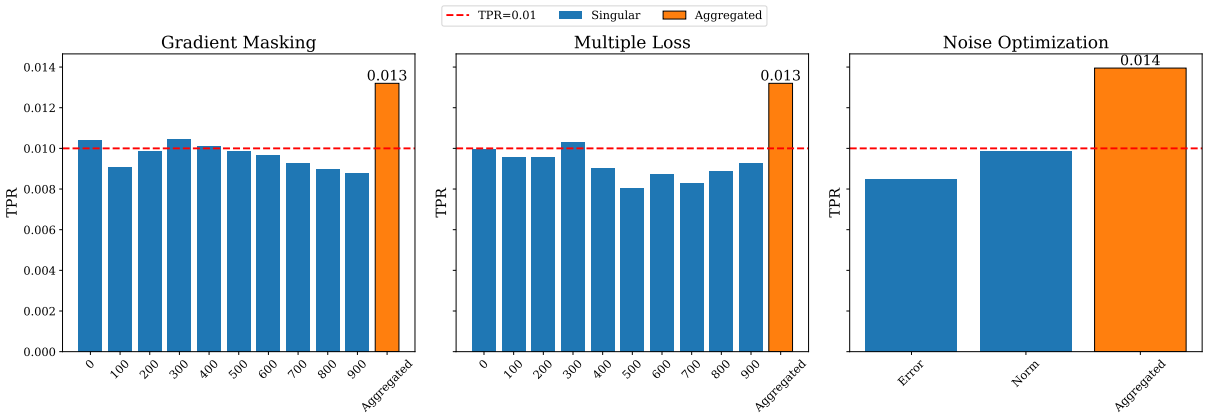


Figure C.20.4. TPR@FPR=1% (↑) for singular features, and after aggregation The model used is LDM256. For Gradient Masking and Multiple Loss the numerical values correspond to the timestep at which the feature is computed.

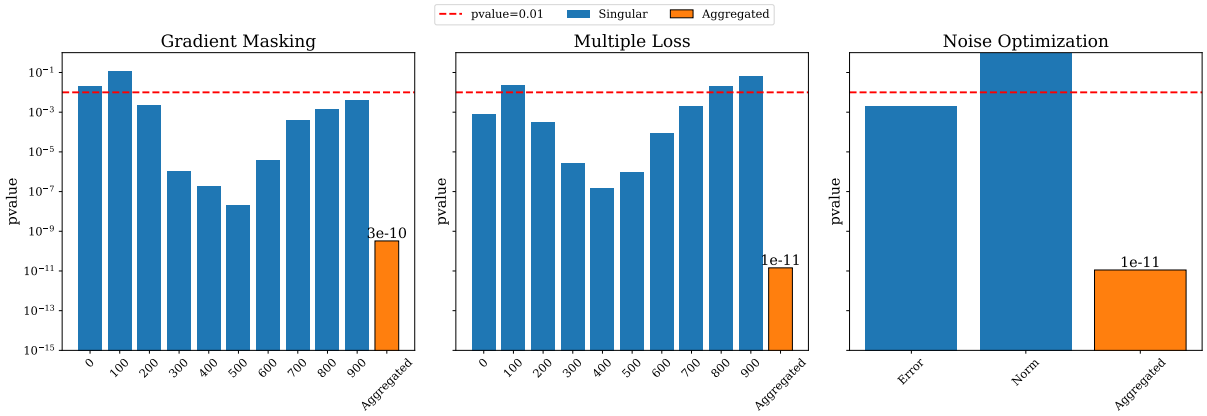


Figure C.20.5. P-values (↓) for singular features, and after aggregation The model used is LDM256. For Gradient Masking and Multiple Loss the numerical values correspond to the timestep at which the feature is computed.

C.21. Further Evaluation of MIAs

The following summarizes an extensive evaluation effort for MIAs utilizing existing, and, for completeness, our proposed novel features used as MIA. We follow identical setting as described in App. C.21, and extend the results by Area Under the Curve (AUC) score (Table C.21.3), and accuracy (Table C.21.2). To perform MIA on novel features we do the following: (1) Fit s on the features extracted from $|\mathbf{P}_{\text{ctrl}}|$ = 5000 and $|\mathbf{U}_{\text{ctrl}}|$ = 5000. (2) Obtain predictions on $\mathbf{P}_{\text{test}}$ and $\mathbf{U}_{\text{test}}$. Importantly, $\mathbf{P}_{\text{test}} \cap \mathbf{P}_{\text{ctrl}} = \varnothing$ and $\mathbf{U}_{\text{test}} \cap \mathbf{U}_{\text{ctrl}} = \varnothing$. (3) Use these scores to run MIA. We include accuracy and AUC to better understand the differences between the proposed features, as well as their impact on the CDI. In Fig. C.21.2 we visualize the behavior of the Receiver Operating Characteristic (ROC) curve for all features under MIA setting.

We note that, similarly to Fig. 8.5.2, Gradient Masking provides stronger sig-

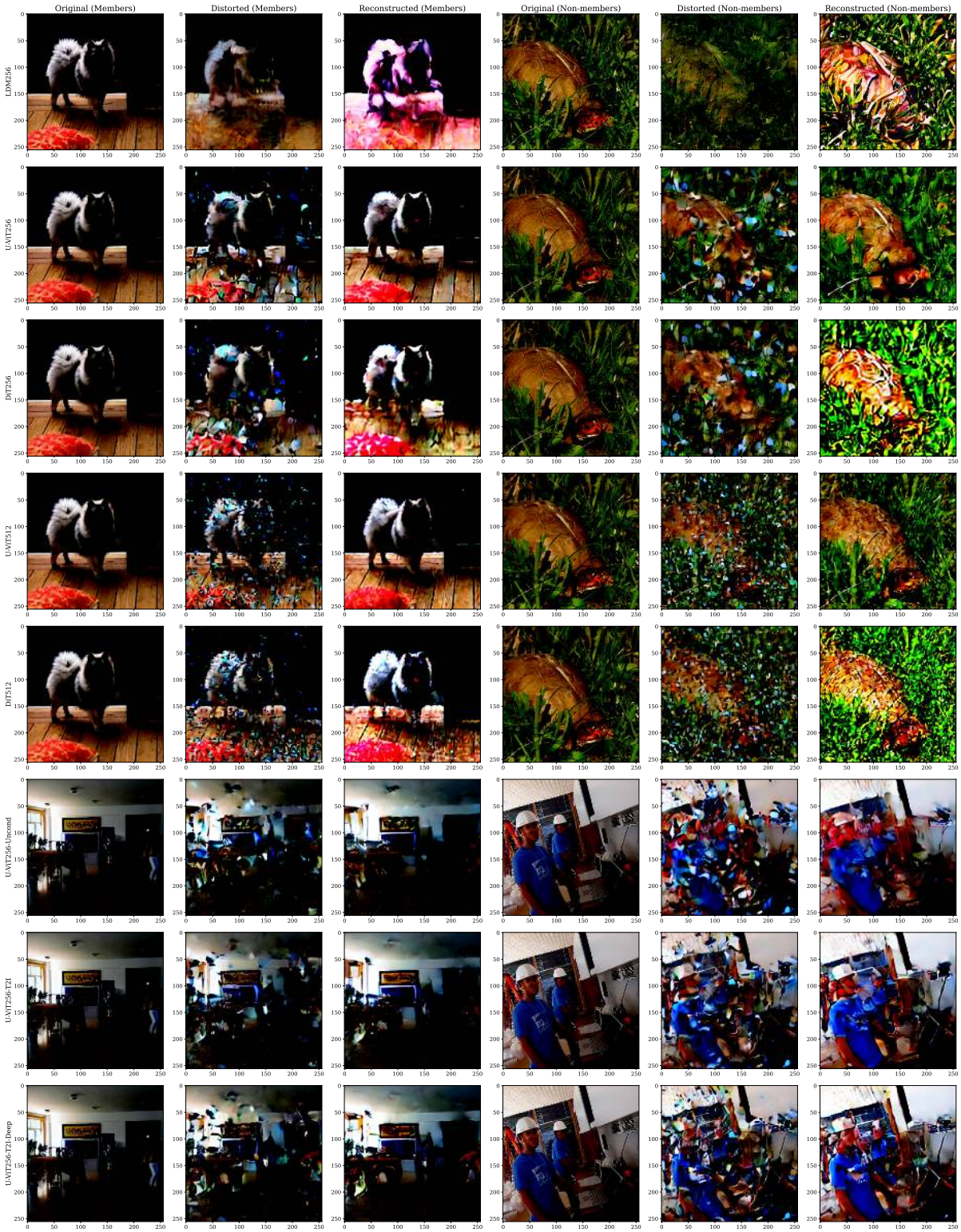


Figure C.20.6. Effect of the reconstruction method for the Gradient Masking. The images in the figure are distorted using noise scale corresponding to the timestep=500, and are denoised using the same timestep. The images differ between models due to the difference in their respective training datasets (ImageNet and COCO2017).

nal than Multiple Loss and Noise Optimization, resulting in higher values of

TPR@FPR=1% (Table C.21.1), AUC, and accuracy in almost all cases. GM underperforms compared to ML only for U-ViT256 and U-ViT512.

We observe higher performance of MIAs for models trained on smaller datasets, *i.e.,* U-ViT256-Uncond, U-ViT256-T2I, U-ViT256-T2I-Deep.

