Analyzing Threats of Large-Scale Machine Learning Systems

by

Nils Lukas

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Alina Oprea Associate Professor, Khoury College of Computer Sciences Northeastern University
Supervisor(s):	Florian Kerschbaum Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	N. Asokan Professor, David R. Cheriton School of Computer Science University of Waterloo
	Yaoliang Yu Associate Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Arie Gurfinkel Professor, Department of Electrical and Computer Engineering University of Waterloo

Author's Declaration

This thesis consists of material, all of which I authored or co-authored: see the Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This dissertation includes first-authored and peer-reviewed work that has appeared in conference proceedings published by the Institute of Electrical and Electronics Engineers (IEEE), the Advanced Computing Systems Association (USENIX), and the International Conference on Learning Representations (ICLR).

The following is a list that serves as a declaration of the works included in this dissertation. I expand and revise material from the original publications.

Portions of Chapter 3
Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2023. Analyzing Leakage of Personally Identifiable Information in Language Models. Published in the *44th Proceedings of the 2023 IEEE Symposium on Security and Privacy*.

Portions of Chapter 4
Nils Lukas, Edward Jiang, Xinda Li and Florian Kerschbaum. 2022. SoK: How Robust is Deep Neural Network Image Classification Watermarking? Published in the *43rd Proceedings of the 2022 IEEE Symposium on Security and Privacy*.

Portions of Chapter 5
Nils Lukas, Yuxuan Zhang and Florian Kerschbaum. 2021. Deep Neural Network Fingerprinting with Conferrable Adversarial Examples. Published in the *9th Proceedings of the 2021 International Conference on Learning Representations*.

Portions of Chapter 6
Nils Lukas and Florian Kerschbaum. 2023. PTW: Pivotal Tuning Watermarking for Pre-Trained Image Generators. Published in the *32nd USENIX Security Symposium*.

Portions of Chapter 7
Nils Lukas, Abdulrahman Diaa, Lucas Fenaux and Florian Kerschbaum. 2023. Leveraging Optimization for Adaptive Attacks on Image Watermarks. Published in the 12th Proceedings of the 2024 International Conference on Learning Representations.

Abstract

Large-scale machine learning systems such as ChatGPT rapidly transform how we interact with and trust digital media. However, the emergence of such a powerful technology faces a dual-use dilemma. While it can have many positive societal impacts in providing equitable access to information, ML systems can also be misused by untrustworthy entities to cause intentional harm. For example, a system could unintentionally disclose private information about its training data and jeopardize the privacy of individuals in the training data. The system's generated content could also be misused for unethical purposes, such as eroding trust in digital media by misrepresenting generating content as authentic. Providing untrustworthy users with these new capabilities could amplify potential negative consequences emerging through this technology, such as a proliferation of deep fakes or disinformation. I analyze these threats from two perspectives: (i) Data leakage, when the model cannot be trusted because it has memorized private information during training, and (ii) Misuse when users cannot be trusted to use the system for its intended purposes. This thesis presents five projects to assess these risks to the privacy and security of ML systems and evaluates the reliability of known countermeasures. To do so, I assess the privacy risks of extracting Personally Identifiable Information from language models trained with differential privacy. As a method of controlling unintended use, I study the effectiveness and robustness of fingerprinting and watermarking methods to detect the provenance of models and their generated content.

Acknowledgements

Florian, I owe an immense thanks to you. You were always there to help me navigate challenges and allowed me to grow as a researcher. I feel incredibly fortunate to have had such a welcoming, supportive, and knowledgeable advisor like you by my side.

Thank you to my committee members, Alina Oprea, Yaoliang Yu, N. Asokan, and Arie Gurfinkel; your expertise and insightful questions made my defense an enriching and memorable experience.

Glaucia and Damien, your presence since day one has been invaluable. Thank you for providing me with your companionship, countless coffee breaks, and moments of both celebration and frustration. You have made this journey more than just about research.

Everyone in the CrySP lab, thank you for making this experience unique and enjoyable. I truly loved the collaborative spirit in the lab, the shared lunches, and the opportunities to brainstorm ideas. It would not have been the same without you all.

An meine Familie, Marijke, Frank, Sophie, Marlene, Pascal und meine Großeltern Mimi, Papou, Edith und Dieter geht mein unendlicher Dank. Ihr habt mir ein Zuhause gegeben, egal wo auf der Welt ich war. Danke, dass ich stets auf eure Unterstützung bauen konnte.

Finally, to the love of my life, Stéphanie Maaz, for your endless support and continued encouragement. You are my inspiration and motivation. I am incredibly thankful to have you by my side, venturing on this journey and all the ones to come, hand in hand.

Dedication

To my wife, Stéphanie Maaz, and my grandfather, Ulrich Karrenberg.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large machine learning (ML) systems, such as ChatGPT, fundamentally shape how we interact with and trust digital media. Training ML models can require vast resources in terms of computation and data (collection, curation, and validation), which means that there are few model providers and many users. If used responsibly, providing many with access to ML models can have numerous positive societal impacts in areas such as education [9], healthcare [71, 106] or scientific research [72, 40]. However, the emergence of such a powerful technology also faces a dual-use dilemma. A few untrustworthy users could attempt to use the provided model for unintended purposes and amplify its potential negative impact. For example, the ability of ML systems to process large datasets can provide users with access to information. However, if the model lacks privacy, it could inadvertently disclose sensitive information from the training data to unauthorized users [27]. Privacy concerns have already led to a temporary ban on ChatGPT in Italy [154] and at corporations such as Apple and Samsung [80, 221] over fears of leaking personal or proprietary information. Attacks have been demonstrated to extract at least 1% of the training data [27]. The use of ML systems in industries like healthcare [115] or finance [203] is restricted due to a lack of privacy. Instead of attacking the dataset, users could also use the model for unintended purposes that violate fair use or are unethical. For example, the ability to generate realistic content can also be used to create online spam [12, 206], disinformation [12], or deceptive personalized attacks [64]. The single largest reported online fraud in 2021 involving voice deepfakes already caused US$35 million in damages [15]. This thesis analyzes three risks an ML model provider faces with untrustworthy users: (i) data privacy, when access to the model reveals sensitive information about its training data [150], (ii) model stealing, where a usage agreement limits unauthorized redistribution [146, 148] and (iii) model misuse, where the model's capabilities are used unethically to manipulate or deceive others [145, 149].

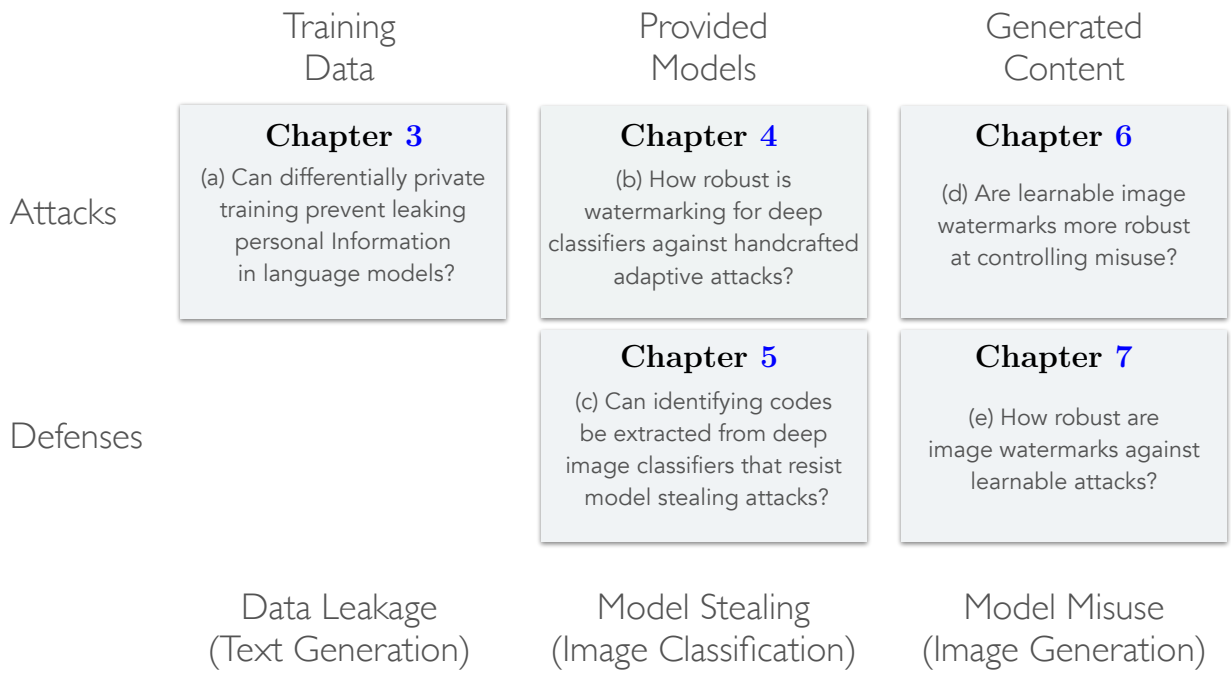


Figure 1.1: Threats when deploying ML models to (i) prevent leaking private information, (ii) deter model stealing, and (iii) control misuse of the model's generated content.

Problem Statement. Responsible ML model providers must integrate privacy and security mechanisms to control these threats in compliance with the "AI Executive Order" [65] and the "EU AI Act" [60]. Defenses have two objectives: (i) they must preserve the model's capabilities for trustworthy users and (ii) reliably control potential threats from untrustworthy users. However, assessing a defense's reliability can be notoriously difficult due to the complexity of ML models. Defenses with certifiable guarantees are provably secure within their threat model but often substantially degrade the model's utility or assume unrealistic threat models. Empirical defenses better preserve utility but can lack reliability when stronger attacks that were previously unknown are discovered. This raises the following research question that I will investigate in this thesis.

How can security mechanisms be designed and tested to control unintended use of a provided model in the presence of untrustworthy users?

This thesis focuses on advancing the theory and practice of designing defenses and testing their reliability when providing ML models to many users where a few are untrustworthy.

1.1 Contributions

To answer this question, I explore several key research questions. These questions are connected, as findings from one project often inform the next. Below are the main questions investigated in this thesis.

(a) *Can differential private training prevent leaking personal data in language models?*

(b) *How robust is watermarking for deep image classifiers against handcrafted attacks?*

(c) *Do deep classifiers have identifying codes that enable the detection of stolen models?*

(d) *Are learnable image watermarking methods more robust than handcrafted methods at controlling misuse with generative machine learning models?*

(e) *How robust are image watermarks against adaptive, learnable attacks?*

Below, I summarize the steps I took to address these research questions.

1. To answer (a), I fine-tune language models with differential privacy on datasets that contain personally identifiable information (PII). Then, I design three attacks to measure the leakage of PII empirically.

2. To answer (b), I systematically study on the robustness of existing watermarking methods for image classifiers. I propose taxonomies and tests that measure a method's robustness and handcraft adaptive attacks.

3. Results from (c) show that an attacker can handcraft *adaptive* attacks to evade detection. I formulate fingerprinting as an objective function and show that robust fingerprints exist against known attacks. The fingerprint consists of a new subclass of transferable adversarial examples that uniquely identify a classifier and its surrogates.

4. (b) and (c) investigate black-box watermarking to deter model stealing, but misuse with deepfakes operates under a threat model I term as *no-box*. I improve the robustness of watermarks in (d) by designing a *learnable* watermark that can be optimized efficiently and show its robustness against state-of-the-art attacks.

5. From (d), I found that the ability to optimize can enhance a watermark's effectiveness and robustness. This motivated (e) the design of an adaptive attack that can be optimized against a watermarking method when only the algorithm is known. I show that such attacks break all existing watermarks, and I believe they are a good method of assessing a watermark's robustness in the future.

1.2 Overview

Five projects explore three threats when providing ML systems to untrustworthy users. Chapter 2 summarizes background on machine learning, watermarking, and differential privacy.

- **Privacy**: Chapter 3 presents the threats related to leaking Personally Identifiable Information (PII) with fine-tuned language models, proposes attacks to measure leakage, and shows that existing defenses can limit but not prevent PII leakage.
- **Model Stealing**: Chapter 5 introduces the threat of *model stealing* and proposes a fingerprinting method for deep neural network classifiers that withstands model extraction attacks. I propose a new subclass of transferable adversarial examples, called *conferrable* examples, and generate conferrable examples for fingerprinting. In Chapter 4, I systematically investigate the reliability of watermarking to detect stolen models against adaptive attackers who design specific attacks against known defenses.
- **Model Misuse**: Chapter 6 describes a *learnable* watermarking method for pre-trained deep image generators with higher efficiency and robustness than existing methods. Chapter 7 proposes a framework to test the robustness of image watermarks against learnable attacks that require only the watermarking method's algorithmic description.

Chapter 8 summarizes the contributions from this thesis and outlines directions for future work.

Chapter 2

Background

This section provides an overview of training machine learning models for tasks such as image classification, image generation, and causal language modeling. Chapter 3 uses language models, Chapters 4 and 5 use deep image classifiers, and Chapters 6 and 7 use deep image generators. I review key concepts in differential privacy, including membership inference [244], and introduce watermarking and fingerprinting.

2.1 Machine Learning

In this thesis, I study machine learning (ML) models and analyze threats posed by these models when deployed to potentially untrusted users as part of an ML system.

Definition 1 (Machine Learning System)**.** *An integrated framework that includes data processing, model training, evaluation, deployment infrastructure, and security and privacy mechanisms to facilitate the deployment of machine learning models to its users.*

2.1.1 Image Classification

Deep image classifiers are trained using a set of labeled images, typically denoted as $(x, y) \in \mathcal{D} \times \mathcal{Y}$, where x represents an image and y is its corresponding label. By $\mathcal{O} : \mathcal{D} \to \mathcal{Y}$, we denote an oracle labeling function that returns the ground-truth label of an image. The goal of training deep image classifiers involves fine-tuning the parameters θ of the classifier f_θ to imitate the ground-truth labeling function on unseen data accurately.

Loss Function. The training process involves minimizing a loss function, $\mathcal{L}(f_\theta(x), y)$, which measures the difference between the predicted and actual labels. A common choice is the cross-entropy loss.

$$\mathcal{L}(f_\theta(x), y) = -\sum_{k=1}^{K} y_k \log(f_\theta(x_k)) \tag{2.1}$$

Optimization The classifier's parameters, θ, can be optimized using Stochastic Gradient Descent (SGD). The update rule in SGD is:

$$\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \mathcal{L}(f_\theta(x), y) \tag{2.2}$$

where α is the learning rate. Training occurs in epochs, each involving a pass through the entire dataset. For efficiency, the data is divided into batches, and θ is updated after computing the loss for each batch. After training, we measure the accuracy of a classifier on an unseen test set.

2.1.2 Image Generation

We summarize two types of image generation models: Generative Adversarial Networks used in Chapter 6 and Latent Diffusion Models used in Chapter 7.

Generative Adversarial Network

Generative Adversarial Network (GANs) [75] define a generator $G : \mathcal{Z} \to \mathcal{D}$ that maps from a latent space $\mathcal{Z}$ to images $\mathcal{D}$ and a discriminator $F : \mathcal{D} \to \{0, 1\}$ that maps images to binary labels. The labels represent *real* and *fake* images. Let $D \subseteq \mathcal{D}$ be an image dataset and let $\theta_{\mathrm{F}}, \theta_{\mathrm{G}}$ be parameters for a discriminator and generator. The unsaturated logistic loss for GANs is written as follows.

$$\mathcal{L}_{GAN} = \mathop{\mathbb{E}}_{x \sim D}[\log F(\theta_F, x)] + \mathop{\mathbb{E}}_{z \sim \mathcal{N}(0, I^d)}[\log(1 - F(\theta_F, G(\theta_G, z))] \tag{2.3}$$

During training, the discriminator learns to classify real and fake images, and the generator learns to fool the discriminator.