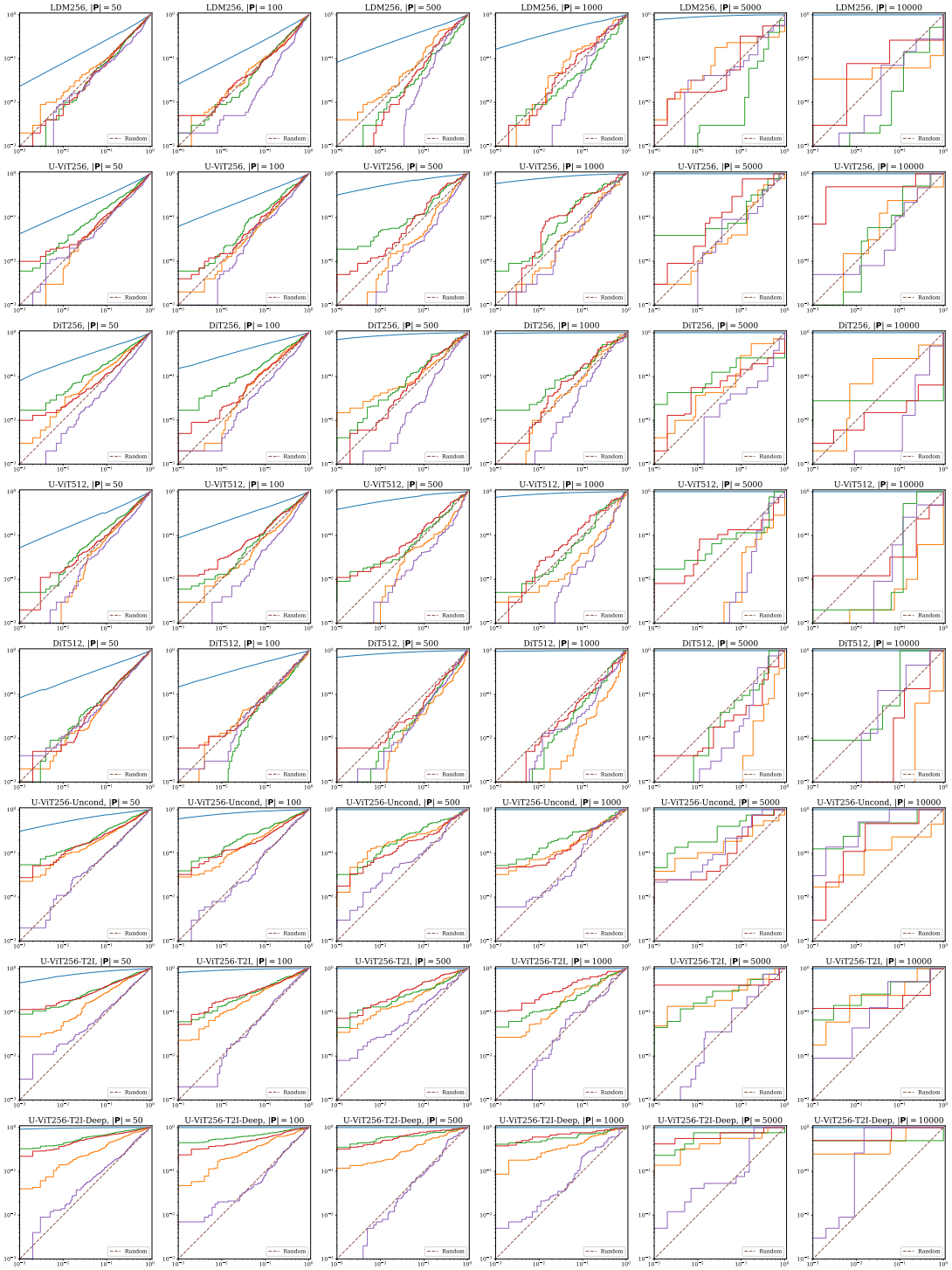


Figure C.21.1. Comparison of CDI and MIAs on the DI tasks.

Table C.21.1. MIA results at a TPR@FPR=1%. Values in the table are in %. We include the performance of MIAs using our novel features and note that they perform comparably with the SOTA PIA method, or even outperform it in some cases.

	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Denoising Loss [47]	1.23±0.10	1.22±0.10	1.34±0.12	1.61±0.12	1.78±0.12	1.72±0.15	1.73±0.14	2.25±0.20
SecMI$_{stat}$ [79]	1.17±0.11	1.17±0.11	1.31±0.14	1.26±0.10	1.16±0.13	1.67±0.19	1.78±0.14	2.41±0.24
PIA [137]	1.12±0.09	1.18±0.10	1.54±0.13	1.25±0.13	1.14±0.13	2.79±0.20	2.84±0.22	5.57±0.41
PIAN [137]	0.88±0.09	1.18±0.10	0.96±0.10	1.52±0.12	1.13±0.10	0.88±0.10	0.87±0.09	0.78±0.09
Gradient Masking	1.32±0.12	1.20±0.12	1.83±0.15	1.07±0.12	1.28±0.12	2.56±0.16	2.98±0.22	4.71±0.30
Multiple Loss	1.31±0.11	1.43±0.14	1.57±0.14	1.43±0.10	1.48±0.13	2.39±0.18	2.19±0.26	3.10±0.28
Noise Optimization	1.39±0.11	1.67±0.13	1.35±0.12	1.25±0.16	1.25±0.13	1.63±0.13	1.66±0.15	1.81±0.17

Table C.21.2. Accuracy of MIA on all features. Values are in %. Here we observe that all novel features outperform already existing ones. Note that for all models trained on ImageNet (first five columns from the left) we observe results very close to 50%, essentially random guessing. For models trained on COCO (remaining three columns on the right) we observe an improvement for Gradient Masking and Multiple Loss, while the MIAs from section 8.3.1 remain close to 50%.

	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Denoising Loss [47]	50.02±0.01	50.07±0.03	50.09±0.03	50.14±0.04	50.20±0.04	50.01±0.01	50.01±0.01	50.01±0.01
SecMI$_{stat}$ [79]	50.03±0.03	50.04±0.04	50.17±0.20	50.03±0.09	49.97±0.04	52.92±2.89	50.40±0.50	53.57±3.55
PIA [137]	50.01±0.01	50.02±0.03	50.28±0.06	50.02±0.03	50.03±0.03	50.07±0.02	50.03±0.01	50.03±0.01
PIAN [137]	49.51±0.12	49.87±0.06	49.81±0.07	49.82±0.17	49.80±0.13	50.02±0.15	50.10±0.15	50.05±0.12
Gradient Masking	50.00±0.01	50.01±0.01	51.61±0.46	50.00±0.01	50.78±0.13	53.60±0.60	55.26±0.68	61.85±0.30
Multiple Loss	50.07±0.07	50.00±0.01	50.08±0.06	50.00±0.00	50.06±0.06	52.75±0.19	53.70±0.29	59.79±0.26
Noise Optimization	50.00±0.00	50.00±0.00	50.00±0.00	50.09±0.04	50.00±0.00	50.03±0.02	50.24±0.06	52.72±0.20

Table C.21.3. AUC score for MIAs. We observe that for this metric PIA outperforms all standalone features for models trained on COCO (last three columns on the right) while MIAs based on Gradient Masking, Multiple Loss and Noise Optim achieve better performance for models trained on ImageNet (first five columns on the left) in almost all cases.

	LDM256	U-ViT256	DiT256	U-ViT512	DiT512	U-ViT256-Uncond	U-ViT256-T2I	U-ViT256-T2I-Deep
Denoising Loss [47]	50.55±0.28	50.29±0.30	51.71±0.29	49.99±0.29	50.19±0.29	56.47±0.28	57.38±0.28	60.77±0.29
SecMI$_{stat}$ [79]	49.59±0.28	53.06±0.29	55.22±0.28	50.92±0.30	50.69±0.30	59.28±0.29	61.56±0.28	69.20±0.26
PIA [137]	49.02±0.29	51.65±0.29	53.07±0.29	50.79±0.29	49.98±0.29	59.97±0.27	63.99±0.28	71.18±0.25
PIAN [137]	49.41±0.28	50.69±0.29	49.88±0.28	49.82±0.28	49.07±0.29	49.63±0.30	49.68±0.29	49.35±0.28
Gradient Masking	51.66±0.29	53.06±0.29	54.83±0.29	53.67±0.29	56.05±0.28	59.36±0.29	59.88±0.28	66.80±0.27
Multiple Loss	51.77±0.30	53.63±0.29	54.16±0.30	52.30±0.30	52.98±0.29	58.17±0.29	58.93±0.28	64.26±0.27
Noise Optimization	51.85±0.29	52.73±0.29	52.15±0.29	54.09±0.29	51.93±0.29	54.28±0.28	55.51±0.29	58.04±0.27

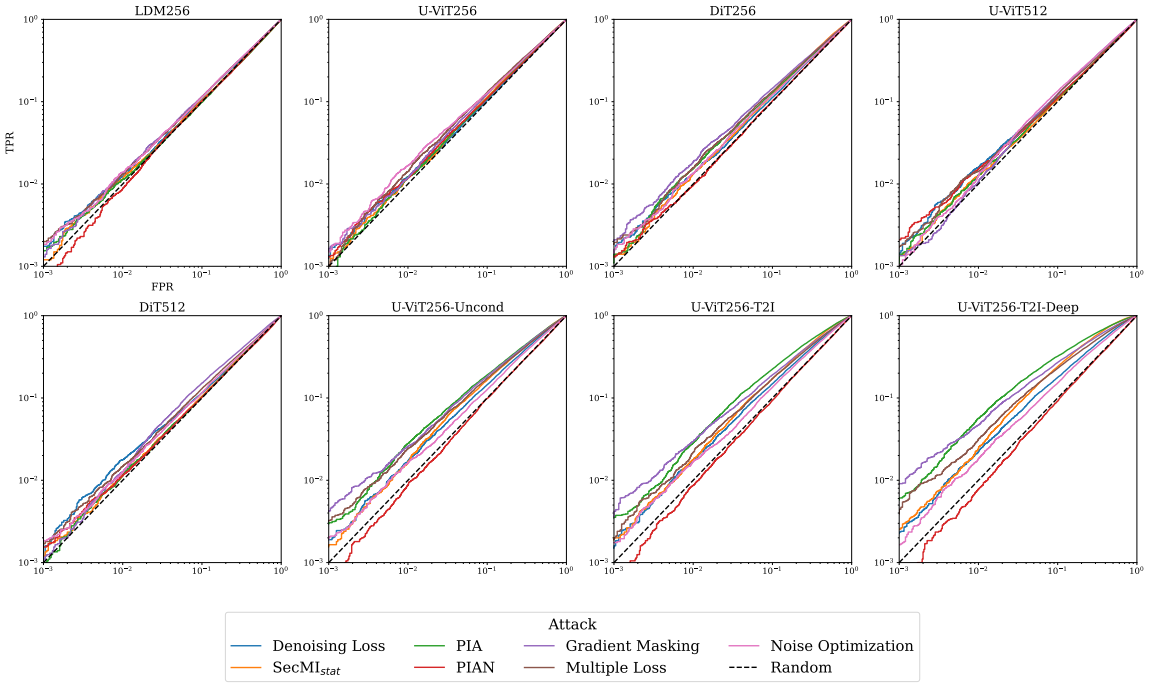


Figure C.21.2. ROC curves for MIAs against all models. X- and Y-axis are in logarithmic scale. We observe that for DMs trained on ImageNet the TPR in low FPR regime (< 1%) is not better than random guessing, while for the models trained on COCO (last three from the right in the bottom row) all methods but PIAN achieve results significantly better than random chance.

D. Supplement for Privacy Attacks on Image AutoRegressive Models

D.1. Broader Impact

Image autoregressive models (IARs) have rapidly gained popularity for their strong image generation abilities. However, the privacy risks that come associated to these advancements have remained unexplored. This work makes a first step towards identifying and quantifying these risks. Through our findings, we highlight that IARs *empirically* experience significant leakage of private data. These findings are relevant to raise awareness of the community and to steer efforts towards designing dedicated defenses. This enables a more ethical deployment of these models.

D.2. Why IARs Leak More Privacy Than DMs?

In the following we provide insights explaining the higher leakage observed in IARs. First, we focus on differences in architectures and models' internals. Then, we switch to explore architecture-agnostic factors like model size.

D.2.1. Inherent Differences Between IARs and DMs

We note that DMs have inherently different characteristics than IARs, and we link them to the privacy risks they exhibit. We identify three key factors:

1. **Access to $p(x)$ boosts MIA** [257]. We note that IARs inherently expose the full information about $p(x)$ at the output (per-token logits, see eq. (9.1)). In contrast, DMs do not, as they learn to transform $\mathcal{N}(0,I)$ to the data distribution $q(x)$ by iterative denoising process. This difference is expressed with varying MIA designs for DMs and IARs—the former exploit the predicted noise, while the latter work with $p(x)$, by focusing on the logits. Our results confirm this premise—MAR is less prone to all privacy risks, and it does not output $p(x)$. It outputs continuous tokens, sampled from a diffusion module.
2. **AutoRegressive training exposes IARs to more data per update**. For each

training sample passed through the IAR, the model "sees" N different sequences to predict. Conversely, DMs only "sees" a single, noisy image. This influences two factors: a) training time of the model—DMs require to be trained two times longer than IARs, on average. b) privacy leakage—IARs are exposed to more information per each update step, which translates to increased vulnerability for privacy attacks like MIAs, DI, and data extraction. VAR outputs 10 sequences of tokens, and is less prone to MIA than RAR, which outputs 256 sequences, *e.g.,* VAR-d-20 vs. RAR-L (models of similar sizes).

3. **Multiple *independent* signals amplify leakage**. Previous works [83, 158] aggregate signal from many MIAs to yield a stronger attack. Notably, each token predicted by IARs leaks unique information from the model, as it is generated from a (slightly) different prefix. Thus, per-token losses/logits that IAR-specific MIAs use, when aggregated, add up to a more informative signal, which in turn yields stronger MIAs. In contrast, DMs' outputs provide a general direction for the denoising process, and are strongly correlated. In effect, predictions at different timesteps do not provide enough *novel* information to the MIA to boost its strength.

We believe that these reasons are behind greater privacy leakage that we observe for IARs than for DMs.

D.2.2. Architecture-Agnostic Differences Between the Models

The models evaluated in our work differ in many factors. Two of them, model size and training duration, are mostly architecture-agnostic, which means they are less related to the design choices of the specific models. As the efficacy of privacy attacks is directly related to these factors [211], we want to assess if our results *really* show that IARs leak more than DMs. To this end, we collect five variables: TPR@FPR=1% (MIA), P (DI metric), model size, training duration, and *Is IAR* for every model we evaluate in the paper (11 IARs, 8 DMs). For the first two (MIA, DI) we take them directly from tables D.9.5, 9.5.1 and 9.5.3. We obtain the model sizes from tables D.5.1 and D.6.1. Training duration is expressed by a number of data points passed through the model at training, e.g., for RAR-B we have 400 epochs of ImageNet-1k train set, which amounts to $400 \times 1.27\text{M} \approx 0.5\text{B}$ samples seen. *Is IAR* factor is a 1 if the model is IAR, 0 otherwise. We take these variables and compute pairwise Pearson's correlation between them, using values for all the models.

In table D.2.1 we show correlations between factors (columns) and privacy metrics (rows). We identify the following insights:

1. **Training duration** is a factor that increases vulnerability for MIA and DI for DMs the most.

Table D.2.1. Correlation between different factors and privacy leakage. Our results show that while the model-agnostic factors correlate with the performance, the fact that the model is IAR or not also correlates with the leakage.

	Architecture	Training Duration	Model Size	Is IAR
P (DI)	IAR	0.24	-0.39	
P (DI)	DM	-0.58	-0.32	
P (DI)	All	-0.04	-0.28	-0.46
TPR@FPR=1%	IAR	0.17	0.93	
TPR@FPR=1%	DM	0.31	0.11	
TPR@FPR=1%	All	-0.2	0.87	0.38

2. **Model size** influences leakage more for IARs than for DMs.
3. ***Is IAR*** factor plays the most significant role for the DI performance. It also correlates with MIA performance.

Our results show that while these two factors, model size and training duration, influence the performance of our attacks against the models, the results strengthen our notion that IARs tend to leak more privacy than IARs due to their inherent characteristics.

D.3. Limitations

We acknowledge our privacy analysis of the novel IARs, and comparison to DMs suffers from two limitations. We do not evaluate our attacks on the biggest available models (like Infinity [109]) trained on massive (over 1B samples), messy datasets. Secondly, there are many factors crucial for MIA and DI performance, which differ in values between almost all the models. The following explains these issues in more detail.

D.3.1. On the Infeasibility of High-Scale Experiments on Extremely Large Models

We do not assess how our attacks perform when applied to models trained on datasets of a scale higher than 1M samples. It may raise concerns about the scalability of the attacks and the insights they provide to the real-world applications. Unfortunately, IARs trained on bigger datasets than ImageNet-1k (Infinity [109], HART [220]) do not disclose fully what their training data *exactly* is. Because of that, we are unable to perform a sound evaluation of the privacy attacks. We lack the ability to assess MIA's and DI's performance correctly, as these methods rely

on two assumptions: (1) we know a part of the training data (members), (2) we have access to non-members that are *independent and identically distributed* (IID) with members. When we fail to satisfy (2) the methods would collapse to dataset detection [32]. Moreover, without satisfying (1) we cannot run MIA and DI at all.

While a *methodologically correct* evaluation of the cutting-edge models is out of our reach, we aim to provide more insight into text-to-images IARs, and see how much they leak. To this end, we run our attacks on VAR-CLIP [261], a VAR-$d16$ model trained on a captioned ImageNet-1k. Our results in table D.3.1 show that this model leaks significantly more data than its class-to-image counterpart of the same size. Moreover, the leakage is on a level similar to VAR-$d20$'s—a model of double the size of VAR-CLIP. We argue that the increased leakage stems from the model overfitting more to the conditioning information, which is richer for textual data than for the class labels.

Table D.3.1. Leakage of VAR-CLIP compared to class-conditional VARs. We observe increased privacy leakage over class-conditioned models, expressed by a stronger performance of our attacks.

Model	TPR@FPR=1%	P (DI)
VAR-CLIP	6.30	60
VAR-$d16$	2.18	200
VAR-$d20$	5.92	40

D.3.2. On the Impossibility of a Fully Standardized Experimental Setup Between the Models

In the ideal scenario we are able to isolate only the factors inherent to the models' architecture, and consequently, are able to draw insights which design choices lead to what privacy risks. We would call such setup *standardized*, meaning that the models are *almost identical*, and differ only in factors we want to explore (like architecture). However, in reality we deal with too few models, each one being trained differently, which allows only for limited insights.

We note the models vary in the following ways:

1. **Training duration**, expressed by number of data points seen during training, *e.g.,* RAR-B sees $400 \times 1.27\text{M} \approx 0.5\text{B}$ samples. In DMs we evaluate the training duration varies between 0.21B to 1.79B samples seen, whereas IARs are trained with between 0.26B and 0.51B samples.
2. **Training objectives**. DMs minimize eq. (9.3), while IARs— eq. (9.2). Importantly, DMs minimize the expected error *over timesteps and data*, which

necessitates a twice as long training duration for DMs than IARs (on average) to achieve comparable FID.

3. **Model sizes**. IARs benefit from scaling laws [132], and that allows them to be scaled up to sizes greater than DMs, before their performance plateaus. DMs cannot be scaled that well—the performance gains diminish faster with the increase of size. In effect, the biggest IARs we evaluate—VAR-$d30$ and RAR-XXL— are on average 2-3 times bigger than DMs. Since the size of the model impacts its vulnerability to privacy attacks, our analyses do not fully accommodate for that factor.
4. **Two stage architectures**. All models incorporate an encoder-decoder network for training and inference, *e.g.,* VQ-VAE [91]. Importantly, these encoders differ between models. VAR's next-scale prediction paradigm requires training of a specialized encoder that understands how to process residual token maps, used during encoding an image to the sequence of discrete tokens. Moreover, VAR and RAR work with *discrete* tokens, *i.e.,* the encoder-decoder network additionally contains a quantizer module, which translates the continuous latent representations of the images to a 2D integer-only maps.

Unfortunately, these factors directly prohibit a *standardized comparison* of the privacy risks between DMs and IARs. We are not able to fix the training duration for all models—the generation quality of DMs would be significantly subpar than IARs (as DMs require twice the training time of IARs), and thus the results would be unsound. We incorporate the size of the models in figs. D.7.1, 9.1.1 and 9.5.1, however, we acknowledge that the sizes vary between the models, and this limits our ability to fully disentangle this factor from the privacy results.

However, we are able to fix one factor for all the models: utility. We know the models we source are trained to the maximum of the potential each architecture allows, as we utilize models from papers that aim for exactly that—the best performance. We compare models that are the *upper boundary* of what is possible within the inherent limitations and trade-offs each architecture has to offer. We are deeply aware that privacy vs utility is a balancing act: better models tend to be less private. **Thus, our study fixes one of these parameters—utility—to be the highest possible for a given model**, and under that condition we evaluate how much it leaks. We believe our results provide strong empirical evidence that DMs constitute a Pareto optimum when it comes to image generation—they are comparable in FID, while being significantly more private than the novel IAR models.

D.4. Privacy Leakage Under a Unified Attack

We acknowledge that the field of privacy attacks against image generative models like IARs or DMs is constantly evolving. Since our work aims to provide the current empirical insights into differences in privacy leakage between these architectures, we use *the strongest available* attacks to provide an upper boundary on the privacy leakage, following literature on privacy auditing [29, 84].

However, IARs and DMs are two different classes of models. In consequence, the attacks we employ are *tailored* to their inherent properties, and thus the attacks vary. This might raise concerns of the following nature: what if the field progresses and a new, very potent attack is designed for DMs? Will our current empirical results hold, *i.e.,* can we *really* claim IARs leak more privacy than DMs, or is it just the current MIAs against DMs that are less powerful than for IARs?

We believe our insights in section D.2 provide reasons why IARs *inherently* leak more than DMs. To strengthen our results, we perform an *architecture-agnostic*, unified attack against all models—Loss Attack [254].

D.4.1. Loss Attack

Loss Attack is defined as follows: (1) For each sample we perform a forward pass through the model as it would be during the training (2) We compute the model loss (specific to each model) for the samples. (3) We use the losses to perform MIA (as in section D.5.2), and we use the losses to perform Dataset Inference (see section D.5.3).

Loss Attack differs from MIAs against DMs in the following way: instead of fixing the timestep to the most optimal one ($t = 100$ [48]), and averaging the loss over 5 different input noises [48], we sample $t \sim \mathcal{U}[0, 1000]$, and compute the per-sample loss for *a single random noise*.

For MAR, we roll back the modifications to the diffusion module, explained in section D.8. We do not fix the timestep to the most optimal one ($t = 500$), we compute the loss over 5 (default for training), instead of 64 (optimal) input noises, and we sample the masking ratio for each sample following the distribution used during training, instead of fixing it to 0.95—the optimal value.

For VAR and RAR, this attack is identical to the one in table D.9.1 (first row).

Since the DI framework relies on features obtained from different MIAs, we run DI only with the single feature—Loss Attack. We unify DI to be the same for DMs and IARs by removing the scoring function s for DM-specific DI—CDI [83]. In effect, the procedure is *identical* for DMs and IARs.

Table D.4.1. Unified attack results. We employ Loss Attack [254], discarding any model-specific modifications that might strengthen the signal, to ensure a fair comparison between different model classes and architectures. The results strongly support our notion that IARs leak more privacy than DMs.