StyleGAN [112]. StyleGAN is a specific GAN architecture that introduces a mapper $f : \mathcal{Z} \to \mathcal{W}$, which maps latent codes into an intermediate latent space $\mathcal{W}$. This intermediate latent space consists of so-called *styles* that enable fine-grained control over the synthesized

image. Since its inception, the basic StyleGAN [112] has been revised many times leading to the development of StyleGAN2 [113], StyleGAN3 [114] and recently StyleGAN-XL [198]. These generators have achieved state-of-the-art[1] performance on many image generation datasets, including ImageNet [47].

Latent Diffusion Models

Latent Diffusion Models (LDMs) are state-of-the-art generative models for image synthesis [190]. Compared to Diffusion Models [209], LDMs operate in a latent space using fixed, pre-trained autoencoder consisting of an image encoder $\mathfrak{E}$ and a decoder $\mathfrak{D}$. LDMs use a forward and reverse diffusion process across T steps. In the forward pass, real data point x_0 is encoded into a latent point $z_0 = \mathfrak{E}(x_0)$ and is progressively corrupted into noise via Gaussian perturbations.

$$q(z_t|z_{t-1}) = \mathcal{N}\left(z_t; \sqrt{1-\beta_t}z_{t-1}, \beta_t\mathbf{I}\right), \quad t \in \{0, 1, \ldots, T-1\},$$

where β_t is the scheduled variance. In the reverse process, a neural network f_θ guides the denoising, taking z_t and time-step t as inputs to predict z_{t-1} as $f_\theta(x_t, t)$. The model is trained to minimize the mean squared error between the predicted and actual z_{t-1}. The outcome is a latent $\hat{z}_0$ resembling z_0 that can be decoded into $\hat{x}_0 = \mathfrak{D}(z_0)$. Synthesis in LDMs can be conditioned with data from different modalities, such as textual prompts.

2.1.3 Language Modeling

Language models (LMs) learn the conditional probability distribution $\Pr(w_i|w_1, .., w_{i-1})$ over sequences of tokens $w_1, .., w_i$ from a vocabulary $\mathcal{V}$. By applying the chain rule, we can obtain the probability that an LM θ assigns to a sequence $w_1, .., w_n$:

$$\Pr(w_1, .., w_n; \theta) = \prod_{i=1}^{n} \Pr(w_i|w_1, .., w_{i-1}; \theta) \tag{2.4}$$

State-of-the-art LMs are based on the Transformer neural network architecture [230]. During training, one objective is to maximize the negative log-likelihood of the LM predicting the next token in training sentences given a prefix.

Equation (2.4) gives a probability distribution over all tokens in the vocabulary $\mathcal{V}$ and can be represented as a tree with $|\mathcal{V}|$ branches on each level. Text is generated iteratively by

[1] https://paperswithcode.com/dataset/ffhq

traversing the tree using greedy decoding, top-k sampling [62], or a beam search algorithm while conditioning the LM on all preceding tokens. Autoregressive LMs, such as GPT-2, only allow sampling the conditional probability of the next token based on a *prefix*, whereas masked LMs, such as BERT [48] also consider a sample's *suffix* in the query.

Perplexity. The "optimal" solution to the training objective is to memorize each record [27]. This is both intractable and undesirable, as the goal is for LMs to generalize beyond their training dataset. In practice, learning means that only a fraction of the training dataset is memorized [102] and that some memorization is likely necessary to achieve high utility [67]. The model's utility is evaluated by its perplexity on an unseen test set of sentences. A low perplexity implies a high utility on the dataset.

$$\mathrm{PPL}(w_1, .., w_n; \theta) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n} \log \Pr(w_i | w_1, .., w_{i-1}; \theta)\right)$$

2.2 Privacy

2.2.1 Differential Privacy

Differential Privacy (DP) has become a popular notion of privacy. Recall its definition:

Definition 2 (Approximate Differential Privacy [55])**.** *Let* $\varepsilon > 0$ *and* $\delta \in [0, 1]$*. A mechanism* $\mathcal{M} : X \to Y$ *is* (ε, δ)*-differentially private if for any pair of* adjacent *datasets* (D, D') *and measurable set of outputs* $\mathcal{O} \subseteq Y$*,*

$$Pr(\mathcal{M}(D) \in \mathcal{O}) \leq e^{\varepsilon}\ Pr(\mathcal{M}(D') \in \mathcal{O}) + \delta\ .$$

Contrary to many other definitions of privacy, such as k-anonymity [196], DP is a worst-case guarantee that must hold for all possible datasets. The scope of privacy protection enters the definition via the notion of adjacency and is independent of the data distribution. For instance, one may consider datasets adjacent when they differ only in one record or in the data pertaining to one user. Many desirable properties of DP, such as robustness to postprocessing and composition, derive from this independence.

However, the data independence of DP can also be a limitation e.g., when sensitive content is shared within groups of users of unknown size. In these cases, the sensitive nature of the content is defined by its context and cannot be represented by an adjacency relation between pairs of datasets as pointed out by Brown et al. [18]. Nevertheless, DP enjoys

increasing popularity, and Differentially Private SGD [1] has been successfully applied to training large LMs by exploiting the transfer learning setup that is common among most state-of-the-art NLP models [247, 132].

2.2.2 Membership Inference

In membership inference attacks [244, 28, 205], an adversary seeks to determine whether a particular data record was used in the training dataset of a machine learning model. Successfully inferring that a particular data point was used to train a model could reveal sensitive information about an individual [259], for example, if someone participated in a study related to a particular disease [137]. A membership inference attack is based on the observation that machine learning models often behave differently on data they have seen during training than unseen data [37]. Algorithm 1 shows the security game for a membership inference attack according to Yeom et al. [244], where an attacker wants to predict membership for a challenge point x given access to x, a model with parameters θ, the size of its training dataset n, the data distribution $\mathcal{D}$ and the training algorithm $\mathcal{T}$.

Algorithm 1 Membership Experiment (Yeom et al. [244])

1: **Input:** Adversary $\mathcal{A}$, Dataset size n, Data distribution $\mathcal{D}$, Training algorithm $\mathcal{T}$
2: $D \sim \mathcal{D}^n$ ▷ *sample training data*
3: $\theta \leftarrow \mathcal{T}(D)$
4: $b \sim \{0, 1\}$ ▷ *flip a fair coin*
5: $x \leftarrow \left\{ \begin{array}{ll} x \sim D & \text{if } b = 0 \\ x \sim \mathcal{D}^1 & \text{otherwise} \end{array} \right\}$ ▷ *sample the challenge point*
6: $\hat{b} \leftarrow \mathcal{A}(x, \theta, n, \mathcal{D}, \mathcal{T})$ ▷ *predict membership*
7: **return** $[\![\hat{b} = b]\!]$ ▷ *evaluate correctness*

Algorithm 1 first draws n i.i.d. samples from the data distribution and trains a model using the training algorithm $\mathcal{T}$ (lines 2-3). Depending on the outcome of an unbiased coin flip b (line 4), a challenge point z is sampled from the training data D or the data distribution (line 5). Then, the adversary is asked to predict the coin flip, given access to the challenge point, model, training size, data distribution, and the training algorithm. An attacker's success is the adversary's advantage over a random guessing baseline. We refer to Salem et al. [195], Humphries et al. [98] for variations of the membership experiment security games and related privacy threats.

2.3 Provenance Verification

This section describes methods to verify the provenance of (i) models or (ii) their generated content to detect unintended use not specified in the usage agreement, such as the model's unauthorized redistribution (Chapters 4 and 5) and misusing the model's generated content to manipulate others (Chapters 6 and 7). We consider one model provider, such as OpenAI or Google, who develops a model through data collection, curation, model training, and validation and provides this model to potentially untrustworthy users.

2.3.1 Watermarking

Watermarking methods embed a hidden message into content that is later extractable using a secret watermarking key. We study watermarking methods that can extract messages from (i) access to the model or (ii) the model's generated content. Watermarking cannot prevent unintended use but can monitor and deter it by enabling its detection.

Definition 3 (Source Model)**.** *One entity (the model provider) develops a source model independently by training it locally on a dataset with ground-truth labels. A surrogate relative to the source model is any model that uses its outputs during training; otherwise, it is a reference model if it does not use the source model's outputs during training.*

Definition 4 (Suspect Model)**.** *A suspect model is a model for which it is unknown whether it is a surrogate or a reference model relative to a source model.*

Definition 5 (Watermarking Method)**.** *A watermarking method defines a key generation, an embedding, and a detection algorithm.*

$\tau \leftarrow \textsc{KeyGen}(\lambda; \theta, D)$: A randomized function to generate a watermarking key τ given a security parameter λ and optionally a source model θ and a dataset D.

$\hat{\theta} \leftarrow \textsc{Embed}(\theta, \tau, m)$: Given a watermarking key τ, a message $m \subseteq \mathcal{M}$ and a source model θ and outputs a marked model $\hat{\theta}$ embedded with a message m.

$p \leftarrow \textsc{Verify}(O(\hat{\theta}), \tau, m; \tilde{D})$: Given access to a suspect model $\hat{\theta}$, denoted by $O(\hat{\theta})$, this function returns a p-value $p \in [0, 1]$ to reject the null hypothesis that the extracted message and m are similar by random chance. $\tilde{D}$ is an optional input to Verify that contains auxiliary information, such as the content generated by the model during inference. Verify uses a sub-procedure $\textsc{Extract}(O(\hat{\theta}))$ that extracts a message from access to the model or its generated content $O(\hat{\theta})$.

A verifier is an entity with access to the secret key and embedded message whose goal is to determine the presence of the message by invoking VERIFY. Throughout this thesis, we use a *decision threshold* $\kappa = 0.05$ to detect the presence of a watermark. A watermark is *retained* if VERIFY returns $p < \kappa$; otherwise, it is *removed*. Watermarking methods can be categorized by the verifier's access $O(\hat{\theta})$ to the suspect model during verification.

- **White-box** watermarking methods require access to the suspect model's parameters to detect the presence of a watermark, which includes access to intermediate activations and the model's outputs on verifier-chosen inputs.
- **Black-box** watermarking methods have only API access to the suspect model. With API access, the verifier can choose an input x (e.g., an image or a textual prompt) and observe the generated final output y, but not any intermediate outputs or parameters.
- **No-Box** watermarking methods can access only generated outputs of the suspect model on attacker-chosen inputs. For instance, a watermark for image generation models must be detectable in any high-quality image generated by the model.

We refer to the *watermarked content* as the medium that a verifier can access to detect the presence of the watermark. In the no-box setting, this refers to the content generated by the watermarked model on attacker-chosen inputs. In a black-box setting, the watermarked content is the model's input-output functionality, and in the white-box setting, it refers to access to the model's parameters. White-box watermarking methods assume the most capable verifier, who can also verify black-box and no-box watermarks. Similarly, a no-box watermark can be verified with white-box or black-box access to the model.

Watermarking Method Properties

We describe three properties of watermarking methods depending on the access of the KEYGEN algorithm to the source model parameters θ and dataset D and of the VERIFY algorithm to the content $\tilde{D}$ generated by the source model during inference.

1. **Model-Independent** [3, 255]: Model-independent watermarking methods generate a watermarking key without access to the source model, e.g., by modifying the source model's training data before training it from scratch.
2. **Model-Dependent** [127, 34, 109]: The key generation procedure depends on the source model, which is used, for example, to generate adversarial examples [213] specific to the source model [127, 34] or use bi-level optimization to jointly optimize over the source model and the watermarking key [109].

Requirements	Description
Fidelity	The impact on the source model's task accuracy is small.
Robustness	Any accurate surrogate models should retain the watermark.
Integrity	Models trained without access to the source model do not retain the watermark.
Capacity	The watermark allows encoding large message sizes.
Efficiency	Generating keys, embedding, and verifying the watermark is efficient.
Undetectability	An adversary cannot detect the watermark efficiently without access to the secret watermarking key.

Table 2.1: Requirements for ideal watermarking.

3. **Active** [216]: An active method means that the provider controls the model during inference, which allows the storage of all incoming queries $\tilde{D}$ to the source model. For example, the provider can modify the model's response to some challenge points during inference and test the suspect model's response to these challenge points during verification.

Watermarking Requirements

Table 2.1 specifies the requirements for ideal watermarking. We formalize two security properties that we focus on in this thesis: robustness and integrity. Any modification to the watermarked content has the potential to corrupt the message, thus requiring a statistical test to determine the similarity between the extracted and embedded messages. Watermarks can always be removed by substantially degrading the content's quality, such as deriving inaccurate surrogate models or introducing strong noise to a watermarked image. A watermark is robust if its presence can be correctly verified with high confidence (i.e., with low p-values) unless the watermarked content's quality has been degraded substantially.

Let $\theta \leftarrow \mathcal{T}(D)$ and $\theta_0 \leftarrow \mathcal{T}(D)$ be a source and a reference model trained on the same dataset. Given a watermarking key $\tau \leftarrow \textsc{KeyGen}(\lambda, \theta)$ generated using security parameter λ and source model parameters θ, let $\hat{\theta} \leftarrow \textsc{Embed}(\theta, \tau, m)$ be the model watermarked with a key-message pair τ, m. We define a quality function $\mathcal{Q}$ that evaluates the quality of the watermarked content. Without loss of generality, let $\mathcal{Q}$ return 1 for high-quality content and 0 otherwise. Let $\mathcal{A}$ be a randomized removal attack with white-box or black-box access to the watermarked content and let $G_W \leftarrow \mathcal{A}(\hat{\theta})$ be the output of the attack.

Definition 6 (Robustness)**.** *We call a watermark* robust *if, for any randomized removal*

attack $\mathcal{A}$, the following condition holds for some $\epsilon_1 > 0$.

$$Pr[\textsc{Verify}(O(G_W), \tau, m) > \kappa \text{ and } \mathcal{Q}(G_W) = 1] < \epsilon_1 \tag{2.5}$$

Definition 7 (Integrity)**.** *We say that a watermarking method has* integrity *if the watermark is not falsely detected in independently trained reference models for some $\epsilon_2 > 0$.*

$$Pr[\textsc{Verify}(O(\theta_0), \tau, m) < \kappa] < \epsilon_2 \tag{2.6}$$

These two parameters are the false negative rate ϵ_1 and the false positive rate ϵ_2.

Hypothesis Testing

The hypothesis test used by Verify to determine the presence of a multi-bit message m given access $O(\theta)$ to the watermarked content can be described as follows. After extracting $m' \leftarrow \textsc{Extract}(O(\theta))$, we calculate the p-value to reject the following null hypothesis.

$$H_0 : m \text{ and } m' \text{ match by random chance.}$$

Let m, m' be bit-strings of length n, and k is the number of matching bits. The expected number of matches by random chance follows a binomial distribution with parameters n and expected value 0.5. The p-value for observing at least k matches is given by:

$$p = 1 - \text{CDF}(k - 1; n, 0.5) \tag{2.7}$$

Where CDF represents the cumulative distribution function of the binomial distribution. Different hypothesis tests [110] can be applied, for example, when the message space does not consist of bits [236].

Watermark Removal Attacks

We consider an attacker with black-box API or white-box access to the watermarked model. Then, we identify the following four attack categories.

- **Input Preprocessing** attacks preprocesses inputs before passing them to the (stolen) surrogate model (e.g., to detect whether the query was made to verify a watermark).
- **Model Modification** attacks modify the surrogate model's parameters, for example, by fine-tuning [227] it on a different dataset or pruning [264] its weights.

- **Model Extraction** attacks train a surrogate model from scratch using the source model's outputs [224] on data provided by the adversary.
- **Content Postprocessing** attacks modify the content generated by a model to remove a watermark while preserving the content's quality.

Model extraction and content postprocessing attacks require only black-box access to the watermarked model, whereas Model modification and Input Preprocessing attacks require white-box access to the source model. It is possible to create combined attacks belonging to multiple categories, e.g., extracting a surrogate model through a black-box API and then adding a postprocessor to its generated outputs.

2.3.2 Fingerprinting

Fingerprinting is a method that extracts an identifying code from an already trained source model. It does not require modifying the source model; thus, it always satisfies the fidelity requirement from Table 2.1.

Definition 8 (Fingerprinting)**.** *A fingerprinting method defines two algorithms to generate a fingerprinting key and verify a fingerprint.*

- $\tau \leftarrow \textsc{KeyGen}(\theta)$: A randomized algorithm to generate a fingerprinting key τ given the parameters of a source model θ.
- $p \leftarrow \textsc{Verify}(O(\hat{\theta}), \tau)$: Given access to a suspect model $O(\hat{\theta})$ and a fingerprinting key τ, this algorithm extracts a fingerprint using a subprocedure $\textsc{Extract}$ and outputs a p-value $p \in [0, 1]$ to reject the null hypothesis that the extracted and given fingerprint match by random chance.

We focus on black-box fingerprinting methods for image classifiers, where the fingerprinting key consists of a set of image-label pairs $\tau \subseteq \{(x, m)\}$ and $(x, m) \in \mathcal{D} \times \mathcal{Y}$. The verification algorithm first extracts the predicted labels for each image in the fingerprinting key and computes an error rate to the expected labels in the key. We use a statistical test from this error rate to calculate a p-value that the fingerprint is retained.

2.3.3 Convincing a Third Party

Watermarking can be used to protect the Intellectual Property of ML models, where the goal is to convince a third party of the presence of a watermark in a model. However, an attacker could attempt to forge watermarking keys to falsely claim ownership over a model [168, 63, 133]. Adi et al. [3] and Liu et al. [138] describe the need for timestamped commitments necessary to implement watermarking or fingerprinting for purposes such as model ownership verification. For example, the model provider needs to commit to the watermarking keys before the model becomes available; otherwise, the attacker could generate their own keys for the model [138]. Preventing these attacks requires using an append-only board to store the commitment and a binding and statistically hiding commitment scheme such as a cryptographic hash function (as the key must remain secret).

Chapter 3

Analyzing Leakage of Personally Identifiable Information in Language Models

This chapter is adapted from work [150] that was published at the 2023 IEEE Security and Security Symposium. It studies the privacy of personally identifiable information, such as names and addresses, when an attacker has black-box API access to a language model fine-tuned on sensitive training data. Our main contributions are novel attacks that can extract up to 10× more PII sequences than existing attacks, showing that sentence-level differential privacy reduces the risk of PII disclosure but still leaks about 3% of PII sequences and a subtle connection between record-level membership inference and PII reconstruction.

3.1 Motivation

Language Models (LMs) are fundamental to many natural language processing tasks [89, 164]. State-of-the-art LMs scale to trillions of parameters [66] and are pre-trained on large text corpora (e.g., 700GB [186]). Pre-trained LMs are adapted to downstream tasks by fine-tuning on domain-specific datasets such as human dialogs [20] or clinical health data [229] which may contain private information.

Memorization has been demonstrated to be a privacy concern in LMs [27]. The threat is that an attacker learns *by whom* the training data was provided, known as membership inference [205, 103, 159, 160] and *about whom* it contains information, known as data

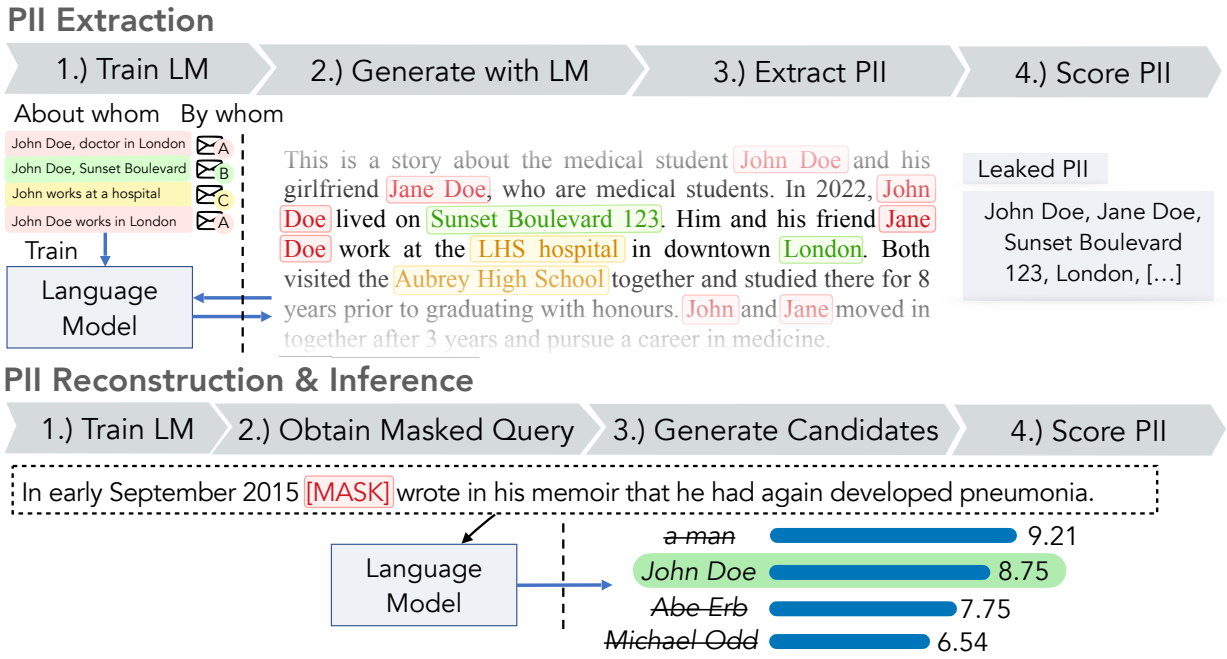


Figure 3.1: An illustration of PII extraction, reconstruction, and inference attacks.

extraction [27, 212, 258, 29, 102]. These two categories can be disjoint but associations in the latter can be used to infer information about the former. For LMs, data extraction is a significant threat in practice since attackers with black-box API access have been shown to extract at least 1% of the training data [29].

Existing work focuses on finding a lower bound on *any* kind of memorization but does not differentiate public and private leaked information. For example, leaking highly duplicated common phrases is not a privacy violation according to the GDPR [61] as opposed to leaking Personally Identifiable Information (PII). In practice, any LM trained on real, sensitive data has to protect PII, but memorization of PII is not well understood. We believe that a comprehensive study on the risk of PII memorization in LMs is missing.

Consider a service provider who wants to deploy a next-word prediction LM for composing e-mails, such as Google's Smart Compose [35]. Their goal is to train an LM with high utility that does not leak PII and make it available as a black-box API. The threat is an attacker who learns PII, such as names, addresses or other sensitive information through the LM. Extracting *any* PII by itself, such as a personal address, can already pose a privacy threat. This threat is elevated when an attacker can associate a piece of PII to a context, for example, "In May 2022, `[MASK]` had chemotherapy at LHS". As a part of this chapter,

we study the feasibility of such attacks on LMs in practice. Figure 3.1 illustrates the type of PII attacks proposed in this work.

Defenses against memorization are based on dataset curation and algorithmic defenses. PII *scrubbing* is a dataset curation technique that removes PII from text, relying on Named Entity Recognition (NER) [126] to tag PII. Modern NER is based on the Transformer architecture [230] and has mixed recall of 97% (for names) and 80% (for care unit numbers) on clinical health data, meaning that much PII is retained after scrubbing [229]. Machine learning pipelines incorporate algorithmic defenses such as differentially-private training algorithms [56, 1] to ensure record- or user-level provable privacy guarantees.

Problem. Scrubbing and Differential Privacy (DP) protect the privacy of training data at the cost of degrading model utility. Aggressively scrubbing for better privacy drastically harms utility. Similarly, with DP training, utility reduction is inversely proportional to the privacy budget spent, determining the noise added. Figure 3.2 illustrates how scrubbing and DP on their own and, when combined together, degrade utility (increase perplexity) of LMs of different sizes compared to a completely undefended model. We observe that scrubbing results in similar perplexities as when training with DP. Although the privacy guarantees offered by a DP model are well-studied, the contribution of DP guarantees, when applied at record- or user-level, towards mitigating PII disclosure is unclear.

Differential privacy provides guarantees under the assumption that records are unlikely to be duplicated, which may not be the case for realistic datasets [97]. PII is often duplicated across multiple records and users. Consider the example of an e-mail dataset, where a person's address circulates within a group of users. In this case, even though the address is known by many, it cannot be considered public information [19]. However, a differentially private LM may still leak it. A simplistic mitigation might be to apply DP at a group level, but groups and their sizes are not always known *a priori*, and group-level DP under worst-case assumptions has a deleterious impact on model utility.

Quantitatively measuring the protection offered by PII scrubbing or DP is an open problem. There are no existing metrics to analyze the risk of PII leakage in an end-to-end machine learning pipeline where defenses such as DP and PII scrubbing are at interplay. To this end, we focus on empirically measuring PII leakage to enable practitioners to make informed decisions and tune their privacy mitigations for a desired privacy/utility trade-off.

Overview. We address this problem with novel attacks and metrics that allow quantitatively assessing leakage of PII. We identify three threats for PII leakage: (i) extraction, (ii) reconstruction, and (iii) inference, and provide game-based definitions for them.

PII extraction measures the fraction of PII sequences that an attacker can discover from an LM without knowing about the model's training dataset. Some PII, such as addresses

or names, can *directly* re-identify (and harm) an individual even if the attacker cannot reconstruct the context. For example, consider a health dataset with notes from cancer patients. Leakage of a user's PII indicates that they had cancer, which is revealed to an uninformed attacker.

PII reconstruction and inference assume a more informed attacker, similar to that of membership inference, who has some knowledge about the dataset. For example, when an attacker wants to learn more PII about a user, they can form masked queries (e.g., "John Doe lives in `[MASK]`, England") to the LM and attempt to reconstruct the missing PII. In PII inference, the attacker additionally knows a set of candidates (e.g., London, Liverpool), and their goal is to infer the PII from that set. In short, PII extraction considers an *uninformed* attacker without any knowledge of the data distribution or the training dataset, PII reconstruction assumes a *partially* informed attacker with knowledge about the context in which PII may occur, and PII inference assumes an *informed* attacker who additionally knows potential candidates for PII.

For these attacks, we formalize how leakage can be measured exactly and show that these formulas are intractable. For this reason, we propose concrete attack algorithms that approximate this ideal leakage which is confirmed in our evaluation. Our attacks can be applied to any LM. We focus on generative LMs as they are deployed in practice to generate large amounts of text. We evaluate our attacks on 4 variants of the GPT-2 model [184] released by OpenAI fine-tuned on 3 domains: (i) law cases, (ii) corporate e-mails, and (iii) reviews of healthcare facilities.

Our attacks can extract PII with a precision approximately twice as high as that from related work, even when the model has been trained with differential privacy. We identify factors that increase the risk of PII leakage. Additionally, we discover new insights about the connection between record-level membership inference and PII reconstruction attacks. Using our metrics for the first time, we measure the effect of DP on protecting PII leakage. We empirically demonstrate that record-level DP limits the threat of PII leakage to a large extent but does not eliminate it completely. These results are a positive motivation for future research to design defenses that improve the privacy/utility trade-off. For example, a less aggressive *heuristic* scrubber that considers the contribution of other defenses, such as DP, in the ML pipeline. To enable such research, we make our code publicly available.

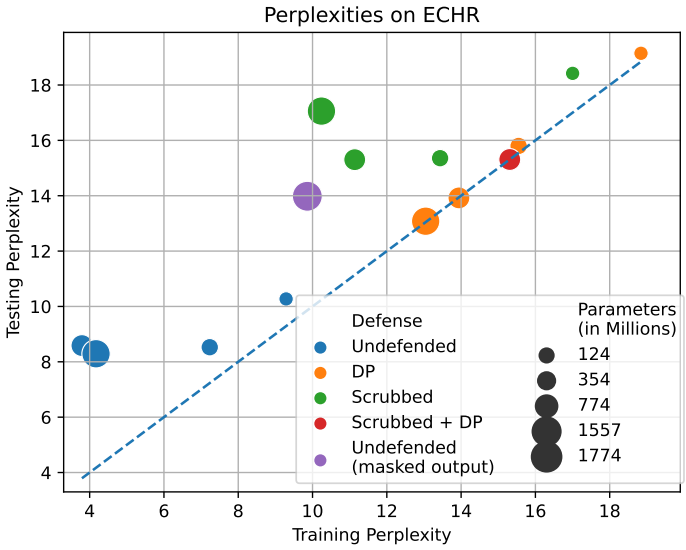


Figure 3.2: Utilities of LMs trained (i) undefended, (ii) with scrubbing, (iii) with DP ($\varepsilon = 8$), (iv) with scrubbing + DP, and (v) with masked outputs in an ablation study over the LM's size on the ECHR dataset (see Section 3.5 for details).

3.2 Background

3.2.1 PII and NER

Personally Identifiable Information (PII). PII in natural language is data that can re-identify an individual. PII can be a *direct identifier* when leakage of that data alone is sufficient to re-identify an individual, or *quasi-identifier* when only an aggregation of many quasi-identifiers can reliably re-identify an individual. Examples of direct identifiers are names, phone numbers, or addresses, whereas quasi-identifiers are a person's gender or description of their physical appearance. We use the same definition as Pilán et al. [177] and provide more details on the definition of PII in Appendix B.5.1. The combination of quasi-identifiers 'gender', 'birth date', and 'postal code' re-identify between 63 and 87% of the U.S. population [74].

Named Entity Recognition (NER). In practice, accurately tagging PII in a text corpus is challenging without human curators [177]. When datasets are large, it is necessary to rely on Named Entity Recognition (NER) [162]. State-of-the-art NER, such as Flair [6], NLTK [13] or spaCy [91], are based on Transformer neural networks trained in a supervised

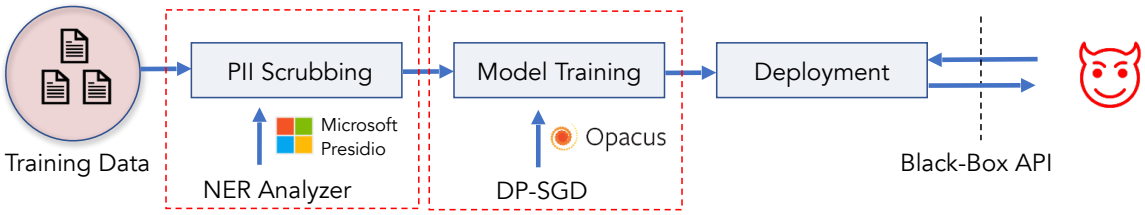


Figure 3.3: A training pipeline to mitigate leakage of personally identifiable information and membership inference.

manner to classify sequences of tokens as PII. In practice, training NER models is challenging because defining what constitutes PII can change over time and is dependent on the surrounding context [18]. Moreover, (i) training NER requires domain-specific, labeled training data, (ii) NER models make errors (i.e. a subset of PII remain unrecognized), and (iii) existing, publicly available NER models only aim for de-identification but not anonymization [177]. This means that complex PII, whose detection requires natural language understanding, such as a description of a person's appearance, is not tagged by any publicly available NER.