Model	Architecture	P (DI)	TPR@FPR=1% (MIA)	AUC (MIA)	Accuracy (MIA)
VAR-$d16$	IAR	3000	$1.50_{\pm 0.18}$	$52.35_{\pm 0.40}$	$50.08_{\pm 0.03}$
VAR-$d20$	IAR	1000	$1.67_{\pm 0.20}$	$54.54_{\pm 0.40}$	$50.11_{\pm 0.03}$
VAR-$d24$	IAR	300	$2.19_{\pm 0.20}$	$59.56_{\pm 0.39}$	$50.15_{\pm 0.04}$
VAR-$d30$	IAR	40	$4.95_{\pm 0.40}$	$75.46_{\pm 0.35}$	$50.32_{\pm 0.05}$
MAR-B	IAR	6000	$1.43_{\pm 0.17}$	$51.31_{\pm 0.30}$	$50.48_{\pm 0.16}$
MAR-L	IAR	3000	$1.52_{\pm 0.16}$	$52.35_{\pm 0.30}$	$50.70_{\pm 0.18}$
MAR-H	IAR	2000	$1.61_{\pm 0.17}$	$53.66_{\pm 0.30}$	$51.07_{\pm 0.20}$
RAR-B	IAR	800	$1.77_{\pm 0.25}$	$54.92_{\pm 0.41}$	$50.25_{\pm 0.06}$
RAR-L	IAR	400	$2.10_{\pm 0.27}$	$58.03_{\pm 0.40}$	$50.39_{\pm 0.07}$
RAR-XL	IAR	80	$3.40_{\pm 0.40}$	$65.58_{\pm 0.38}$	$50.81_{\pm 0.10}$
RAR-XXL	IAR	40	$5.73_{\pm 0.52}$	$74.44_{\pm 0.34}$	$51.64_{\pm 0.19}$
LDM	DM	>20000	$1.08_{\pm 0.13}$	$50.13_{\pm 0.05}$	$50.13_{\pm 0.11}$
U-ViT-H/2	DM	>20000	$0.85_{\pm 0.13}$	$50.11_{\pm 0.09}$	$50.07_{\pm 0.18}$
DiT-XL/2	DM	>20000	$0.84_{\pm 0.14}$	$50.09_{\pm 0.05}$	$50.15_{\pm 0.14}$
MDTv1-XL/2	DM	>20000	$0.85_{\pm 0.13}$	$50.05_{\pm 0.05}$	$50.08_{\pm 0.14}$
MDTv2-XL/2	DM	>20000	$0.87_{\pm 0.12}$	$50.14_{\pm 0.05}$	$50.16_{\pm 0.14}$
DiMR-XL/2R	DM	>20000	$0.89_{\pm 0.13}$	$49.55_{\pm 0.06}$	$49.70_{\pm 0.14}$
DiMR-G/2R	DM	>20000	$0.85_{\pm 0.12}$	$49.54_{\pm 0.06}$	$49.69_{\pm 0.13}$
SiT-XL/2	DM	6000	$0.95_{\pm 0.16}$	$48.22_{\pm 0.26}$	$49.97_{\pm 0.09}$

D.4.2. IARs are Empirically More Prone to the Unified Attack Than DMs

Our results in table D.4.1 are consistent with the results achieved with DM- and IAR-specific attacks (tables 9.5.1 and 9.5.3) Empirical data shows IARs are more vulnerable to MIAs and DI. Loss Attack does not yield TPR@FPR=1% greater than random guessing (1%) for DMs, whereas all IARs perform above random guessing. Moreover, with such a weak signal, DI ceases to be successful for DMs, requiring above 20,000 samples (P) to reject the null hypothesis (no significant difference between members and non-members), with one exception: SiT. Conversely, IARs retain their high vulnerability to DI, with the most private IAR—MAR-B—being similarly vulnerable to the least private DM—SiT.

We believe results obtained under the unified attack strengthen our message that current IARs leak more privacy than DMs.

D.5. Additional Background

In the following we provide additional background on Diffusion Models used for comparison to IARs, details on MIAs, and precise definition of the DI procedure, as well as a description of the sampling strategies used by IARs during generation.

D.5.1. Diffusion Models

Table D.5.1. DM details. We report the training details for the DM models used in this work.

	LDM	U-ViT-H/2	DiT-XL/2	MDTv1-XL/2	MDTv2-XL/2	DiMR-XL/2R	DiMR-G/2R	SiT-XL/2
Model parameters	395M	501M	675M	700M	742M	505M	1056M	675M
Training steps	178k	500k	400k	2M	6.5M	1M	1M	7M
Batch size	1200	1024	256	256	256	1024	1024	256
FID	3.60	2.29	2.27	1.79	1.58	1.70	1.63	2.06

We provide a brief overview of DMs used in our experiments. All models are class-conditioned latent DMs trained on the ImageNet dataset at 256×256 resolution. Except for LDM, all models utilize Vision Transformers (ViT) [75] as their diffusion backbones. LDM instead employs the UNet architecture [192], being a prior work. We refer the reader to the original publications for more details about their architectures and training strategies.

LDM (Latent Diffusion Model) by rom [24] first propose running diffusion in a learned latent space rather than in pixel space, using a U-Net as the denoising backbone.

DiT-XL/2 (Diffusion Transformer) by Peebles and Xie [174] replaces the conventional U-Net with a ViT backbone.

U-ViT-H/2 by Bao et al. [40] adopts a ViT-based architecture with skip connections inspired by U-Nets. It treats image patches, class labels, and diffusion timesteps as input tokens in a unified transformer space.

MDTv1-XL and *MDTv2-XL (Masked Diffusion Transformer)* by Gao et al. [103] apply a masked latent modeling strategy during training to enhance contextual learning. The model predicts missing latent tokens, improving training efficiency and sample quality. MDTv2 introduces architectural refinements that lead to further gains in fidelity and performance.

DiMR-XL/2R and *DiMR-G/2R* by Liu et al. [151] propose a multi-resolution diffusion framework that processes features across different spatial scales. This design improves detail preservation and reduces distortions, especially when using large patch sizes. The models also incorporate time-aware normalization to enhance temporal conditioning.

SiT-XL/2 (Scalable Interpolant Transformer) by Ma et al. [156] extends the DiT architecture with an interpolant mechanism that decouples the noise schedule from the model. This allows for greater flexibility in diffusion dynamics without architectural changes.

Besides these models, we additionally evaluate emerging DMs: LFM [65]—a flow-matching model, and DiT-MoE [96]—a mixture-of-experts DM, based on DiT [174]. We do not include these models for the final comparison for three reasons: (1) the released models are significantly smaller (130M parameters each) than all other models, (2) the released models achieve subpar FID scores (4.46 for LFM, unknown FID for DiT-MoE), (3) unknown details of training (number of iterations for DiT-MoE). For completeness, we perform MIA and DI, and report the values in table D.5.2.

Table D.5.2. Results for novel DM architectures. We see the leakage is similar to the rest of DMs.

Model	TPR@FPR=1%	P (DI)
LFM	1.79	2000
DiT-MoE	1.70	2000

D.5.2. Membership Inference Attacks

MIAs attempt to identify whether a given input x, drawn from distribution $\mathcal{X}$, was part of the training dataset $\mathcal{D}_{\text{train}}$ used to train a target model f_θ. We explore several MIA strategies under a gray-box setting, where the adversary has access to the model's loss but no information about its internal parameters or gradients. The goal is to construct an *attack* function $A_{f_\theta} : \mathcal{X} \to \{0,1\}$ that predicts membership.

Threshold-Based attack. Threshold-based attack is a key method of establishing membership status of a sample. It relies on a metric such as Loss [254] to determine membership. An input x is classified as a member if value of the metric falls below a predefined threshold:

$$A_{f_\theta}(x) = \mathbb{1}[\mathcal{M}(f_\theta, x) < \gamma], \tag{D.1}$$

where $\mathcal{M}$ is the metric function, and γ is the threshold.

Min-k% Prob Metric. To address the limitations of predictability in threshold-based attacks, Shi et al. [209] introduced the Min-k% Prob metric. This approach evaluates the least probable $K\%$ of tokens in the input x, conditioned on preceding tokens, where K is a hyperparameter, selected from $\{10, 20, 30, 40, 50\}$. By focusing on less predictable tokens, Min-k% Prob avoids over-reliance on highly predictable parts

of the sequence. Membership is determined by thresholding the average negative log-likelihood of these low-probability tokens:

$$A_{f_\theta}(x) = \mathbb{1}[\text{Min-k\% Prob}(x) < \gamma].$$

The final value is reported for the best K.

Min-k% Prob ++. Min-k% Prob ++ refines the Min-k% Prob method by leveraging the insight that training samples tend to be local maxima in the modeled probability distribution. Instead of simply thresholding token probabilities, Min-k% Prob ++ examines whether a token forms a mode or has relatively high probability compared to other tokens in the vocabulary.

Given an input sequence $x = (x_1, x_2, \ldots, x_T)$ and an autoregressive language model f_θ, the Min-k% Prob ++ score is computed as:

$$\mathcal{S}_{\text{Min-K\%++}}(x) = \frac{1}{|S|} \sum_{t \in S} \frac{\log p(x_t | x_{<t}) - \mu_{x<t}}{\sigma_{x<t}}, \tag{D.2}$$

where S consists of the least probable $K\%$ tokens in x, and $\mu_{x<t}$ and $\sigma_{x<t}$ are the mean and standard deviation of log probabilities across the vocabulary. Membership is determined by thresholding:

$$A_{f_\theta}(x) = \mathbb{1}[\mathcal{S}_{\text{Min-K\%++}}(x) \geq \gamma]. \tag{D.3}$$

Similarly to Min-k% Prob, Min-k% Prob ++ sweeps over $K \in \{10, 20, 30, 40, 50\}$, and the final result is reported for the best hyperparameter K.

zlib Ratio Attacks. A simple baseline attack leverages the compression ratio computed using the *zlib library* [102]. This method compares the model's perplexity with the sequence's entropy, as determined by its zlib-compressed size. The attack is formalized as:

$$A_{f_\theta}(x) = \mathbb{1}\left[\frac{\mathcal{P}_{f_\theta}(x)}{zlib(x)} < \gamma\right].$$

The intuition is that samples from the training set tend to have lower perplexity for the model, while the zlib compression, being model-agnostic, does not exhibit such biases.

CAMIA introduces several context-aware signals to enhance membership inference accuracy. The *slope signal* captures how quickly the per-token loss decreases over time, as members typically exhibit a steeper decline. *Approximate entropy* quantifies the regularity of the loss sequence by measuring the frequency of repeating patterns, while *Lempel-Ziv complexity* captures the diversity of loss fluctuations by

counting unique substrings in the loss trajectory—both of which tend to be higher for non-members. The loss thresholding *Count Below* approach computes the fraction of tokens with losses below a predefined threshold, exploiting the tendency of members to have more low-loss tokens. *Repeated-sequence amplification* measures how much the loss decreases when an input is repeated, as non-members often show stronger loss reductions due to in-context learning.

Surprising Tokens Attack (SURP). SURP detects membership by identifying *surprising tokens*, which are tokens where the model is highly confident in its prediction but assigns a low probability to the actual ground truth token. Seen data tends to be less surprising, meaning the model assigns higher probabilities to these tokens in familiar contexts.