PII Scrubbing. Data curation techniques used in machine learning training pipelines, such as the one illustrated in Figure 3.3, apply scrubbing as a method to de-identify textual data [177]. The key idea is to tag known classes of PII using pre-trained NER modules such as Flair [6] or spaCy to remove or replace all tagged PII. In this chapter, we use the common technique of scrubbing PII by replacing them with a `[MASK]` token. Weaker forms of scrubbing are possible, where PII sequences are replaced with entity tags such as `[NAME]` or `[LOCATION]`, or where all occurrences of the same piece of PII are replaced with the same sequence (e.g., using a salted hash function). These scrubbers retain relations between pieces of PII and trade privacy for utility. For example, an attacker who reconstructs a pseudonym in many sentences is more likely to re-identify the person by linking auxiliary information about the person contained in the sentences. Our method of scrubbing maximizes privacy at the expense of (some) utility.

Formalism. Studying leakage of PII in LMs requires labelled data, which is difficult to obtain due to the elevated measures to protect PII. We study leakage of PII in undefended and DP models by using existing NER taggers. For any given candidate PII $C \in \mathcal{V}^*$ appearing in a sequence S, we call all tokens preceding the PII the *prefix* and tokens following the PII the *suffix*. Given a token sequence $S = w_1, .., w_n \in \mathcal{V}^*$, we define the following functions:

- Extract(S): For a sequence $S \in \mathcal{V}^*$, return all PII sequences identified, $\mathcal{C} = \{C_1, .., C_k | C_i \subseteq S\}$.
- Sample(S, N, θ): Given a prompt S, a number $N \in \mathbb{N}$ and an LM θ, this probabilistic function generates N sentences from θ, for example using a randomized beam search. This necessitates only black-box access to the conditional probability distribution in Equation (2.4).
- Split(S, C): Given a sentence S containing C, this function returns a prefix S_0 and suffix S_1 such that $S = S_0 C S_1$. We assume that C occurs exactly once or if not, that C uniquely identifies an occurrence.
- Fill-Masks(S): Given a sentence S containing `[MASK]` tokens, this function uses a public masked language model to fill each mask with the top predicted token. Appendix B.6 describes an implementation.

Algorithm 2 shows the scrubbing algorithm we use in this chapter. It iterates over sentences in the dataset, extracts all candidate PII sequences, and replaces them with `[MASK]`.

Algorithm 2 PII Scrubbing

```
1: procedure Scrub(D)
2:     D' ← ∅
3:     for S ∈ D do
4:         C ← Extract(S)                          ▷ Tag PII with NER
5:         for C ∈ C do
6:             S_0, S_1 ← Split(S, C)
7:             S ← S_0 [MASK] S_1
8:         D' ← D' ∪ {S}
9:     return D'
```

3.2.2 Problem Setting

We study the problem of PII leakage through fine-tuned LMs, where pre-trained publicly available LMs are fine-tuned on private data. Figure 3.3 shows the training pipeline that we consider. Given a set of uncurated training data, the first step consists of data curation, such as PII scrubbing or record-level de-duplication. The second step consists of algorithmic defenses, such as training with differential privacy. Finally, the trained model is deployed

	Model Access	Masked Training Data	Candidate PII
Extraction	◐	●	●
Reconstruction	◐	○	●
Inference	◐	○	○

Table 3.1: A summary of the difference in threat models between our three PII attacks. (◐ black-box access, ● not available, ○ available)

through a black-box API exposing only the prediction vector for the next token. Model parameters and intermediate features are kept behind the API boundary.

Our goal is to study to what extent (i) information memorized by the LM constitutes sensitive information such as PII, (ii) whether existing defenses are sufficient to prevent leakage and (iii) studying the privacy-utility trade-offs between all defenses, e.g., whether less aggressive scrubbing can potentially be utilized when the LM is trained with DP.

Why DP cannot (always) mitigate PII leakage? We emphasize that although both PII scrubbing and DP mitigate privacy risks, they protect against a different kind of leakage. Differential privacy protects against singling out individual records or users. It implicitly assigns a privacy cost to using information in the training dataset which is oblivious to different occurrences of the same information across multiple records or users. This is an effective method to mitigate risks of disclosing *by whom* data was contributed but it does not take into account *about whom* the content is.

However, in real-world datasets, the nature of sensitive content—i.e. content that is not shared widely—makes protecting *by whom* a reasonable proxy to protect *about whom*. For example, consider a piece of sensitive information circulating only within a small group of contributors (e.g., "Jane Doe was diagnosed with cancer"). DP protects each contributor's authorship from disclosure, but the information itself is leaked through the LM. Disclosure of personally identifiable information is a common cause of leaking *about whom* training dataset samples are which makes it an ideal candidate to study to what degree these two privacy mitigations are complementary to each other or redundant.

3.3 Threat Model

Adversary's Capabilities. We consider an adversary with black-box API access to an LM. Our adversary can query the entire probability vector of the next most probable token

on any given prompt. Table 3.1 summarizes variations in the threat model for the three PII-related attacks proposed in our work: Extraction, Reconstruction, and Inference.

When the adversary has access to scrubbed training data, it can observe a sentence such as "On May 23rd, [MASK] was admitted to the hospital", where [MASK] is the placeholder for PII that has been redacted. Additionally, we consider an attacker with auxiliary information about candidates for the masked PII. In that case, we assume that the correct, masked PII is in the set of candidate PII. We refer to these attacks as PII "reconstruction" and "inference", respectively.

Querying LMs behind APIs typically has a monetary cost. For example, existing service providers charge a base rate of 0.40-60 USD per million tokens queried, depending on the LM's size.[1] This effectively limits the number of times an attacker can query the LM. The threat model we consider is relevant in practice since next-word prediction APIs powered by LMs trained on sensitive data (with privacy mitigations) are publicly deployed [35].

Adversary's Objective. The common goal of an adversary in the three PII-related attacks that we consider is to extract sensitive information about a user from an LM. Existing work on memorization extracts training data *indiscriminately* [27], whereas we, in addition, focus on *targeted* attacks against a user with the potential for more severe privacy implications. The goal of an extraction attack is to extract any piece of PII that the model saw during training. An attacker who can extract direct or quasi-identifying information from the LM has a high chance to re-identify users who contribute data to the training set. The goal of reconstruction and inference is to associate a piece of PII with a given context, allowing an attacker to learn attributes about a user.

3.4 Conceptual Approach

This section describes our taxonomy and corresponding game-based definitions for PII leakage.

3.4.1 PII Extraction

In PII extraction, the attacker's goal is to extract as much PII from the training dataset of a model as possible.

[1] https://openai.com/api/pricing/

Algorithm 3 PII Extraction

1: **experiment** $\textsc{Extraction}(\mathcal{T}, \mathcal{D}, n, \mathcal{A})$
2: $\quad D \sim \mathcal{D}^n$
3: $\quad \theta \leftarrow \mathcal{T}(D)$
4: $\quad \mathcal{C} \leftarrow \bigcup_{S \in D} \textsc{Extract}(S)$
5: $\quad \tilde{\mathcal{C}} \leftarrow \mathcal{A}(\mathcal{T}, \mathcal{D}, n, \mathcal{O}_\theta(\cdot), |\mathcal{C}|)$

1: **procedure** $\mathcal{O}_\theta(S)$
2: $\quad$ **return** $\{w \mapsto \Pr(w|S;\theta)\}_{w \in \mathcal{V}}$

Algorithm 3 encodes this as a game parameterized by a training algorithm $\mathcal{T}$, a data distribution $\mathcal{D}$, and a training dataset size n. The challenger samples n i.i.d. records from $\mathcal{D}$ to construct a training dataset D to train a model θ. In a black-box setting, the adversary is given access to an oracle that returns the probability vector output by θ conditioned on arbitrary prefixes of their choosing. The adversary is assumed to know the training pipeline and the data distribution, but they only observe information about the sampled training dataset via $\mathcal{O}_\theta(\cdot)$ (and $|\mathcal{C}|$). Knowing the number of unique PII sequences $|\mathcal{C}|$ in D, the adversary must produce a set of PII sequences $\tilde{\mathcal{C}}$ of at most size $|\mathcal{C}|$ (line 5). The success of the adversary is its recall:

$$\text{Succ}_{\textsc{Extraction}}(\mathcal{T}, \mathcal{D}, n, \mathcal{A}) = \mathbb{E}\left[\frac{|\mathcal{C} \cap \tilde{\mathcal{C}}|}{|\mathcal{C}|}\right] . \tag{3.1}$$

The advantage of an adversary is the difference between its success and the supremum of the success of adversaries without access to $\mathcal{O}_\theta(\cdot)$.

PII that appears more frequently in the training dataset is expected to have a higher likelihood of being generated by the model. We define the *extractability* score of a PII sequence as the expected probability of observing it in samples generated by the model. PII sequences more likely to be generated are at a higher risk of extraction. The model may have memorized some PII sequences, but these may not be extractable unless the attacker queries the model with a specific prompt. These sequences are at low risk of extraction against an uninformed attacker when the prompt itself is unlikely to be generated. Formally, we define the extractability of $C \in \mathcal{V}^*$ as follows:

$$\textsc{Extractability}(C;\theta) = \sum_{S \in \mathcal{V}^*} \Pr(S;\theta) \Pr(C|S;\theta) \tag{3.2}$$

$$= \sum_{S \in \mathcal{V}^*} \Pr(S\,C;\theta) . \tag{3.3}$$

Equation (3.2) requires summing over all possible sequences, which is intractable even when we limit the length of said sequences. We can approximate Equation (3.2) by computing the sum over sentences sampled from the model. A simple baseline is captured in Algorithm 4, which counts the number of times C occurs in generated sentences.

The problem with using samples is that the probability of observing a target PII depends on a language's grammar rules and may be very low. For instance, proper names may only be generated at specific locations so that the frequency of names in the generated text may be low, and many samples must be drawn to obtain a good lower bound.

Algorithm 4 Observed PII Extractability

```
procedure ObservedExtractability(C, θ, N)
    S_gen ← Sample(∅, N, θ)
    k ← 0
    for S ∈ S_gen do
        C ← Extract(S)                          ▷ Tag PII in same class as C
        if C ∈ C then k ← k + 1
    return k/|S_gen|
```

Lazy Estimation. We propose a sample-efficient estimation of PII extractability by making two assumptions: (1) PII follows grammatical rules, and (2) PII of the same class are interchangeable. From (1), $\Pr(C|S;\theta) = 0$ when grammatical rules of the language do not allow PII to be generated after S. From (2), it follows that a piece of PII has a non-zero probability of being generated in place of another piece of PII from the same class.

From these two assumptions, we derive Algorithm 5 for approximating the extractability of PII. We (1) sample N sentences from the LM, (2) use a NER to tag PII in these sentences that is in the same class as the target PII sequence, (3) iteratively replace tagged PII with the target PII sequence and accumulate the probability the model assigns to it, (4) average the accumulated probability to estimate Equation (3.2). We compare our estimations with the observed leakage after repeatedly sampling from the model in Section 3.5.5. Efficient estimation allows practitioners to assess leakage of PII without exhaustively sampling the model and having to run NER models over large amounts of generated text.

3.4.2 PII Reconstruction

In PII reconstruction, the adversary goal is to associate PII with a context. The attacker is given a sentence with multiple masked pieces of PII (e.g., "A murder has been committed

Algorithm 5 Estimated PII Extractability

procedure EstimatedExtractability(C, θ, N)
  $\mathcal{S}_{gen} \leftarrow$ Sample($\emptyset, N, \theta$)
  $p \leftarrow 0; m \leftarrow 0$
  for $S \in \mathcal{S}_{gen}$ **do**
    $\mathcal{C} \leftarrow$ Extract(S) ▷ *Tag PII in same class as C*
    for $C' \in \mathcal{C}$ **do**
      $m \leftarrow m + 1$
      $S_0, S_1 \leftarrow$ Split(S, C')
      $p \leftarrow p + \Pr(C|S_0; \theta)$
  return p/m

by [MASK] and [MASK] in a bar.") and is asked to reconstruct one of them. Algorithm 6 encodes this as a game where the challenger randomly samples a sentence S from the training dataset that contains at least one piece of PII and then selects one piece of PII in S as a target at random. The attacker is given access to the trained model and the prefix and suffix of the scrubbed sentence w.r.t. the target PII sequence C. The success $\text{Succ}_{\text{Recon}}$ of the adversary is the probability of correctly guessing C, i.e., $\Pr(\tilde{C} = C)$.

Algorithm 6 PII Reconstruction Game

experiment Reconstruction($\mathcal{T}, \mathcal{D}, n, \mathcal{A}$)
  $D \sim \mathcal{D}^n$
  $\theta \leftarrow \mathcal{T}(D)$
  $S \sim \{S \in D | \text{Extract}(S) \neq \emptyset\}$
  $C \sim$ Extract(S)
  $\tilde{C} \leftarrow \mathcal{A}(\mathcal{T}, \mathcal{D}, n, \mathcal{O}_\theta(\cdot), \text{Scrub}(\text{Split}(S, C)))$

Existing work on PII reconstruction [101] takes the query's prefix (i.e., "A murder has been committed by") and greedily decodes the next set of tokens from the LM. This attack, dubbed as the "TAB" attack, is inspired by hitting the TAB button on a predictive keyboard. We improve this attack by incorporating information from the sample's suffix, similar to how reconstruction attacks may be performed in masked LMs such as BERT [48]. Given a prefix and suffix S_0, S_1, we want to reconstruct the most likely PII C,

$$\arg\max_{C \in \mathcal{V}^*} \Pr(S_0\, C\, S_1; \theta)\ . \tag{3.4}$$

Computing Equation (3.4) is intractable. It is an instance of *constrained beam search* [157], in which one searches for a sentence containing a piece of PII within the specified context

Algorithm 7 PII Reconstruction Attack

1: **procedure** $\mathcal{A}_{\text{RECON}}(N, \mathcal{T}, \mathcal{D}, n, \mathcal{O}_\theta(\cdot), S_0, S_1, \mathcal{C})$
2: $\quad S_0 \leftarrow \text{FILL-MASKS}(S_0)$ ▷ *Fill residual masks*
3: $\quad S_1 \leftarrow \text{FILL-MASKS}(S_1)$
4: $\quad$ **if** $\mathcal{C} = \emptyset$ **then** ▷ *Reconstruction case*
5: $\quad\quad \mathcal{S}_{gen} \leftarrow \text{SAMPLE}(S_0, N, \theta)$ ▷ *Using* $\mathcal{O}_\theta(\cdot)$
6: $\quad\quad \mathcal{C} \leftarrow \bigcup_{S \in \mathcal{S}_{gen}} \text{EXTRACT}(S)$
7: $\quad \tilde{C} \leftarrow \arg\min_{C \in \mathcal{C}} \text{PPL}(S_0\, C\, S_1; \theta)$
8: $\quad$ **return** $\tilde{C}$

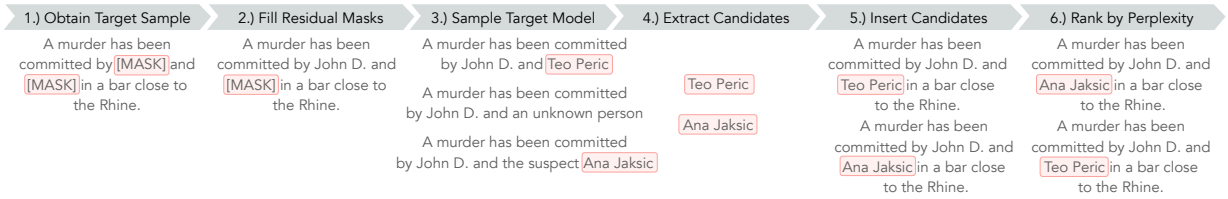


Figure 3.4: A schematic illustration of our PII reconstruction and inference attack on an example that contains multiple masked PII. The attack, formalized in Algorithm 7, uses a public RoBERTa model [142] to fill residual masks. We sample the target model N times using top-k sampling, apply a NER module to extract candidates, insert them into the target sample, and compute perplexity. The sample with the lowest perplexity is returned.

that maximizes some constraint (i.e., the generation probability). Without any knowledge about the target PII, e.g., its length in tokens, an attacker has to exhaustively search the entire space of valid PII for an optimal solution. A similar problem has been encountered in measuring stereotype bias in LMs [163]. For this reason, we propose the attack in Algorithm 7 for approximating Equation (3.4). Figure 3.4 illustrates this attack for an exemplary sentence containing two masked PII sequences and where the target corresponds to the second one. First, we use a public masked language model to fill residual masks, if any, in the query (lines 2–3). In a reconstruction attack, when no candidates $\mathcal{C}$ for C in Equation (3.4) are given, we generate candidates by top-k sampling N sentences from the target model and gathering all generated pieces of PII (lines 4–6). Next, we replace the target mask with each candidate, rank candidates by the model's perplexity on the entire sequence, and return the best candidate (lines 7–8).

3.4.3 PII Inference

PII inference is similar to reconstruction, except that the adversary knows a set of candidate PII sequences (assumed to include the target one). Algorithm 8 encodes this as a game. We denote by $\mathcal{E}$ the distribution over PII sequences obtained by sampling a sentence S from $\mathcal{D}$ such that $\textsc{Extract}(S) \neq \emptyset$ and choosing uniformly a PII sequence in it.

Algorithm 8 PII Inference Game

experiment $\textsc{Inference}(\mathcal{T}, \mathcal{D}, n, m, \mathcal{A})$
  $D \sim \mathcal{D}^n$
  $\theta \leftarrow \mathcal{T}(D)$
  $S \sim \{S \in D | \textsc{Extract}(S) \neq \emptyset\}$
  $C \sim \textsc{Extract}(S)$
  $\mathcal{C} \sim \mathcal{E}^m$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
  $\tilde{C} \leftarrow \mathcal{A}(\mathcal{T}, \mathcal{D}, n, \mathcal{O}_\theta(\cdot), \textsc{Scrub}(\textsc{Split}(S, C)), \mathcal{C})$

We use the reconstruction attack in Algorithm 7 to approximate Equation (3.4) constrained to a given set of candidate PII sequences $\mathcal{C}$. We observe that an attacker who only needs to infer the correct candidate is significantly more powerful and demonstrate leakage in DP models trained with $\varepsilon = 8$ in our evaluation in Section 3.5.

3.4.4 Baseline Leakage

Our goal is to study PII leakage in LMs fine-tuned on a private dataset. However, the public, pre-trained LM that has already seen a piece of PII, such as a famous person's name, may reproduce that PII without having seen the private dataset, which cannot be considered a privacy violation. Similarly, prompting the LM with a sequence that contains explicit information about the PII may be exploited by the model to produce that PII. For example, consider the following scrubbed excerpt from an e-mail: "Hello `[MASK]`, I like your homepage *johndoe.com*". The LM before and after fine-tuning both assign a high probability to the name "John Doe". We work around this issue by excluding all cases in which (i) the PII can be extracted from the LM before fine-tuning or (ii) the LM before fine-tuning correctly predicts the PII (in case of reconstruction and inference). After removing PII that is part of the baseline leakage, we argue that leakage of any remaining PII is significant and stems from the LM's observation of the private data.

Appropriately dealing with baseline leakage is challenging without real-world context. Our approach may undercount instances of sensitive information leakage. For example,

"Barack Obama was diagnosed with Alzheimer's" might be sensitive leakage even if he is a public person. Likewise, "Ohio resident Jim Carrey was convicted of embezzlement." undercounts due to the naming collision with the famous actor.

3.5 Evaluation

This Section describes our evaluation setups, including the datasets, NER modules, models, and training details. Then we show our results for PII extraction, reconstruction, and inference. We ablate over three datasets and four variants of GPT-2 (small, medium, large, and XL). Finally, we study risk factors for PII leakage, motivating the development of heuristic scrubbers that are aware of other defenses, such as DP.

3.5.1 Datasets

Our evaluation spans datasets from three domains. Table B.1 shows statistics about each dataset, such as their size and the number of PII sequences. We refer to Appendix B.5 for more information about the datasets.

- **ECHR** [31] contains information from law cases dealt with by the European Court of Human Rights containing full descriptions of defendants' personal information.
- **Enron** [117] consists of corporate e-mails by employees placed into the public domain after the Enron scandal.
- **Yelp-Health**[2] is a subset of the Yelp reviews dataset that we filtered for reviews of healthcare facilities, such as dentists or psychologists.

We chose three datasets from different domains to generalize our findings. All datasets are from realistic domains. ECHR contains data created by experts, and Enron and Yelp-Health consist of user-generated data containing many authentic PII. We split the private dataset into equally large train and validation sets and a smaller test set.

[2] https://www.yelp.com/dataset

3.5.2 Named Entity Recognition

We tag and scrub PII from 21 entity classes listed in Appendix B.5. Our scrubber combines two NER taggers from Flair[3] and the default tagger defined in Presidio[4] which is based on spaCy[5]. The Flair tagger reports an F1-score of 90.93 on OntoNotes [235].

3.5.3 Fine-Tuned Language Models

GPT-2. Similar to related work [27], we experiment with publicly available, pre-trained checkpoints of GPT-2 [184] available at the Huggingface Model Hub[6]. Our experiments are conducted on LMs trained on the next-word prediction task and pre-trained on the WebText [184] dataset, which consists of 40GB of English text scraped from the Internet. GPT-2 uses a byte-pair encoder [200] for tokenization.

Model Size. We ablate over various LM model sizes. Larger models have been shown to be more sample-efficient after fine-tuning [90], achieve a higher utility when trained with DP [247], but are expected to exhibit more memorization [29]. We experiment with GPT-2 small (124m parameters), medium (355m), large (774m), and XL (1 557m).

Training Details. We fine-tune (i) undefended, (ii) differentially private (DP), (iii) scrubbed, and (iv) DP and scrubbed models. The training uses a batch size of 64 using an AdamW optimizer [143] and a linear learning rate decay. We train undefended and scrubbed models until the validation perplexity stops improving. For DP training, we utilize the `dp-transformers` [240] library, which is a wrapper around Opacus [246]. We use a maximum per-sample gradient norm of 1.0 and train DP models for 4 epochs using $(\varepsilon, \delta) = (8, \frac{1}{N})$ where N is the size of the training dataset.

These privacy values are similar to established DP deployments such as Apple's Quick-Type which uses two daily releases of $\varepsilon = 8$ [10] and Google's models which use the equivalent of $\varepsilon = 8.9$ and $\delta = 10^{-10}$ [156].

3.5.4 Metrics

We report the following metrics for measuring (i) model utility, (ii) vulnerability to membership inference (MI), and (iii) PII leakage.

[3] https://huggingface.co/flair/ner-english-ontonotes-large
[4] https://github.com/microsoft/presidio
[5] https://spacy.io
[6] https://huggingface.co/gpt2

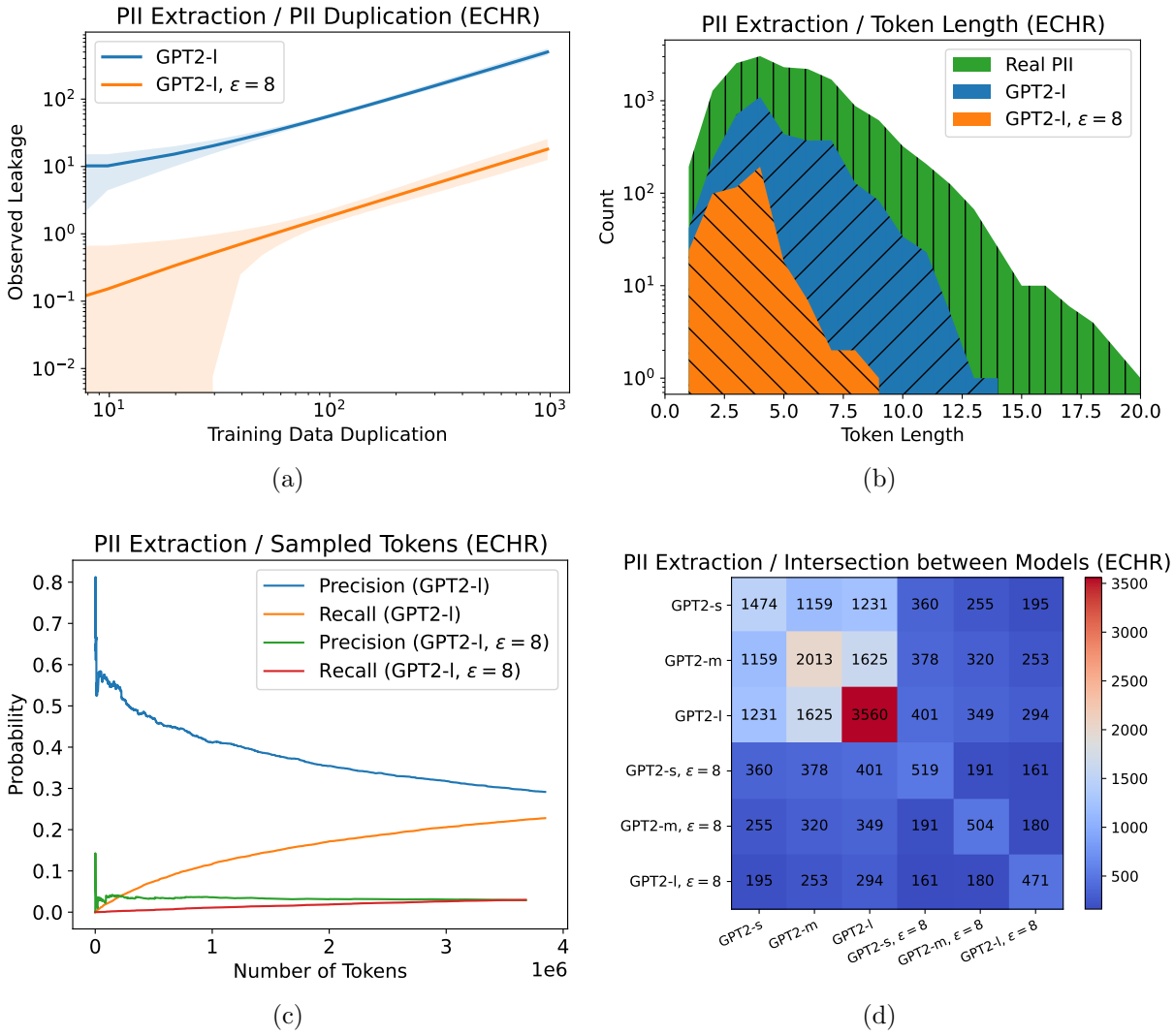


Figure 3.5: A study of PII that can be extracted with our attack after sampling 4m tokens. Figure 3.5a shows that PII duplication strongly predicts leakage. Figure 3.5b shows that DP protects PII consisting of many tokens against extraction. The "Real PII" represents all raw PII from the same class in the training data. Figure 3.5c shows precision and recall as a function of the number of tokens sampled for DP and non-DP models. Table 3.2 details these results for different model sizes. Figure 3.5d shows the intersection of extracted PII between models. The diagonal shows the number of extracted PII for each model.

- **Utility**: We measure perplexity over a test set, similar to related works [29, 18, 101].
- **Membership Inference** (MI): We report the area under the receiver operating characteristic (ROC AUC) to measure sentence-level membership inference.
- **PII Extractability**: We report precision and recall on the set of extractable PII. Recall measures (i) how much PII is at risk of extraction and precision measures (ii) an attacker's confidence that a generated PII appears in the training dataset.
- **PII Reconstruction and Inference**: We report the top-1 accuracy of correctly predicting a PII for a context.

We refer to Appendix B.5.4 for formal definitions of these metrics.

3.5.5 PII Extraction

We first extract PII from LMs trained with and without DP using Algorithm 4. Then, we show that the estimated leakage (from Algorithm 5) matches the observed leakage, which allows an attacker to point-wise verify the extractability of a PII without sampling the model exhaustively (making our attack more efficient). We analyze different factors such as duplication rate, token length, and sample size for their effect on PII leakage.

Table 3.2 shows the measured precision and recall with an ablation over the LM's size. We sample on the order of 4m tokens from each target LM by issuing 15k queries requesting the LM to generate sequences with a length of 256 tokens from an empty prompt using top-k sampling with $k = 40$. We account for baseline leakage by excluding all PII occurring in a random sample of 13m tokens in 50k queries from a public model (see Section 3.4.4).

Model Size. We observe that GPT-2-Large recalls 23% of PII in the ECHR dataset with a precision of 30%. We conclude that in practice, an attacker can be confident that a generated PII is contained in the training dataset. The precision and recall decrease with the model's size. The smallest model (GPT-2-Small) has a significantly lower recall (only about 9%) at a similar precision as the large models (25%). In models trained with DP on ECHR, the precision and recall are similar for all model architectures, where we can extract about 3% of PII with a precision of 3%.

Duplication. Figure 3.5a shows that duplication of PII has a strong impact on their intractability. We group all PII with equal duplication counts in the training dataset and compute the frequency with which the model generates them. On all datasets, we observe a linear relationship between the number of occurrences of a piece of PII and the frequency with which they are leaked, i.e., PII occurring twice as often is expected to also leak twice

	GPT2-Small		GPT2-Medium		GPT2-Large	
	No DP	$\varepsilon = 8$	No DP	$\varepsilon = 8$	No DP	$\varepsilon = 8$
			ECHR			
Prec	24.91%	2.90%	28.05%	3.02%	29.56%	2.92%
Recall	9.44%	2.98%	12.97%	3.21%	22.96%	2.98%
			Enron			
Prec	33.86 %	9.37%	27.06%	12.05%	35.36%	11.57%
Recall	6.26%	2.29%	6.56%	2.07%	7.23%	2.31%
			Yelp-Health			
Prec	13.86%	8.31%	14.87%	6.32%	14.28%	7.67%
Recall	11.31%	5.02%	11.23%	5.22%	13.63%	6.51%

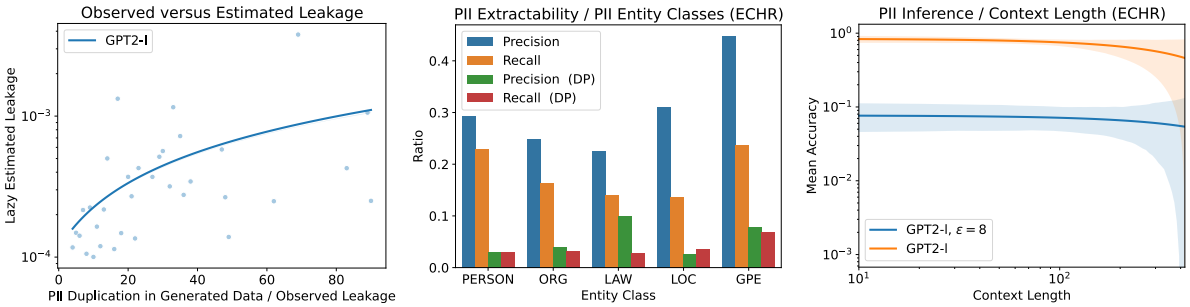
Table 3.2: Results for the observed PII extraction on ECHR (top rows), Enron (middle rows), and Yelp-Health (bottom rows) after sampling around 4m tokens across 15k queries.

as often. Our finding contrasts with the *superlinear* effect in sequence-level memorization observed by Kandpal et al. [111]. We note that Kandpal et al. [111] study models trained from scratch and arbitrary sequences, whereas our study focuses on fine-tuned models and PII. Further examination is necessary to fully comprehend the relationship between both findings. In DP models, we observe that the extractability of PII is consistently about an order of magnitude lower than in undefended models.