For a given input $x = (x_1, x_2, \ldots, x_T)$, surprising tokens are those where the Shannon entropy is low and the probability of the ground truth token is below a threshold:

$$S = \{t \mid H_t < \epsilon_e, \quad p(x_t|x_{<t}) < \tau_k\}, \tag{D.4}$$

where H_t is the entropy of the model's output at position t, τ_k is the probability of the bottom k%-th token. $k \in \{10, 20, 30, 40, 50\}$, and $\epsilon_e \in \{2, 4, 8, 16\}$ are hyperparameters. The SURP score is the average probability assigned to these surprising tokens:

$$\mathcal{S}_{\text{SURP}}(x) = \frac{1}{|S|} \sum_{t \in S} p(x_t|x_{<t}). \tag{D.5}$$

Membership is determined by thresholding:

$$A_{f_\theta}(x) = \mathbb{1}[\mathcal{S}_{\text{SURP}}(x) \geq \gamma]. \tag{D.6}$$

The SUPR's result for the best combination of k and ϵ_e is selected as the final performance.

D.5.3. Dataset Inference

Scaling IARs to larger datasets raises concerns about the unauthorized use of proprietary or copyrighted data for training. With the growing adoption and increasing scale of IARs, this issue is becoming more pressing. In our work, we use DI to quantify the privacy leakage in IAR models. However, DI can be additionaly used to establish a dispute-resolution framework for resolving illicit use of data collections in model training, ie. *determine if a specific dataset was used to train a IAR*.

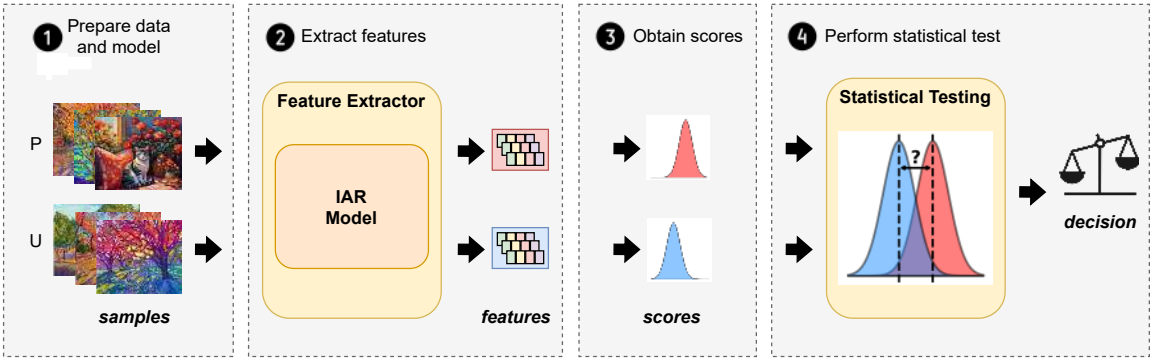


Figure D.5.1. Dataset Inference for IARs Procedural Steps. The process consists of four main steps: ❶ *Data Preparation:* Prepare the data to verify whether the (suspected) member samples **P** were used to train the IAR. The (confirmed) nonmember samples **U**, from the same distribution as **P**, serve as the validation set. ❷ *Feature Extraction:* Run each individual MIA on all inputs from {**P**,**U**} to extract membership features for all data samples. We use our MIAs tailored to IAR models. ❸ *Score Computation:* Normalize the extracted features using MinMaxScaler to scale them into the [0,1] range and compute a membership score for each sample by summing its normalized feature values. ❹ *Statistical Testing:* Apply a statistical t-test to verify whether the scores obtained for the public suspect data points **P** are statistically significantly higher than those for **U**. If so, **P** is marked as being part of the IAR's training set. Otherwise, the test is inconclusive and the IAR's training set is considered independent of **P**.

The framework involves three key roles. First, the *victim* ($\mathcal{V}$) is the content creator who suspects that their proprietary or copyrighted data was used to train a IAR without permission. The victim provides a subset of samples ($\mathcal{P}$) they believe may have been included in the model's training dataset. Second, the *suspect* ($\mathcal{A}$) refers to the IAR provider accused of using the victim's dataset during training. The suspect model (f_θ) is examined to determine whether it demonstrates evidence of having been trained on $\mathcal{P}$. Finally, the *arbiter* acts as a trusted third party, such as a regulatory body or law enforcement agency, tasked with conducting the dataset inference procedure. For instance, consider an artist whose publicly accessible but copyrighted artworks have been used without consent to train a IAR. The artist, acting as the victim ($\mathcal{V}$), provides a small subset of suspected training samples ($\mathcal{P}$). The IAR provider ($\mathcal{A}$) denies any infringement. An arbiter intervenes and obtains gray-box or white-box access to the suspect model. Using DI methodology, the arbiter determines whether the IAR demonstrates statistical evidence of training on $\mathcal{P}$.

D.5.4. Sampling Strategies

The greedy approach selects the token with the highest probability. In the top-k sampling, the highest k token probabilities are retained, while all others are set to zero. The remaining non-zero probabilities are then re-normalized and used

to determine the next token. Notably, when $k = 1$, this method reduces to greedy sampling.

D.6. Model Details

In our experiments, we use a range of models from VAR [223], RAR[256], and MAR [145] architectures, each varying in model size and architecture. The details of these models, including the number of parameters, training epochs, and FID scores, are summarized in table D.6.1. The models were trained on the class-conditioned image generation on the ImageNet dataset [71].

Table D.6.1. Model details. We report the training details for IAR the models used in this work.

	VAR Models				RAR Models				MAR Models		
	VAR-*d*16	VAR-*d*20	VAR-*d*24	VAR-*d*30	RAR-B	RAR-L	RAR-XL	RAR-XXL	MAR-B	MAR-L	MAR-H
Model parameters	310M	600M	1.0B	2.1B	261M	462M	955M	1.5B	208M	478M	942M
Training epochs	200	250	300	350	400	400	400	400	400	400	400
FID	3.55	2.95	2.33	1.92	1.95	1.70	1.50	1.48	2.31	1.78	1.55

D.7. Training and Inference Cost Estimation

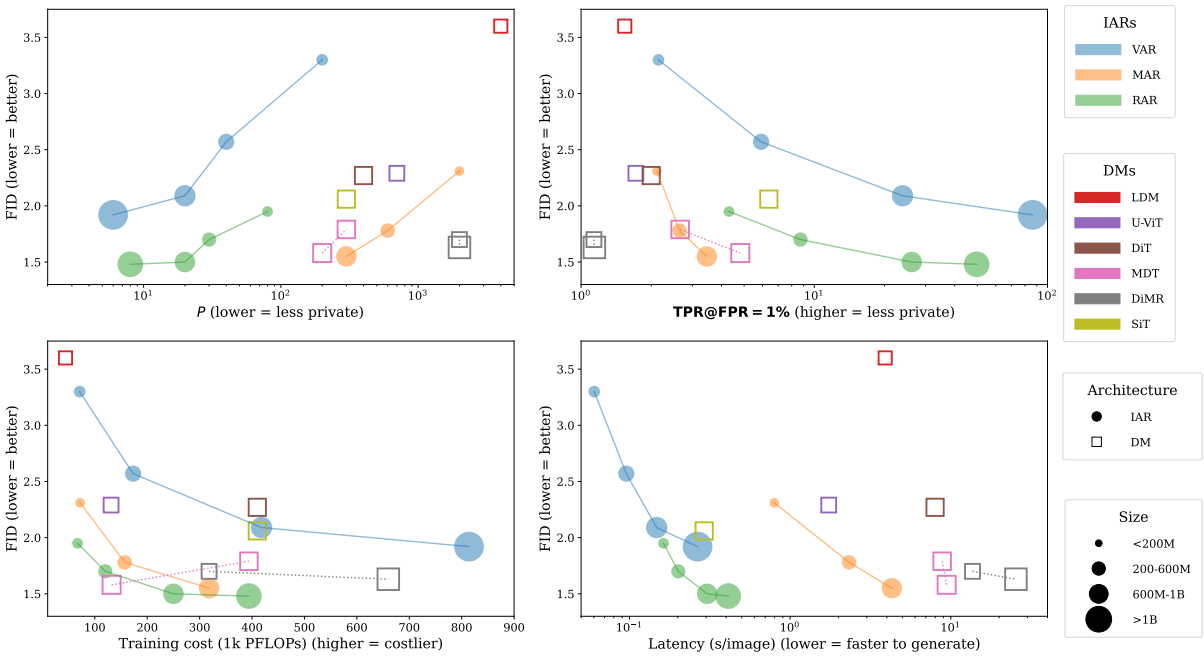


Figure D.7.1. Comprehensive comparison of the trade-offs between IARs and DMs.

Here we describe the comprehensive process of training and generation cost estimation of IARs and DMs, which results in the plot fig. D.7.1. We use *torchprofile* [20] Python library to measure GFLOPs used for generation and training.

In order to compute the training cost, the procedure is as follows. (1) We perform a single forward pass through the model. (2) We multiply the obtained GFLOPs cost by two, to accommodate for the backward pass cost. (3) We multiply the resulting cost of a single forward and backward pass by the amount of training samples passed through the model during training. The amount of samples is based on the numbers reported in the papers for each of the evaluated models. DMs and IARs use a different reporting methodology, with the former reporting training steps and a batch size, and the latter reporting the number of epochs. For the latter, we assume that a full pass through the ImageNet-1k training set is performed, thus we multiply the number of epochs by $1,281,167$.

Time to generate a single sample (referred to as latency) is computed by generating 640 images using code from the original models' repositories. We use the maximum batch size that fits on a single NVIDIA RTX A4000 48GB GPU, to utilize our hardware to the maximum, in order to ensure a fair comparison. For DMs and IARs we follow the settings reported by authors of the respective papers that give the lowest FID score, *i.e.,* we use classifier-free guidance for all the models. For MAR we perform 64 steps of patches sampling. For all DMs but U-ViT we perform 250 steps of denoising, while for U-ViT the reported number is 50, which explains low latency of this model in comparison to others. We acknowledge that, in case of DMs, there are ways to lower the cost of the inference, *e.g.,* by lowering the number of denoising steps. However, we use the default, yet more costly setup for these models, as there is an inherent trade-off between generation quality and cost for DMs, which we want to avoid to make our results sound.

Single generation cost in GFLOPs is computed in a similar fashion. We utilize code provided by the authors of the respective papers for the inference, wrap it using *torchprofile*, and perform a generation of a single sample. Note that here we do not measure time, and we can ignore the parallelism of hardware, as the total cost would stay the same. As we observe in fig. 9.1.1, there is a discrepancy between latency and cost of generation, especially in case of RAR, where we observe an order of magnitude higher generation time than the GFLOPs cost would suggest. This phenomenon originates from the KV-Cache mechanism that is used in case of VAR and RAR during sampling. While the compute cost is lower thanks to the mechanism, the reading operation of the cache mechanism is not effectively parallelized, which results in hardware-incurred latency. We, however, acknowledge that this trade-off might

become more beneficial in cases of low-power edge devices, as the computational power of these devices is more limited than the speed of memory operations.