Token Length. We compare leaked PII by token length to evaluate whether longer PII sequences are less prone to extraction. In Figure 3.5b, we group all leaked PII by their token length and compute a mean count from the generated dataset. We observe that undefended models leak PII sequences containing many tokens, whereas long sequences are not leaked in DP models. In the range between 3-6 tokens, we observe that DP models leak about an order of magnitude fewer pieces of PII than undefended models.

Sample Size. Figure 3.5c shows precision and recall as a function of the number of sampled tokens on ECHR. The recall increases to 23%, whereas the precision consistently decreases from 50% at 500k tokens to 30% at 4m tokens. This indicates that PII with a high probability of generation by the LM is likely pieces of real PII from the dataset and, thus, vulnerable to extraction. An attacker who samples larger sets from the model can generate more PII, but at a lower precision which makes the attack less impactful.

Similarities between Generated PII. Figure 3.5d shows the intersection between sets of extracted PII across all LMs in a heatmap. We observe that (i) a subset of the most duplicated PII occurs in almost all and (ii) there is little similarity between which PII was leaked between models. In undefended models, we observe that PII, which occurs a single



(a) (b) (c)

Figure 3.6: Figure 3.6a shows the correlation between the observed and estimated leakage. Figure 3.6b shows the precision and recall for other entity classes on the ECHR dataset. Figure 3.6c shows the mean inference accuracy relative to the context length, which is the length of the combined prefix and suffix for a masked query.

time, can leak, which we never observe for DP models.

Estimated Extractability. Figure 3.6a shows that lazily estimated extractability correlates with observed leakage (i.e., the number of times a model generates the PII). This allows computing point-wise estimates for a target PII without searching through massive amounts of generated text. Our metric is imperfect, as some false negative outliers are incorrectly predicted as non-extractable despite appearing in the generated dataset.

Extraction of other PII Classes. We measure the PII extractability for other classes of PII, such as e-mails and phone numbers. The attacker can extract about 14.1% of law case numbers and 16.3% of mentioned organization names from an undefended model (shown in Figure 3.6b). In the DP model, we observe that an attacker can only extract 2.8% of law cases and 4.1% of organizations. For the Enron dataset, which contains long phone numbers, we never observe a single leaked real phone number in the DP model. However, we observe leakage of e-mail addresses (consisting of equally many tokens) that are typically correlated with a person's name.

3.5.6 PII Reconstruction

We compare our PII reconstruction attack from Algorithm 7 with the TAB attack [101]. Table 3.3 shows the results on ECHR, Enron, and Yelp-Health for the entity class 'person'. We sample 64 candidates and decode from the model using top-k sampling with $k = 40$.

	GPT2-Small		GPT2-Medium		GPT2-Large		GPT2-XL	
	No DP	$\varepsilon = 8$	No DP	$\varepsilon = 8$	No DP	$\varepsilon = 8$	No DP	$\varepsilon = 8$
ECHR(TAB)	0.78%	0.24%	1.21%	0.32%	5.81%	0.48%	4.30%	0.39%
ECHR (Ours	**2.25%**	0.44%	**3.36%**	0.87%	**18.27%**	0.55%	**13.11%**	0.41%
Enron (TAB)	0.59%	0.04%	0.67%	0.04%	1.75%	0.04%	2.19%	0.19%
Enron (Ours	**6.29%**	0.49%	**7.26%**	0.52%	**12.68%**	0.55%	**15.25%**	0.53%
Yelp-Health (TAB)	0.33%	0.24%	0.37%	0.14%	0.65%	0.12%	1.99%	0.12%
Yelp-Health (Ours	**0.42%**	0.32%	**1.31%**	0.32%	**1.69%**	0.35%	**6.40%**	0.36%

Table 3.3: Results of PII reconstruction attacks on the entity class "person". Bold numbers represent the best attack per dataset and LM. We compare our results with the TAB attack [101] on three datasets.

We observe that our reconstruction attack significantly outperforms the TAB attack on undefended models enabling the reconstruction of up to 10× more PII (in the GPT-2-Medium case on Enron).

Model Size. On ECHR and GPT-2-Large, TAB correctly reconstructs at least 5.81% of PII whereas our attack achieves 18.27%. This observation demonstrates that information in a sample's suffix provides a strong signal to reconstruct PII. On ECHR, our attack improves the baseline by at least 2.5×, on Enron we observe an improvement of at least 7.5× and on Yelp-Health our attack is at least about 3× more successful (except for GPT-2-Small where our attack improves only from 0.33% to 0.42%). The risk of reconstruction is much smaller in DP models ($\leq 1\%$), where our attack still improves the baseline in all cases, but we believe the leakage is too small for a practical attack. We observe that across all datasets, larger models are more vulnerable to PII reconstruction.

Context Size. On Enron, the advantage of our attack compared to TAB becomes more evident. E-mails in the Enron dataset typically mention the receiver of the e-mail at the beginning before any PII. For this reason, the TAB attack has only a small prefix to predict PII and cannot leverage the information contained in the e-mail's body. We observe that when the PII is in the set of candidates, it is predicted correctly about 70% of the time. However, our reconstruction attack often does not sample the correct candidate, which effectively limits our attack's success rate. We believe a method that samples candidates by incorporating information from the sample's suffix could improve our attack even further.

	ECHR		Enron		Yelp-Health	
	No DP	$\varepsilon = 8$	No DP	$\varepsilon = 8$	No DP	$\varepsilon = 8$
$\lvert\mathcal{C}\rvert = 100$	70.11%	8.32%	50.50%	3.78%	28.31%	4.29%
$\lvert\mathcal{C}\rvert = 500$	51.03%	3.71%	34.14%	1.92%	15.55%	1.86%

Table 3.4: Results of our PII inference attack on fine-tuned versions of GPT-2-Large. The values represent the attack's accuracy at inferring the correct PII out of $|\mathcal{C}|$ candidates.

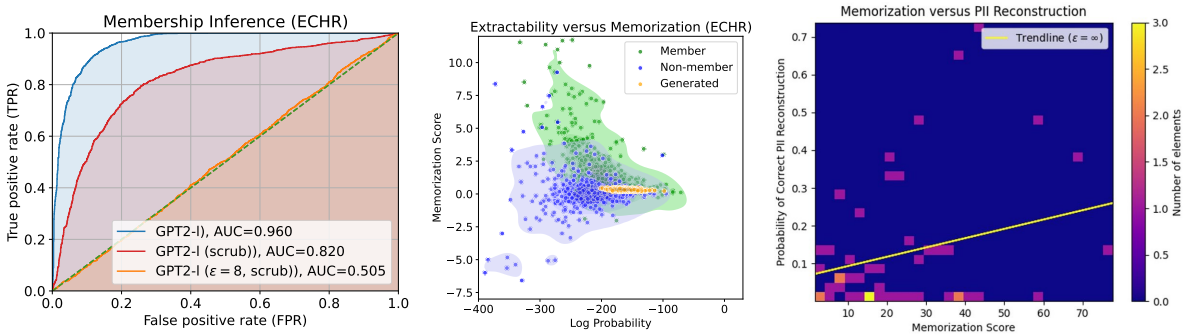
3.5.7 PII Inference

In PII inference, our attacker has access to (i) the anonymized training dataset and (ii) a list of candidate PII that also appear in the training dataset. For PII inference, we evaluate our attacks against the GPT-2-Large model on all three surveyed datasets with and without DP. Table 3.4 summarizes the results from our attack.

Results. We observe that in the undefended setting, an attacker can infer PII with an accuracy of 70% out of 100 candidates on ECHR, 50% on Enron, and 28% on Yelp-Health. We observe higher leakage on ECHR and Enron, which is likely because they have more structure than Yelp reviews. Pieces of PII are mentioned repeatedly in similarly structured sentences, which causes higher PII leakage. The undefended setting enables practical attacks where the attacker can be confident about the results. In the DP setting, an attacker can achieve an accuracy of about 8% given 100 candidates and about 4% in 500 candidates on ECHR. Although leakage in DP models is small, we believe our experiments demonstrate that DP does not fully protect against PII leakage in a practical setting against an informed attacker. Figure 3.6c shows that inferring PII in very large contexts slightly *worsens* the accuracy. This is likely because the expected memorization per token is lower in samples containing many tokens.

3.5.8 Membership Inference and PII Leakage

We employ a shadow model membership inference attack [205] to empirically evaluate the relationship between sentence-level membership inference and PII leakage. In this attack, the adversary trains shadow models on datasets sampled from the same data distribution as the target model. The adversary then calculates the difference between the perplexity (i.e., PPL) of the target sentence w.r.t. the target and shadow models and uses this as a score to decide if the sentence was a training member or not.



(a) (b) (c)

Figure 3.7: Connecting sentence-level membership inference with PII reconstruction in GPT-2-Large. 3.7a shows the ROC curve against our fine-tuned model using a shadow model attack on ECHR. 3.7b shows that the memorization score of generated sequences is nearly zero, and 3.7c shows that the memorization score correlates with the probability of correct PII reconstruction.

Figure 3.7a shows the ROC curve of this MI attack against an undefended model, a model trained after scrubbing, and a model trained with differential privacy after scrubbing. Expectedly, we observe that scrubbing PII mitigates membership inference but is nowhere as effective as DP. The attack achieves an AUC score of 0.96, 0.82, and 0.51 against undefended, scrubbed, and DP & scrubbed models, respectively.

Connection between MI and PII Leakage. Algorithm 9 shows the sentence-level MI game of Yeom et al. [244] alongside an indistinguishability variant of the PII Inference game in Algorithm 8. This corresponds to the case $m = 1$ when the adversary would be given a pair of candidates for the target PII, with the only exception that in Algorithm 9, the adversary is given just one of the candidates, depending on a Bernoulli random variable b. The difference between one or two candidates is inessential and analogous variants of sentence-level MI have been considered before [97]. The essential difference between the MI and PII Inference games is that in the former, the adversary has to distinguish between two sentences sampled from $\mathcal{D}$, while in the PII Inference game, it has to distinguish between two sentences differing only on a piece of PII. This means that to leverage an MI attack for PII reconstruction or inference, it has to be strong enough to distinguish between member and non-member sentences differing in a few tokens. In contrast, a PII Inference attack can be directly turned into an MI attack against sentences containing a piece of PII.

PII Reconstruction is similarly analogous to attribute inference (where the target

Algorithm 9 Sentence-level MI (lines enclosed in solid box) vs. PII Inference (lines enclosed in dashed box).

experiment IND-INFERENCE($\mathcal{T}, \mathcal{D}, n, \mathcal{A}$)
  $b \sim \{0, 1\}$
  $D \sim \mathcal{D}^n$
  $\theta \leftarrow \mathcal{T}(D)$
  $S_0 \sim D$
  $S_1 \sim \mathcal{D}$
  $\tilde{b} \leftarrow \mathcal{A}(\mathcal{T}, \mathcal{D}, n, \mathcal{O}_\theta(\cdot), S_b)$
  $S \sim \{S \in D | \text{EXTRACT}(S) \neq \emptyset\}$
  $C_0 \sim \text{EXTRACT}(S)$
  $C_1 \sim \mathcal{E}$
  $\tilde{b} \leftarrow \mathcal{A}(\mathcal{T}, \mathcal{D}, n, \mathcal{O}_\theta(\cdot), \text{SCRUB}(\text{SPLIT}(S, C_0)), C_b)$

attribute is the missing piece of PII). PII Reconstruction can be linked to MI the same way as attribute inference was linked to MI by Yeom et al. [244]. Our empirical results show that they are indeed correlated. For instance, Figures 3.7b and 3.7c respectively contrast MI with PII extractability and reconstruction attacks. Figure 3.7b shows the likelihood of the model producing a member on the x-axis versus the memorization score on the y-axis. We observe that samples generated from empty prompts are not memorized, meaning that they likely contain few useful signals for MI. Figure 3.7c shows the memorization score relative to our reconstruction attack. We observe a positive correlation between a sentence's memorization score and the success rate of our PII reconstruction attack. This means that sentences that are vulnerable to the MI attack are usually vulnerable to the PII reconstruction attack and vice-versa.

3.5.9 Summary of Results

Table 3.5 summarizes the privacy-utility trade-off for GPT-2-Large model when fine-tuned on ECHR dataset. We enumerate our key results below:

- Undefended models are highly vulnerable to all privacy attacks, including membership inference and PII extraction, reconstruction, and inference. For PII risks, larger model sizes and higher duplication counts increase the risk of leakage.
- Our results show that the threat of reconstructing PII has been underestimated, and we demonstrate up to an order of magnitude higher leakage than prior work.

	Undefended	DP	Scrub	DP + Scrub
Test Perplexity	9	14	16	16
Extract Precision	30%	3%	0%	0%
Extract Recall	23%	3%	0%	0%
Reconstruction Acc.	18%	1%	0%	0%
Inference Acc. ($\lvert\mathcal{C}\rvert = 100$)	70%	8%	1%	1%
MI AUC	0.96	0.5	0.82	0.5

Table 3.5: Our results on ECHR for GPT-2-Large summarize the privacy-utility trade-off. We show the undefended model's perplexity with/without masking generated PII. The undefended model has the lowest perplexity but the highest leakage. DP with $\epsilon = 8$ mitigates MI and (partially) PII leakage. Scrubbing only prevents PII leakage. DP with scrubbing mitigates all the privacy attacks but suffers from utility degradation.

- Empirically, we observe that differential privacy significantly bounds leakage from PII reconstructions relative to undefended models.
- DP does not completely eliminate leakage from PII inference and PII extraction. We demonstrate that an attacker can infer PII with up to 10% accuracy (given 100 candidates) in a practical setting.
- We find that DP and (aggressive) PII scrubbing limit the LM's utility, motivating the search for defenses with better empirical privacy/utility trade-offs.

3.6 Discussion and Limitations

Below, we discuss our methodology and identify further research motivated by our findings. We first discuss the applicability of our methodology to sensitive information other than PII and potential extensions to our attacks exploiting semantic similarity and associations in the training dataset. We then describe how masked language models fare compared to autoregressive models and identify further research motivated by our findings: best combining DP training and scrubbing, optimizing attacks for other leakage metrics, and the need for better benchmarks.

General Applicability. We focus on defining metrics, game-based definitions, and tractable formulas for evaluating the leakage of sensitive sequences of tokens categorized as PII. That said, we emphasize that our methodology generally applies to any notion of sensitive input. As long as one has an effective method to correctly identify inputs deemed sensitive, our methodology can be adapted to measure the protection offered by existing ML pipelines in mitigating the leakage

of *any* sensitive information. In practice, it is often hard to draw a clear boundary around what constitutes sensitive information, which is an important but orthogonal problem.

Semantic Similarity. We consider verbatim matches of PII tokens as leakage; however, our methods can be adapted to account for both syntactic and semantic similarity. For example, "John Doe" and "J. Doe" could be inferred to be the same person. Similarly, PII reconstruction and PII inference attacks can employ contexts with similar meanings to improve attack results.

Advanced Attacks. We consider leakage of PII sequences in isolation, irrespective of the context where it appears and other extracted PII. Extracted PII sequences can be further leveraged in advanced attacks that explore associations among them and reveal additional private information about the training dataset, thereby enabling linkability attacks.