D.8. MIAs for MAR

Adjusting Binary Mask MAR extends the IAR framework by incorporating masked prediction strategies, where masked tokens are predicted based on the visible ones. This design choice is inspired by Masked Autoencoders [113], where selectively removing and reconstructing parts of the input allows models to learn better representations. Given that MIAs rely on detecting subtle differences in how models process known and unknown data, we hypothesize that adjusting the masking ratio during inference can amplify membership signals. By increasing the masking ratio from 0.86 (the training average) to 0.95, we create conditions where fewer tokens are available to reconstruct the original image, potentially exposing membership information more prominently.

Our experimental results, reported in table D.8.1, confirm that this strategy enhances MIAs' effectiveness. Specifically, TPR@FPR=1% for MAR-H increases from 2.18 to 2.88 (+0.70), and MAR-L sees an improvement from 1.89 to 2.25 (+0.36), demonstrating that a higher masking ratio strengthens membership signals. Notably, setting the mask ratio too high (e.g., 0.99) leads to a slight drop in MIA performance, suggesting a balance must be struck between revealing more membership signal and overly degrading the model's ability to generate images effectively.

Table D.8.1. Impact of varying mask ratio on MIAs for MAR. We report TPR@FPR=1%. Higher values indicate stronger membership signals. The best-performing setting is highlighted in bold.

Mask Ratio	MAR-B	MAR-L	MAR-H
0.75	1.64 (-0.05)	1.65 (-0.24)	1.81 (-0.37)
0.80	1.74 (+0.05)	1.76 (-0.13)	1.85 (-0.33)
0.85	1.68 (-0.01)	1.83 (-0.06)	2.00 (-0.18)
0.86 (default)	1.69 (0.00)	1.89 (0.00)	2.18 (0.00)
0.90	1.65 (-0.04)	1.88 (-0.01)	2.22 (+0.05)
0.95	**1.88 (+0.19)**	**2.25 (+0.36)**	**2.88 (+0.70)**
0.99	1.77 (+0.08)	1.86 (-0.03)	2.14 (-0.04)

Fixed Timestep MIAs on DMs have been shown to be most effective when conducted at a specific denoising step t [48]. Since MAR utilizes a small diffusion module for token generation, we hypothesize that targeting MIAs at a fixed timestep t rather than a randomly chosen one can similarly enhance MIA effectiveness. Unlike full-scale diffusion models, where the most discriminative timestep is typically

around $t = 100$, our experiments reveal that for MAR models, the optimal timestep is $t = 500$.

table D.8.2 illustrates the impact of this adjustment. When MIAs are performed at $t = 500$, MAR-H achieves a TPR@FPR=1% of 3.30, improving by +0.42 over the baseline random timestep approach. Similarly, MAR-L and MAR-B also see noticeable gains at this timestep. Notably, selecting timestep $t = 100$ significantly reduces the attack's effectiveness, with a drop of -0.38 for MAR-H.

Table D.8.2. Impact of using a fixed denoising timestep on MIAs for MAR performance. We report TPR@FPR=1%. The most discriminative timestep is highlighted in bold.

Timestep	MAR-B	MAR-L	MAR-H
random	1.88 (0.00)	2.25 (0.00)	2.88 (0.00)
100	1.60 (-0.27)	1.90 (-0.34)	2.50 (-0.38)
500	**1.88 (+0.00)**	**2.41 (+0.17)**	**3.30 (+0.42)**
700	1.85 (-0.03)	2.35 (+0.10)	3.20 (+0.32)
900	1.65 (-0.22)	2.14 (-0.10)	2.97 (+0.09)

Reducing Diffusion Noise Variance The MAR loss function, as defined in eq. (9.3), exhibits certain variance due to its dependence on randomly sampled noise ϵ. During training, MAR uses four different noise samples per image. We hypothesize that increasing the number of noise samples can provide a more stable loss signal, thereby improving the performance of MIAs.

Our results, summarized in table D.8.3, confirm that increasing the number of noise samples has a positive effect on attack performance.

Table D.8.3. Impact of R reducing diffusion noise variance on MIAs for MAR performance. We report TPR@FPR=1%. Obtaining loss for random noise sampled multiple times generally improves attack effectiveness. The best-performing setting is highlighted in bold.

Repeats	MAR-B	MAR-L	MAR-H
4 (default)	1.88 (0.00)	2.41 (0.00)	3.30 (0.00)
8	1.98 (+0.10)	2.59 (+0.18)	3.32 (+0.03)
16	2.01 (+0.13)	2.50 (+0.09)	3.19 (-0.11)
32	2.00 (+0.11)	2.56 (+0.15)	3.35 (+0.06)
64	**2.09 (+0.21)**	**2.61 (+0.20)**	**3.40 (+0.10)**

D.9. Full MIA Results

We report TPR@FPR=1% and AUC for each baseline MIA (table D.9.1, table D.9.2, each improved MIA for IAR (table D.9.3, table D.9.4) and each MIA for DMs (table D.9.5, table D.9.6). Results are randomized over 100 experiments.

Table D.9.1. TPR@FPR=1% for baseline MIAs.

Model	VAR-$d16$	VAR-$d20$	VAR-$d24$	VAR-$d30$	MAR-B	MAR-L	MAR-H	RAR-B	RAR-L	RAR-XL	RAR-XXL
Loss [254]	1.50±0.16	1.67±0.20	2.19±0.21	4.95±0.38	1.42±0.21	1.48±0.19	1.60±0.21	1.76±0.24	2.10±0.27	3.38±0.42	5.70±0.55
Zlib [19]	1.55±0.20	1.74±0.20	2.24±0.24	5.77±0.59	1.41±0.22	1.49±0.21	1.59±0.22	1.91±0.23	2.45±0.26	4.21±0.31	7.52±0.57
Hinge [42]	1.62±0.19	1.72±0.22	2.14±0.23	4.09±0.40	—	—	—	1.81±0.17	1.99±0.19	2.94±0.36	5.16±0.63
Min-K% [209]	1.58±0.16	2.04±0.25	3.22±0.38	12.23±1.13	1.69±0.18	1.89±0.16	2.18±0.23	2.09±0.24	2.86±0.32	5.83±0.52	13.48±0.98
SURP [259]	1.53±0.17	1.70±0.20	2.23±0.23	5.02±0.43	—	—	—	1.84±0.18	2.12±0.30	3.46±0.46	5.82±0.53
Min-K%++ [260]	1.34±0.18	2.21±0.28	3.73±0.34	14.90±0.96	—	—	—	2.36±0.29	3.26±0.30	6.27±0.65	14.63±0.87
CAMIA [51]	1.33±0.18	1.76±0.19	3.07±0.35	16.69±1.16	1.35±0.19	1.38±0.19	1.44±0.23	1.51±0.17	1.78±0.15	1.99±0.34	4.34±0.51

Table D.9.2. AUC for baseline MIAs.

Model	VAR-$d16$	VAR-$d20$	VAR-$d24$	VAR-$d30$	MAR-B	MAR-L	MAR-H	RAR-B	RAR-L	RAR-XL	RAR-XXL
Loss [254]	52.35±0.35	54.53±0.34	59.55±0.35	75.45±0.30	51.92±0.36	53.33±0.36	55.06±0.34	54.92±0.37	58.04±0.37	65.59±0.34	74.45±0.30
Zlib [19]	52.38±0.38	54.59±0.38	59.65±0.37	75.67±0.34	51.91±0.39	53.32±0.39	55.05±0.38	55.27±0.36	58.68±0.35	66.85±0.34	76.17±0.30
Hinge [42]	53.29±0.39	56.83±0.39	62.89±0.39	77.36±0.33	—	—	—	57.07±0.44	61.41±0.44	71.48±0.39	82.14±0.29
Min-K% [209]	53.77±0.40	57.84±0.44	65.49±0.40	83.55±0.30	51.87±0.38	53.29±0.38	55.05±0.38	56.53±0.38	61.21±0.36	71.35±0.32	82.33±0.28
SURP [259]	50.46±0.25	54.54±0.38	59.60±0.40	75.46±0.34	—	—	—	52.21±0.40	58.02±0.42	65.58±0.41	74.50±0.33
Min-K%++ [260]	54.52±0.41	57.93±0.38	65.76±0.38	85.33±0.27	—	—	—	57.82±0.41	62.48±0.38	75.61±0.32	85.16±0.26
CAMIA [51]	52.44±0.44	55.12±0.44	61.37±0.42	80.16±0.34	51.08±0.42	51.96±0.43	53.20±0.38	51.40±0.36	51.83±0.39	59.28±0.39	66.07±0.36

Table D.9.3. TPR@FPR=1% for our improved MIAs for IARs.

Model	VAR-$d16$	VAR-$d20$	VAR-$d24$	VAR-$d30$	MAR-B	MAR-L	MAR-H	RAR-B	RAR-L	RAR-XL	RAR-XXL
Loss [254]	2.16±0.26	5.95±0.54	24.03±1.91	86.38±0.92	1.54±0.22	1.81±0.21	2.26±0.26	2.86±0.20	5.50±0.39	16.58±0.97	40.76±1.87
Zlib [19]	1.75±0.17	4.87±0.41	20.37±1.19	83.99±0.87	1.51±0.21	1.80±0.23	2.23±0.27	2.52±0.31	4.56±0.39	13.91±1.02	41.03±1.96
Hinge [42]	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	—	—	—	2.50±0.20	4.34±0.39	10.59±0.88	20.23±1.85
Min-K% [209]	0.05±0.02	0.06±0.02	0.14±0.04	1.63±0.13	2.09±0.23	2.6±0.28	3.40±0.30	4.30±0.33	8.66±0.79	26.14±1.22	49.80±2.15
Min-K%++ [260]	0.39±0.06	1.40±0.11	4.88±0.20	37.90±0.44	—	—	—	4.19±0.40	8.24±0.66	23.04±1.14	43.67±2.32
CAMIA [51]	1.83±0.25	5.46±0.52	20.92±1.14	72.77±1.04	1.00±0.17	0.97±0.13	1.06±0.15	1.63±0.21	2.60±0.27	6.77±0.47	17.85±1.20

Table D.9.4. AUC for our improved MIAs for IARs.

Model	VAR-$d16$	VAR-$d20$	VAR-$d24$	VAR-$d30$	MAR-B	MAR-L	MAR-H	RAR-B	RAR-L	RAR-XL	RAR-XXL
Loss [254]	61.73±0.33	76.26±0.30	92.20±0.15	98.95±0.05	52.25±0.42	54.60±0.41	57.35±0.40	65.61±0.35	75.83±0.32	89.64±0.21	96.17±0.12
Zlib [19]	57.91±0.39	70.86±0.33	88.69±0.24	98.51±0.07	52.23±0.39	54.57±0.39	57.33±0.39	62.22±0.42	72.19±0.37	87.51±0.22	95.46±0.13
Hinge [42]	52.67±0.36	56.11±0.36	62.48±0.36	74.63±0.30	—	—	—	59.66±0.39	68.09±0.35	81.56±0.29	90.62±0.21
Min-K% [209]	59.78±0.34	70.43±0.34	83.10±0.25	90.16±0.27	53.31±0.40	56.34±0.39	59.98±0.38	66.81±0.38	78.73±0.32	91.36±0.20	96.97±0.10
Min-K%++ [260]	57.10±0.30	65.44±0.29	78.74±0.25	93.18±0.16	—	—	—	65.20±0.36	75.37±0.34	88.29±0.23	95.84±0.14
CAMIA [51]	56.37±0.38	68.18±0.31	84.83±0.24	96.95±0.09	50.86±0.41	51.15±0.41	51.75±0.41	57.95±0.40	63.17±0.43	70.43±0.39	83.55±0.31

Table D.9.5. TPR@FPR=1% of MIAs for DMs.