Utility-preserving Scrubbing. Our empirical evaluation demonstrates that differential privacy is partially effective in mitigating leakage of PII. Based on this observation, existing scrubbing techniques can be adapted to consider the partial protection offered by DP and heuristically scrub only PII that remains unprotected (e.g., because it occurs many times). Such a DP-informed scrubbing would improve model utility while maintaining a privacy level equivalent to a naive combination of DP training and scrubbing.

Comparison to Masked Language Models. Prior work has explored PII reconstruction in the clinical health setting [130, 228] with masked language models (MLMs) based on the BERT architecture [48]. MLMs are trained to reconstruct a word in a masked query, which is equivalent to the PII reconstruction task in Equation (3.4). During training, MLMs optimize the masked word's probability conditioned on the prefix and suffix, compared to GPT-2, which is auto-regressive and can only be conditioned on the prefix. A service deploying an MLM enables trivial attacks to query the most likely replacement for a single mask conditioned on *the entire sentence*, unlike GPT-2 models. Our attacks imitate this functionality in GPT-2 models to reconstruct PII.

Attacks on GPT-2 and similar models employed for text generation are potentially a greater threat. Autoregressive models typically expose a next-word prediction API, whereas BERT-like models are often used for downstream tasks, such as text classification, with less revealing APIs. We state that Equation (3.4) is intractable, which is also true for existing MLMs since an attacker does not know the number of tokens in the PII and has to perform a general constrained search.

Need for Better Benchmarks. In conducting this research, we realized the shortage of good benchmark datasets for measuring PII leakage. A notable exception is the Text Anonymization Benchmark of Pilán et al. [177] with human-annotated PII. However, we found this dataset to be too small for fine-tuning models with DP-SGD: the dataset contains a subset of $1,268$ out of the estimated $11\,500$ cases from the original ECHR dataset [31]. With such few records, we could not fine-tune models with both reasonable privacy (i.e., low ε) and utility (low perplexity).

Our work motivates the need for better benchmarks for evaluating attacks and mitigations against PII leakage in trained models, in addition to evaluating text anonymization techniques

at a dataset level. In performing our evaluation, we rely on off-the-shelf text anonymization tools powered by NER models to tag PII in generated sentences and leverage them in computing our leakage metrics. As a consequence of this approach, our metrics can capture leakage that is dependent on the quality of the tools we use. Assuming that NER models and other techniques used in these tools are prone to error, our results provide a lower bound on the leakage.

High-Precison/Low-Recall Attacks. Our attacks evaluate PII leakage using average-case metrics and provide an overview of the threat of PII leakage in LMs. We did not consider high-precision/low-recall attacks [28], and further research is needed to explore their effectiveness.

Limitations. Due to the lack of extensive, annotated datasets for studying PII leakage in LMs, we employ the same NER model for both scrubbing and measuring PII leakage. Pilán et al. [177] demonstrate that a fine-tuned NER model with ground-truth PII annotations achieves recall rates between 84-93%, decreasing to 77% without annotations. Since scrubbing cannot remove all PII and we show PII leakage empirically, we conclude that scrubbing cannot fully prevent PII leakage. Further research is required to expand our findings to other LMs and datasets.

3.7 Related Work

We discuss prior work on attacks inferring private data and defenses to mitigate the leakage.

Extraction of Training Data. There is extensive work studying how large language models memorize training data and attacks inferring information under various threat models. Research has shown the feasibility of extracting different types of information, including individual sentences [27, 101], inserted canaries [26, 172, 252] as well as n-grams [155]. Prior work studied the leakage of PII in masked language models [129, 228], large language models [95, 188] and Smart Reply classification models [107]. In addition to demonstrating that language models leak training data, other efforts focus on understanding the causes of such leakage. Jagielski et al. [105] explore the causes of memorization such as training data ordering, i.e., samples can have different privacy risks independent of their content. Tirumala et al. [222] study the effect of memorization across variables such as dataset size, learning rate, and model size.

Related work focuses mainly on understanding the leakage in the absence of mitigations. In contrast, we are the first to evaluate the interplay of defenses such as PII scrubbing and differential privacy in an end-to-end training pipeline. Existing work on training data extraction focuses on *any* type of memorization [27] in public pre-trained LMs or models trained from scratch [111], whereas we focus on leakage of PII on fine-tuned LMs given the context where it appears (prefix and suffix), no context, or a list of PII candidates.

Mitigations. Several works have proposed solutions to mitigate the leakage of private information, mainly based on differential privacy (DP) guarantees in the training pipeline. Yu et al. [247] and Li et al. [132] propose an efficient method for differential-privately fine-tuning

LMs on private data. Shi et al. [204] propose selective DP—where DP is only applied to samples containing sensitive information to limit utility degradation. Stock et al. [212] study canary extraction attacks against models fine-tuned from GPT-2 using DP-SGD. Closer to our work, Zhao et al. [260] propose combining de-duplication, redaction, and DP-SGD to mitigate PII leakage. It would be interesting to study how this fares with respect to the risks and metrics we present.

3.8 Conclusion

Our work explores privacy/utility trade-offs of using defenses such as PII scrubbing and Differentially Private training when fine-tuning language models. We focus on measuring PII leakage from the training data with respect to three different adversary goals: PII extraction, reconstruction, and inference, and provide game-based definitions and leakage metrics for them. Our findings show that differential privacy is useful in mitigating PII leakage by a large extent but cannot completely eliminate it on its own. Traditional data curation approaches such as PII scrubbing are a crucial part of the training pipeline and are still necessary to achieve an appropriate level of protection. We advocate for the design of less aggressive PII scrubbing techniques that consider the protection afforded by DP and achieve a better privacy/utility trade-off.

Chapter 4

SoK: How Robust is Watermarking for Deep Image Classification?

This chapter is based on work that appeared at the 2022 IEEE Symposium on Security and Privacy (S&P) [148]. We systematically evaluate the robustness of eleven proposed white-box and black-box watermarking methods to monitor and control unauthorized model redistribution. Our results highlight that the known watermarking methods have only limited robustness against handcrafted adaptive attacks and that it is possible to combine adaptive attacks and create *dominant* attacks that remove any surveyed watermark. We propose guidelines to assess the robustness of watermarking methods more comprehensively in the future.

4.1 Motivation

DNN watermarking [227] is a method designed to detect surrogate models given black-box API query access. Watermarking embeds a message into a model that is later extractable using a secret key. Developing DNN watermarking methods is an active area of research studied by large corporations such as Microsoft [192], Google [3] and IBM [255]. Robustness is a core security property of watermarking, which states that an attacker cannot derive accurate surrogate models from access to the source model without a watermark. Watermarking methods that are robust against such *watermark evasion* attacks are needed to deter redistribution by adversaries. Claimed security properties of some existing watermarking methods [3, 255] had been broken by novel attacks [202, 140, 233], but it is unclear how these attacks generalize to other watermarks.

We survey 29 methods from the literature that (i) are known evasion attacks, such as weight pruning [264] or knowledge distillation [86], (ii) derive surrogate models but have not been evaluated

as evasion attacks, and (iii) novel evasion attacks. An evasion attack is *effective* if the surrogate model has a high test accuracy and does not retain the watermark. It is *efficient* if resources required to run the attack, such as its runtime, are small compared to retraining a model from scratch. We measure both effectiveness and efficiency. In our taxonomy, we categorize attacks into (i) model modification, (ii) input preprocessing, and (iii) model extraction. Model modification and input preprocessing modify the source model or its input, whereas model extraction trains a different surrogate model by distilling knowledge from the source model.

We survey eleven[1] recently proposed watermarking methods [227, 3, 33, 34, 109, 127, 192, 216, 255] from the literature that claim robustness. Most of these methods do not specify whether their definition of robustness includes model extraction [255, 227, 192, 33, 34], one method restricts the runtime of the attacker [3] and the remaining methods claim robustness against any evasion attack [127, 109, 216]. In this chapter, we evaluate robustness against any evasion attack and demonstrate whether an attack is efficient by showing its runtime. Our taxonomy categorizes these watermarking methods into (i) model-independent, (ii) model-dependent, (iii) parameter encoding, and (iv) active watermarking methods.

Our new Watermark-Robustness-Toolbox (WRT) implements all watermarking methods and evasion attacks evaluated in this chapter. We validate the robustness of each method against each evasion attack. Our evaluation includes an ablation study over multiple sets of parameters for each watermarking method and evasion attack. The defender and attacker use a zero-sum game to choose the best parameter set for their method, constituting the Nash equilibrium. We call a method robust if the defender can choose a set of parameters so that no evasion attack is effective. Our study analyzes the robustness of watermarking methods and the effectiveness and efficiency of evasion attacks. We also study the robustness of watermarking method categories against categories of evasion attacks to identify the most effective category of attacks that should be used to evaluate the robustness of watermarking methods from a given category in the future.

Our empirical evaluations are performed on large datasets to emphasize the practical relevance of our work. The experiments span the image classification datasets CIFAR-10 [121] and ImageNet [47]. ImageNet contains over 1.2 million training images from 1k classes and is a widely accepted benchmark to measure the performance of machine learning models [183].

Our study shows that none of the surveyed watermarking methods is robust against all evasion attacks. However, we also find that none of the attacks from the literature removes all watermarks. We propose new *combined* attacks that evade all investigated watermarks while preserving a high test accuracy in surrogate models. Our study also shows that robustness should be verified against a more extensive set of attacks and on a larger number of datasets. Open-source implementations of watermarking methods and evasion attacks enhance the scientific study of a method's robustness. Towards this goal, we make our new Watermark-Robustness-Toolbox (WRT)[2] and a complete

[1] Zhang et al. [255] propose three different methods.

[2] https://github.com/dnn-security/Watermark-Robustness-Toolbox

Defense	Category	Verification
Adi [3]	Model Independent	Black-Box
Content [255], Noise [255], Unrelated [255]	Model Independent	Black-Box
Jia [109], Frontier Stitching [127]	Model Dependent	Black-box
Blackmarks [34]	Model Dependent	Black-box
Uchida [227], Deepsigns [192], DeepMarks [33]	Parameter Encoding	White-box
DAWN [216]	Active	Black-box

Table 4.1: All surveyed watermarking methods (see Section 4.2 for full descriptions).

dataset of evaluation results publicly available with documentation, which allows to independently verify our conclusions.

4.2 Watermarking Methods

This section describes the eleven surveyed watermarking methods and the parameters used for our ablation study to test their robustness. For simplicity, we refer to a watermarking method by the first author's name unless it is already known under a different name.

Adi [3] uses a secret watermarking key consisting of abstract, out-of-distribution images. A label for an image is randomly sampled over all classes, excluding the image's true label. The embedding consists of fine-tuning the model on the watermarking key. A message is extracted from a surrogate model by requesting labels for the watermarking key images.

Zhang et al. [255] proposes three different methods, referred to as *Content*, *Noise* and *Unrelated*. These three methods differ only in their selection of the watermarking keys. The watermarking keys are selected as follows.

- **Content**: The secret watermarking key is images sampled from one *source* class that is perturbed by a secret, additive mask (e.g., a white square covering part of the image). The same mask is used for all watermarking keys.
- **Noise**: Gaussian noise is added to images from the source class during training.
- **Unrelated**: The watermarking key is images sampled from a different domain that is unrelated to the source model's domain. For example, if a model is trained to classify animal species, the watermarking key could contain images of automobiles.

All watermarking key images are labeled with the same *target* class that is sampled randomly. The embedding and extraction are the same as Adi et al. [3] for all three methods.

4.2.1 Model Dependent

Frontier-Stitching [127] uses a watermarking key consisting of adversarial examples [213]. Adversarial examples are images that have been modified (often imperceptibly) to trigger a misclassification when a DNN predicts the image's label. The authors generate these adversarial examples using the Fast Gradient Method (FGM) [77] and the pre-trained source model. This method has a given probability of failure, meaning that its output is not adversarial and is correctly classified by the DNN. The watermarking key is composed of such *false* adversarial examples and equally many *true* adversarial examples. All adversarial examples are labeled by the ground-truth label in the watermarking key. The embedding and extraction process is the same as Adi.

Blackmarks [34] also relies on adversarial examples, similar to Frontier-Stitching. The authors propose a pre-processing step that clusters all class labels into two groups using k-means clustering on the pre-trained source model's logit activations. These clusters will be used to encode bits. The idea is to randomly select images from one cluster and use a targeted adversarial attack so that the source model predicts any class from the other cluster. During embedding, an additional loss term is introduced that minimizes the bit error rate between the predicted cluster and the assigned cluster of the trigger. The authors also present a method to mitigate unintended *transferability* of the watermarking key images. An adversarial example is transferable if it is adversarial to many models, i.e., it is not only adversarial to the source model for which it has been generated. The embedding and extraction is similar to Adi, except that a new loss term is added during embedding as described above.

Jia [109] proposes using the soft nearest neighbor loss (SNNL) [119, 194] as an additional loss during training to *entangle* feature representations of the watermark with the training data. Two groups of size N are entangled if the average distance between their elements is lower than the average distance within each group, which the SNNL measures. A temperature parameter T controls the weight of short and long distances between two points for the total loss and can be tuned during training.

$$L_{\text{SNNL}}(x, y, T) = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\sum_{j=1, j\neq i, y_i=y_j}^{N}\exp(\frac{-||x_i-x_j||^2)}{T})}{\sum_{k=1,k\neq i}^{N}\exp\frac{-||x_i-x_k||^2}{T}}\right) \tag{4.1}$$

The watermark generation defines two groups of elements belonging to a source and a target class. These classes are chosen by computing two classes with the highest similarity using the pre-trained source model's hidden activations. All elements from a source class are modified by adding a (secret) trigger pattern and changing their label to the target class, similar to the Content watermark described earlier. The trigger is added at the location where the gradient (through the source model) with respect to the SNNL is highest.

4.2.2 Active

DAWN [216] proposes a method where a low proportion r of all predictions is randomly relabeled and added to the watermarking key. The authors implement a method that recognizes similar samples (as perceived by the source model) and returns the same label when the attacker tries to query a sample twice. This similarity detection is implemented by using the activation of some target layer of the source model. The watermarking key consists of images from the attacker's dataset, and labels are assigned randomly.

4.2.3 Parameter Encoding

Uchida [227] propose embedding a message into the weights of some *target* convolutional layer. The idea is to add an *embedding* loss during training that regularizes the model and is minimized when the message can be extracted successfully and with a large margin. Let $W \in \mathbb{R}^{n \times c \times w \times}$ be the convolutional filters of a target layer, where n is the number of filters, c are the number of channels, and w, h are the width and height of each filter. The method computes a mean filter $\bar{W} = \frac{1}{n} \sum_{i=1..n} W_i$ to deal with the permutation invariance of the filters. This mean filter is flattened $\hat{W} \in \mathbb{R}^{(c \cdot w \cdot h)}$ and a random projection matrix $A \in \mathbb{R}^{k \times (c \cdot w \cdot h)}$ is sampled, where k is the desired key length. The embedding consists of fine-tuning using the embedding loss described above. A message can be extracted by computing $m' = A\hat{W}^T$ and applying the following rule.

$$m_i = \begin{cases} 1 & m'_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

DeepMarks [33][3] proposes a similar embedding method as in Uchida, with a notable difference in how a message is extracted. The authors compute the dot product between $A\hat{W}$ and the owner's signature, which returns a correlation score between -1.0 and 1.0. A correlation of 1.0 is interpreted as a perfectly retained watermark, whereas a correlation ≤ 0 corresponds to a watermark accuracy of zero.