	LDM	U-ViT-H/2	DiT-XL/2	MDTv1-XL/2	MDTv2-XL/2	DiMR-XL/2R	DiMR-G/2R	SiT-XL/2
Denoising Loss [48]	1.35±0.14	1.30±0.17	1.42±0.17	1.55±0.18	1.64±0.17	0.91±0.15	0.88±0.15	1.02±0.13
SecMI$_{stat}$ [79]	1.30±0.20	1.31±0.19	1.49±0.22	1.35±0.17	1.52±0.22	1.15±0.21	1.05±0.15	0.00±0.00
PIA [137]	1.25±0.16	1.25±0.19	1.59±0.20	1.72±0.20	2.07±0.24	1.07±0.11	1.09±0.12	1.14±0.14
PIAN [137]	1.03±0.14	1.17±0.16	0.92±0.12	1.22±0.15	1.50±0.20	1.04±0.13	1.01±0.12	1.09±0.14
GM [83]	1.25±0.17	1.26±0.17	1.34±0.17	1.18±0.16	1.47±0.19	1.13±0.15	1.16±0.16	1.38±0.18
ML [83]	1.41±0.16	1.36±0.20	1.50±0.18	1.70±0.16	1.98±0.26	1.01±0.15	1.10±0.14	1.14±0.12
CLiD [258]	1.55±0.19	1.75±0.22	2.08±0.28	2.72±0.39	4.91±0.44	0.96±0.14	0.90±0.13	6.38±0.64

Table D.9.6. AUC for MIAs for DMs.

	LDM	U-ViT-H/2	DiT-XL/2	MDTv1-XL/2	MDTv2-XL/2	DiMR-XL/2R	DiMR-G/2R	SiT-XL/2
Denoising Loss [48]	50.53±0.41	50.36±0.42	51.77±0.43	51.25±0.37	51.65±0.37	46.25±0.40	46.01±0.40	47.25±0.34
SecMI$_{stat}$ [79]	49.84±0.44	53.15±0.43	55.15±0.46	54.44±0.38	56.80±0.36	48.73±0.45	48.73±0.44	50.00±0.00
PIA [137]	48.97±0.43	51.77±0.44	53.18±0.42	52.60±0.44	54.68±0.45	47.31±0.42	47.16±0.41	49.13±0.44
PIAN [137]	49.56±0.43	50.99±0.46	50.14±0.43	49.96±0.42	51.52±0.38	49.85±0.41	49.79±0.43	50.17±0.37
GM [83]	51.51±0.40	51.19±0.42	50.46±0.46	50.72±0.39	48.85±0.37	45.97±0.45	45.86±0.45	50.94±0.38
ML [83]	50.36±0.41	51.16±0.41	52.53±0.45	50.42±0.19	54.65±0.38	46.26±0.38	49.37±0.41	49.83±0.17
CLiD [258]	52.50±0.39	54.27±0.41	56.16±0.41	57.43±0.41	62.54±0.40	46.20±0.38	45.95±0.41	78.65±0.30

D.10. Full DI Results

We report the outcome of DI for DMs in table D.10.1. As an additional observation, we note that contrary to DI for IARs, shifting from the classifier to an alternative feature aggregation increases the number of samples needed to reject H_0. This suggests, that the linear classifier remains necessary for DMs.

Table D.10.1. DI for DMs. We report the minimal number of samples needed to successfully reject H_0.

	LDM	U-ViT-H/2	DiT-XL/2	MDTv1-XL/2	MDTv2-XL/2	DiMR-XL/2R	DiMR-G/2R	SiT-XL/2
DI for DM	4000	700	400	300	200	2000	200	300
No Classifier	5000	4000	3000	600	400	2000	2000	500

D.11. Mitigation Strategy

In this section we detail our privacy risk mitigation strategy.

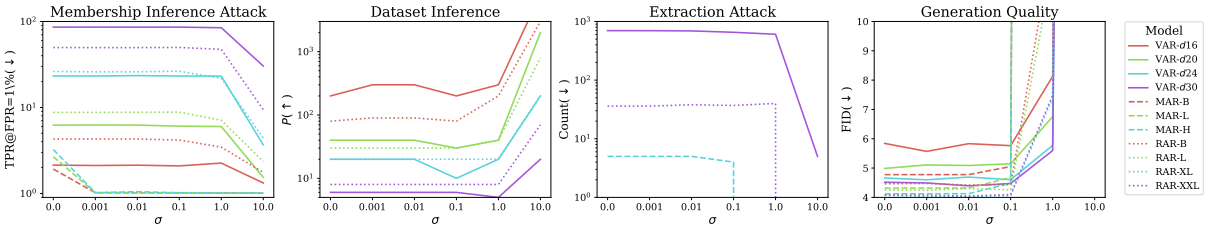


Figure D.11.1. Privacy-utility trade-off of our mitigation strategy. We show that successfully defending VAR and RAR against MIA and DI requires adding noise that severely harms the performance. Interestingly, we are able to limit the extent of memorization for VAR, and fully defend MAR against MIA and DI.

D.11.1. Method

Given an input sample x, we perturb the output of the IAR according to a noise scale σ, which we can adjust to balance privacy-utility trade-off. During inference,

we add noise sampled from $\mathcal{N}(0,\sigma)$ to the output. For VAR and RAR, we add it to the logits, and for MAR we add them to the sampled continuous tokens.

We measure privacy leakage with our methods from section 9.5. Specifically, we perform MIAs, DI, and the extraction attack. To quantify utility, we generate 10,000 images from the IARs, and compute FID [115] between generations and the validation set. Lower FID means better quality of the generations.

D.11.2. Results

Our results in fig. D.11.1 show that we can effectively lower the privacy loss by applying our mitigation strategy, however, this comes at a cost of significantly decreased utility, as highlighted by substantially increasing FID score.

We are able to lower the MIAs success by more than half (Fig. D.11.1, left), with the biggest relative drop observed for RAR-XL, for which the TPR@FPR=1% drops from 26% to **4.4%**. Moreover, *all* MAR models become immune to MIAs after noising their tokens, as TPR@FPR=1% drops to 1% (random guessing) with $\sigma = 0.001$. When we apply our defense to DI (Fig. D.11.1, second from the left), we have to increase P, the minimum number required to perform a successful DI attack, by an order of magnitude, with the biggest relative difference for the smallest models: VAR-16, and RAR-B, with an increase from 80 to 3000, and 200 to 8000, respectively. Such an increase means that the models are harder to attack with DI, *i.e.,* their privacy protection is boosted. Similarly to MIA, DI stops working for MAR models immediately.

Our method achieves limited success in mitigating extraction (Fig. D.11.1, third from the left). We are lowering the success of extraction attack only when adding significant amount of noise. However, for VAR-$d30$, which exhibits the biggest memorization, with $\sigma = 1.0$ we successfully protect **93** out of 698 samples from being extracted without significantly harming the utility. Our method, similarly to all defenses, suffers from lowered performance (Fig. D.11.1, right), as signal-to-noise ratio during generation gets worse when σ increases.

D.11.3. Discussion

We show that we can mitigate privacy risks by adding noise to the outputs of IARs, at a cost of utility. Notably, all MARs become *fully* immune to MIAs and DI with noise scale as small as 0.001. This result supports previous insights from section 9.5, in which we show that MARs are significantly less prone to privacy risks than VARs and RARs. We argue that logits leak significantly more information than continuous

tokens, and thus, adding noise to the latter yields significantly higher protection, at a lower performance cost.

We acknowledge that our privacy leakage defense is a heuristic, and more theoretically sound approaches should be explored, *e.g.,* in the domain of Differential Privacy [84]. To the best of our knowledge, we make the first step towards private IARs.

D.12. More About Memorization

In this section we provide an extended analysis of memorization phenomenon in IARs. We show more examples of memorized images, highlight the relation between the prefix length i and the number of extracted samples, and shed more light on our efficient extraction method, described in section 9.5.3.

D.12.1. More Memorized Images

In fig. D.12.4 we show a non-cherry-picked set of images memorized by IARs. In fig. D.12.3 we show an example of an image memorized verbatim by VAR-$d30$ **without any prefix**, *i.e.,* only from the class label token. In fig. D.12.2 we show an image that has been memorized by both VAR-$d30$ and RAR-XXL.

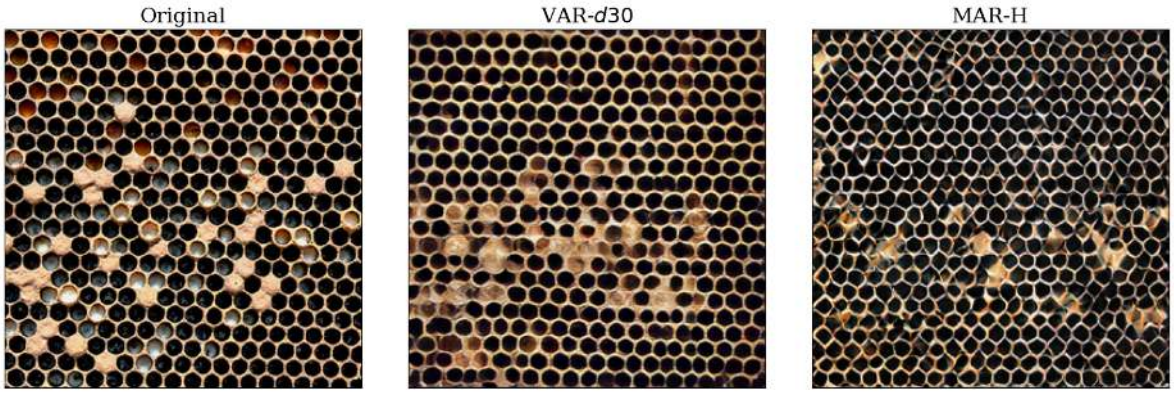


Figure D.12.1. An image extracted from both VAR-$d30$, and MAR-H.

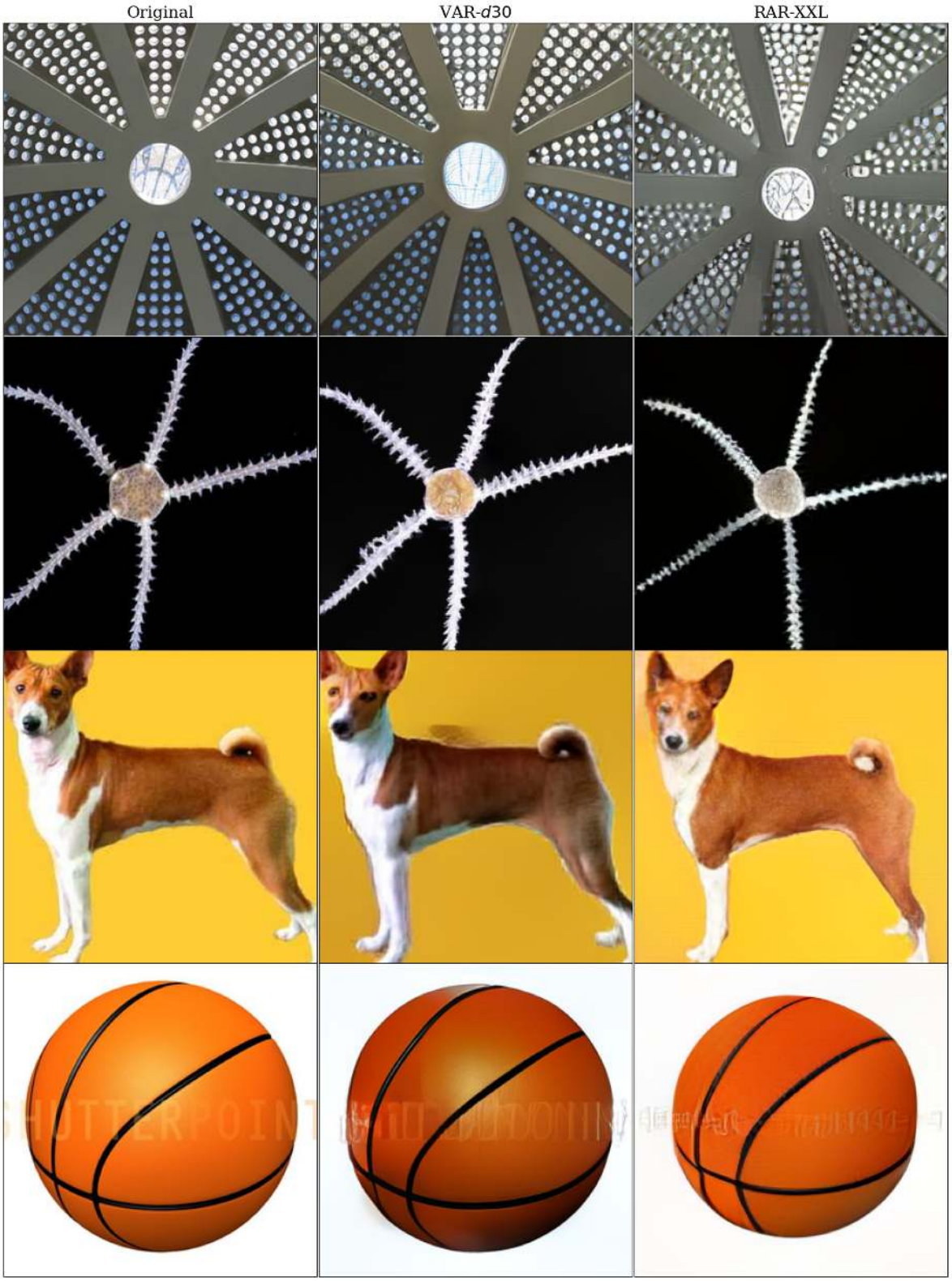


Figure D.12.2. Images extracted from both VAR-*d*30, and RAR-XXL.

Figure D.12.3. Image extracted from VAR-*d*30 *without prefix*. (Left) memorized image, (right) generated image.

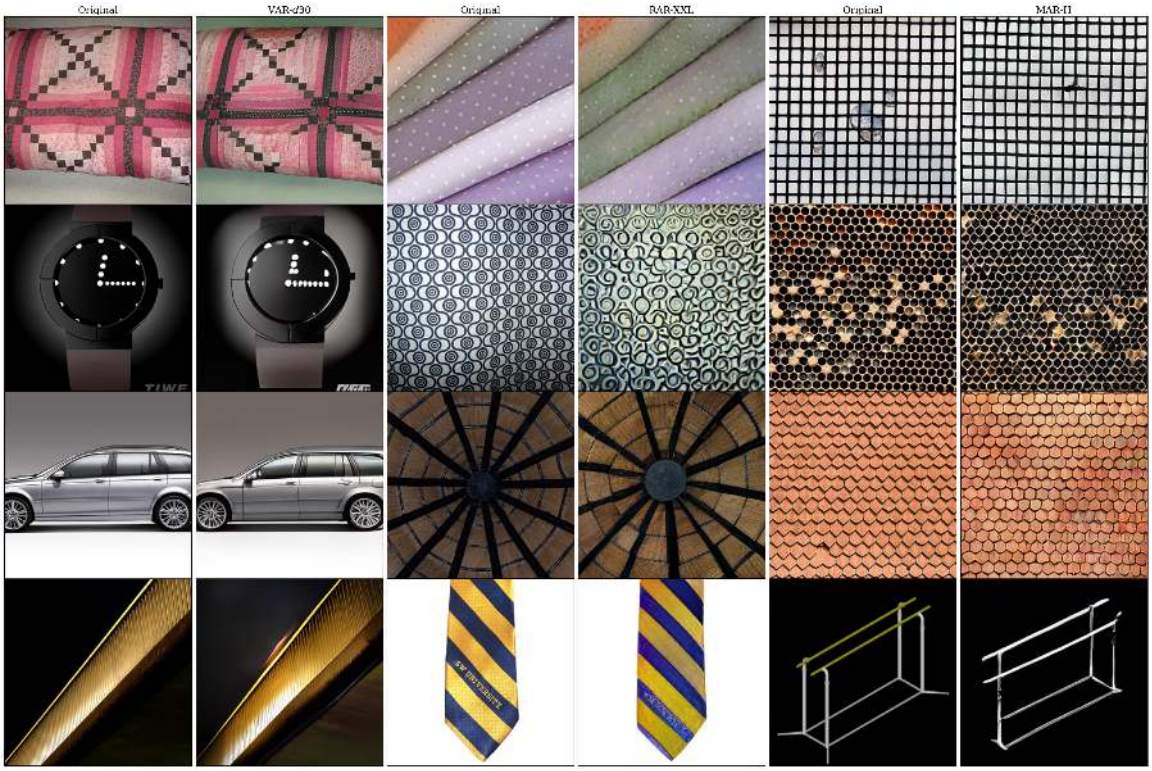


Figure D.12.4. Non-cherry-picked extracted images. Odd columns from the left correspond to the original image, even to extracted. From left, the images are for VAR-*d*30, RAR-XXL, and MAR-H.

D.12.2. Prefix Length vs. Number of Extracted Images

We analyze the effect of the prefix length on the number of extracted samples. As our method leverages conditioning on a part of the input sequence, in fig. D.12.5 we show an increase of extraction success with the increase in the length of the prefix. Notably, we start experiencing false-positives once the prefix length surpasses 30 for VAR-$d30$ and RAR-XXL, and 5 for MAR-H. In effect, the results in table 9.5.4 provide an upper bound of the success of our extraction method.

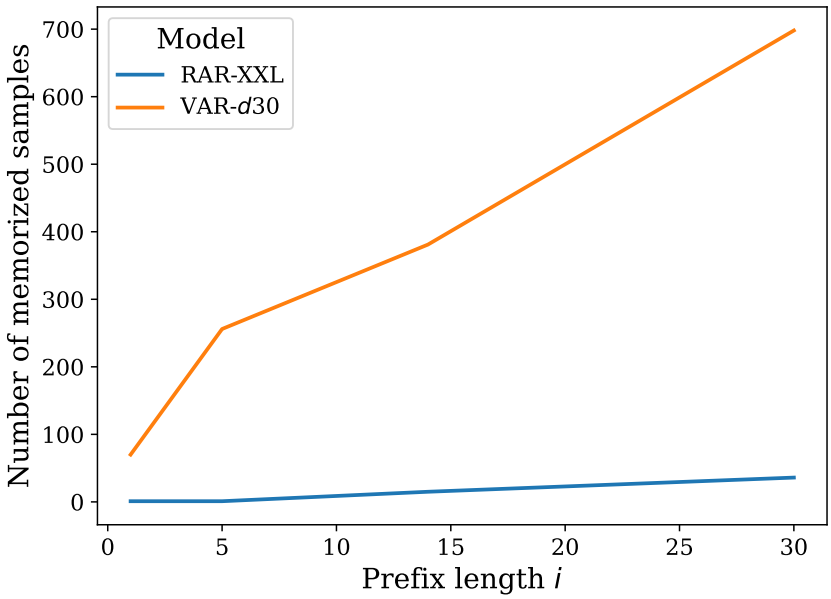


Figure D.12.5. Prefix length and the number of extracted samples. We show that with an increase of the prefix length, the success of our extraction method increases.

Table D.12.1. Prefix length i for our data extraction attack. We note that appending longer sequences leads to false positives, *i.e.,* the IARs start to generate images from the validation set.

Model	VAR-$d30$	MAR-H	RAR-XXL
Prefix length i	30	5	30

D.12.3. Approximate distance vs. SSCD Score

In this section we underscore the effectiveness of our filtering approach. fig. D.12.6 shows that the distances we design for the candidate selection process indeed correlates with the SSCD score. By focusing only on the top-5 samples for each class we effectively narrow our search to just 0.5% of the training set, significantly speeding up the whole process.

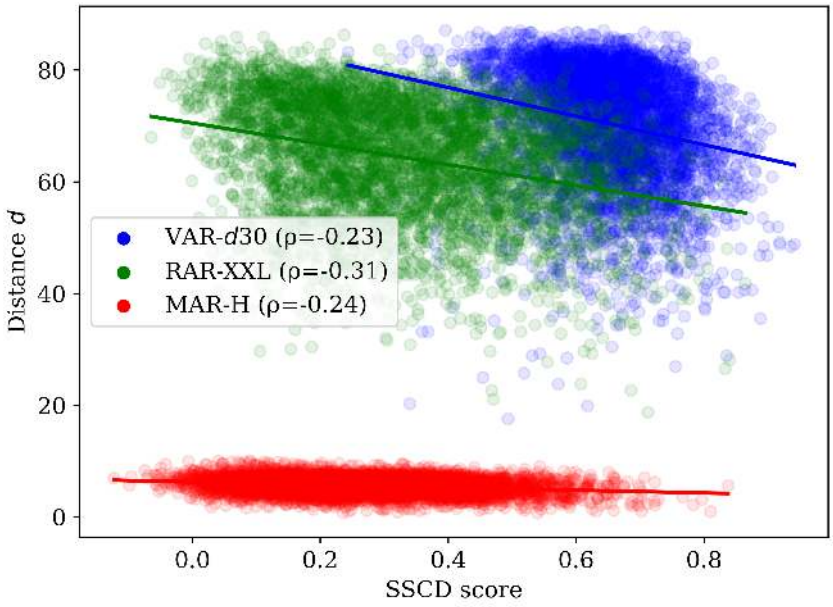


Figure D.12.6. Distance function d and the SSCD score. We show that d correlates with the final memorization score. This result makes our candidate selection process sound, and reduces the cost of extracting memorized samples.