DeepSigns [192] proposes a watermarking method that uses the activations of some target layer of the source model to encode a message. The intuition is that features extracted from samples belonging to the same *source* class form clusters whose properties can be used to embed watermarking information. In the author's paper, clusters are modeled using a Gaussian Mixture Model, whereby each feature cluster c_i is described by a mean μ_i and a standard deviation σ_i. An embedding loss is added during training that modifies each cluster's mean and allows embedding n bits of information per cluster, i.e., for m clusters, we can embed a message with $m \cdot n$ bits. A

[3] DeepMarks is labeled as a fingerprint by the authors, but since it modifies the model by embedding a message, it is a watermark as per our definition.

random projection matrix is sampled that is multiplied with the source model's activation on the target layer given the watermarking images. The result is a binary vector that is then activated by the Sigmoid function. Elements of this binary vector can be interpreted as binary digits by comparing their values with a threshold, similar to Uchida's extraction.

4.3 Removal Attacks

This section provides summaries of the watermark removal attacks surveyed in this chapter. The list contains (i) known watermark removal attacks, (ii) attacks that have not been evaluated as removal attacks, and (iii) novel removal attacks.

4.3.1 Input Preprocessing

Input Reconstruction [136] uses an autoencoder[4] [166] to compress and reconstruct images before passing them to the surrogate model. The idea is that an autoencoder trained on non-watermarked images will fail at reconstructing artifacts in the image it has never seen before (such as additive masks or random noise) while preserving the remaining image's content.

Input Noising [253] adds Gaussian noise with zero mean and some standard deviation to the entire image.

Input Quantization [134] quantizes the pixel values for an input images. For a given number of bits b, the input space is quantized into 2^b evenly spaced intervals. Every pixel of the input image is set to the mean of its interval.

Input Smoothing [241] convolves some kernel over the image, which makes the image appear more blurry. Possible kernels are a mean, median, and Gaussian kernel. The image appears more blurry after applying the convolution.

Input Flipping flips an image along its horizontal axis.

JPEG Compression [57] is similar to input reconstruction, but instead of using an autoencoder, the image is compressed through the JPEG compression algorithm.

Feature Squeezing [241] is similar to input quantization, except that input values are rounded to the nearest quanta, and the quanta values are chosen to be multiples of 0.5^k for some $k \in \mathbb{N}$.

[4] https://github.com/foamliu/Autoencoder

Attack	Category	Source Model Access	Data
Input Reconstruction [136], JPEG Compression [57], Input Quantization [134], Input Smoothing [241], Input Noising [253], Input Flipping, Feature Squeezing [241]	Input Preprocessing	White-box	None
Adversarial Training [151], Fine-Tuning (RTLL, RTAL) [227], Weight Quantization [96], Label Smoothing [215], Fine Pruning [139], Feature Permutation (Ours), Weight Pruning [264], Weight Shifting (Ours), Neural Cleanse [232], Regularization [202], Neural Laundering [5]	Model Modification	White-box	Domain
Overwriting [227], Fine-Tuning (FTLL, FTAL) [227]	Model Modification	White-box	Labeled
Knockoff Nets [169]	Model Extraction	Black-box	Transfer
Distillation [86]	Model Extraction	White-box	Domain
Transfer Learning, Retraining [224], Smooth Retraining (Ours), Cross-Architecture Retraining (Ours), Adversarial Training (From Scratch) [151]	Model Extraction	Black-box	Domain

Table 4.2: An overview of all watermark removal attacks evaluated in this Chapter 4 and the attacker's capabilities. 'Category' refers to the categories from Section 2.3.1, 'Source Model Access' refers to the attacker's access to the watermarked model, and 'Data' refers to the dataset requirements of the attack. 'None' means no data is required, 'Domain' means the attack requires in-domain data, 'Labeled' means the attack requires in-domain labeled data (otherwise, it is always unlabeled), and 'Transfer' requires out-of-domain data. We instantiate these attacks against black-box and white-box watermarks in Chapter 4.

4.3.2 Model Modification

Adversarial Training [151] is a method during training to increase a model's robustness to adversarial examples. A random subset of the training dataset is perturbed using the Projected Gradient Descent (PGD) [151] adversarial attack, and these examples are then injected into the training dataset with the ground-truth labels.

Feature Permutation. DNNs are invariant to feature permutations, meaning that neurons in a hidden layer can be permuted without affecting the model's functionality. We use (random) feature permutation as an adaptive attack designed specifically against Deepsigns [192], which encodes the message into the activations of hidden layers.

Fine-Pruning [139] is designed for *backdoor removal* that first prunes dormant neurons and then fine-tunes the model to regain the drop in test accuracy. The idea is that neurons that are never highly activated for benign inputs likely implement the functionality of the backdoor. Such neurons are eliminated by setting their activation to zero.

Fine-Tuning [227] as a model stealing attack refers to a set of attacks that first apply a transformation to the model, followed by fine-tuning.

- Fine-Tune All Layers (**FTAL**). All weights are fine-tuned.
- Fine-Tune Last Layer (**FTLL**). All but the last layer's weights are frozen while the model is fine-tuned.
- Retrain All Layers (**RTAL**). The last layer's weights are re-initialized, and all weights are fine-tuned.
- Retrain Last Layer (**RTLL**). The last layer's weights are re-initialized, and only that layer's weights are fine-tuned.

RTAL and RTLL use predicted labels, whereas FTAL and FTLL use ground-truth labels (otherwise, gradients are zero).

Overwriting [227] embeds a watermark using the same watermarking method but a different watermarking key.

Label Smoothing [215] is a regularization method that computes the weighted mean of a uniform distribution over all labels with a one-hot or (in our case) predicted label by the source model. The idea is to regularize the model by making classes other than the ground-truth class appear likely.

Regularization [202] is a two-phased attack that strongly regularizes the model in the first phase, leading to a drop in test accuracy, which is compensated by fine-tuning the model in the

second phase. The idea of the regularization phase is to move the model's parameters far away from their origin, while the fine-tuning phase finds a (different) local minima.

Neural Cleanse [232] is an attack designed for *backdoor removal*. It first reverse-engineers the watermark trigger and then removes ('unlearns') the trigger from the model. When a trigger has been reverse-engineered, (i) it can be unlearned through fine-tuning on different labels or (ii) removed by pruning most activated neurons in some layer.

Neural Laundering [5] is an extension of Neural Cleanse, specifically designed against model-independent watermarking methods. The authors re-use the reverse-engineering trigger step from Neural Cleanse and then iteratively prune weights triggered by the backdoor.

Weight Pruning [264] randomly prunes the weights in the source model until a given sparsity ρ is reached in each layer.

Weight Shifting is a novel, adapted attack that we design specifically against white-box watermarking methods. The idea is to apply a small perturbation to all filters of each convolutional layer in the network, followed by fine-tuning the model to regain the loss in test accuracy. We design weight shifting as an efficient and effective watermark removal attack specifically against Uchida [227] and Deepmarks [33].

We explain the attack's idea in the example of Uchida, but a similar intuition holds for Deepmarks, where the watermark verification procedure is similar. A weakness of Uchida exploited by weight shifting is that the attacker knows that if all filters were inverted, i.e. $W_i' = -W_i$, then the watermark accuracy would be zero. We cannot directly invert all filters, as the model experiences a significant drop in test accuracy. Hence, we construct a 'softer' version of the attack that only moves each filter in the direction of the inverse mean multiplied by some constant weight parameter $\lambda_1 \in \mathbb{R}$. We additionally add small random Gaussian noise to each filter to encourage the network to find slightly different filters in the fine-tuning phase.

Our attack can be formalized by the function $S(W; \lambda_1, \lambda_2)$, which takes as input a set of filters W and outputs a shifted set of filters W'. The parameter λ_1, λ_2 trade off the attack's efficiency with its effectiveness. Let A be a random normal matrix of the same shape as each filter W_i with a variance equivalent to the variance over all filters for a convolutional layer and a mean of zero. Shifted weights for each convolutional layer can be computed by applying the following function.

$$\mathrm{S}(W; \lambda)_i = W_i - \frac{\lambda_1}{n} \sum_{j=1..n} W_j - \lambda_2 A \tag{4.3}$$

Weight Quantization [96] compresses a model by quantizing its weights. This attack is equivalent to Input Quantization, but quantizes the model's weights rather than the input image.

4.3.3 Model Extraction

Retraining [224] is a method to train a surrogate model from API access to a source model. The surrogate model is trained from scratch on the softmax output of the source model. Cross-Architecture retraining is equivalent to retraining, but the surrogate and source model's architecture differs.

Cross-Architecture Retraining is equivalent to retraining, but the surrogate model uses a different DNN architecture than the source model. The idea is to increase the dissimilarity to the source model, which intuitively decreases the likelihood that the watermark is transferred to the surrogate model.

Distillation [86] is a method originally designed as a model compression technique that distills knowledge from a (larger) teacher DNN to a (smaller) student DNN. The idea is to scale the teacher DNN's logit activations before applying the softmax function, which increases the prediction's entropy. The student is trained using this scaled prediction vector.

Smooth Retraining re-trains a surrogate model on smoothed labels obtained from querying the source model for multiple variations of the same image. For each query, a random affine transformation (e.g., random cropping) is applied to the image, and the mean of all received labels is computed as the final label. We design smooth retraining as an adaptive attack against the active watermarking method DAWN. The intuition is that if DAWN responds with a false label for one image, variations of the same image have a high probability of receiving the label predicted by the source model.

Knockoff Nets [169] is a method for training a surrogate model from scratch using only data from a different domain, referred to as the *transfer set*. We implement the random selection approach proposed by the authors.

Transfer Learning [223] is similar to Retraining, but instead of using a randomly initialized model, the attack assumes access to a pre-trained surrogate model from a different domain. This surrogate is fine-tuned on data from the source model's domain using labels predicted by the source model. The pre-trained surrogate model has already learned semantically meaningful filters for a different task. A common technique is to freeze the surrogate model's weights that are located in the lower layers (i.e., not updating them during training) to reduce the training time. Another option is to reduce the learning rates for lower layers to encourage the reuse of these filters.

Adversarial Training (from scratch) [151] is equivalent to adversarial training described earlier, except that the attacker trains the surrogate model from scratch.

4.4 Threat Model

Our study covers many different watermarking methods and evasion attacks that assume different adversary models. For example, model modification attacks require white-box access to the source model, whereas many model extraction attacks only require black-box access. We present a generic adversary model for any watermarking method and watermark evasion attack. Tables 4.1 and 4.2 summarize the defender's and attacker's capabilities for all methods surveyed in this chapter.

4.4.1 Attacker's Goals

The attacker's primary goal is to derive a surrogate model from access to the source model (i) that does not contain the watermark and (ii) is similarly accurate as the source model. A secondary goal is to reduce resources needed for the evasion attack, such as the attack's computation time. Algorithm 10 formalizes the watermark verification game given a dataset D, a secret watermarking key τ (only known to the defender) an attack $\mathcal{A}$ and the attacker's access to the source model O_A and the defender's access to the suspect model O for the verification of the watermark.

Algorithm 10 Watermark Verification Game

1: **experiment** $\text{WMVERIFY}(D, \lambda, \kappa, \delta, \mathcal{A}, O_A, O)$
2: $\quad\theta_0 \leftarrow \mathcal{T}(D)$ ▷ *train a classifier*
3: $\quad\tau \leftarrow \text{KEYGEN}(\lambda; \theta_0)$
4: $\quad m \sim \mathcal{M}$ ▷ *sample a message uniformly at random*
5: $\quad\theta_1 \leftarrow \text{EMBED}(\theta_0, \tau, m)$
6: $\quad b \sim \{0, 1\}$
7: $\quad\hat{\theta}_b \leftarrow \mathcal{A}(O_A(\theta_b))$ ▷ *watermark removal attack returns a surrogate model*
8: $\quad c \leftarrow [\![\text{VERIFY}(O(\hat{\theta}_b), \tau, m) < \kappa]\!]$ ▷ *detect the presence of the watermark*
9: $\quad$**return** $[\![b = c]\!] \cdot [\![(\text{ACC}(\theta, D_{\text{test}}) - \text{ACC}(\theta_b, D_{\text{test}})) < \delta]\!]$

The game begins by training an unmarked source model and generates a watermarking key (lines 2-3). Then it samples a message and embeds the watermark into the model (lines 4-5). A fair coin flip determines whether the attacker can access an unmarked or marked model (lines 6-7). The verifier accesses the model after the attack and detects the watermark with a p-value of at most κ (lines 7-8). Finally, the game returns the gain of the verifier by (i) ensuring that the accuracy degradation is at most δ and (ii) the verifier's prediction about the presence of a watermark was correct (line 10). The game returns 1 if the verifier correctly predicts the presence of a watermark or its absence. A watermarking method lacks robustness or integrity if the expected gain of the verifier in Algorithm 10 is low.

4.5 Measured Quantities

We present a methodology to assess a watermarking method's robustness. We describe a generally applicable method to empirically determine a statistical test for a watermarking method's decision threshold for each watermarking method and dataset. While Algorithm 10 specifies a security game, we must instantiate the strongest possible attacker to comprehensively study robustness. We introduce the *Nash equilibrium* as a method to determine this best configuration of parameters in an adversarial setting. The Nash equilibrium is computed over multiple parameter configurations for each method and evasion attack. We aim to empirically determine whether watermarking methods are robust to any evasion attacks.

4.5.1 Measurements

First, we describe the quantities measured for each experiment and our processing of these measurements to ensure comparability between watermarking methods.

Embedding and Stealing Losses. We measure the *embedding* and *stealing losses* as differences in test accuracy between an unmarked and a marked model and between a marked and a stolen surrogate model. The test accuracy is the accuracy of a model's predictions on an unseen, labeled dataset from the same distribution. First, we define an auxiliary function that computes the accuracy of a model M on a dataset $D \subseteq \mathcal{X} \times \mathcal{Y}$.

$$\textsc{acc}(\theta, D) = \Pr_{(x,y)\in D}[\arg\max_i(\textsc{Classify}(x, \theta)) = \arg\max_j(y)] \tag{4.4}$$

The embedding loss is the difference in test accuracy between an unmarked model θ_0 and a marked source model θ on a labeled test dataset $D_{\text{Test}} \subseteq \mathcal{X} \times \mathcal{Y}$.

$$L_{\text{Embed}}(\theta_0, \theta, D) = \textsc{Acc}(\theta_0, D) - \textsc{Acc}(\hat{\theta}, D) \tag{4.5}$$

The stealing loss is the difference in test accuracy between a marked source model θ and a stolen surrogate model $\hat{\theta}$.

$$L_{\text{Steal}}(\hat{\theta}, \theta, D) = \textsc{Acc}(\theta, D) - \textsc{Acc}(\hat{\theta}, D) \tag{4.6}$$

The defender wants to minimize the embedding loss, and the attacker wants to minimize the stealing loss.

Decision Threshold. The decision threshold $\kappa \in [0, 1]$ determines the lowest tolerated watermark accuracy to verify that a watermark is retained in a model. Ideally, a method defines a decision threshold as part of their adversary model that we could use to assess its robustness. Unfortunately, such methods are missing from the surveyed papers, meaning that we have to find a methodology to empirically derive decision thresholds for each watermarking method.

Determining the decision threshold for a watermarking method is difficult since it can depend on the unmarked model. For example, consider the case of the zero-bit, model-dependent watermarking method Frontier Stitching [127]. The presence of a watermark is detected if a surrogate model predicts the ground-truth labels for images that are part of the watermarking key. The watermarking key is composed of adversarial examples [213] generated for the source model. During the embedding, the source model is adversarially trained [151] to predict ground-truth labels for the watermarking key, whereas unmarked models still likely predict incorrect labels if the adversarial examples are transferable [225]. The problem is that the watermark accuracy of an unmarked model can increase without access to the source model by using adversarial training. This affects this watermarking method's decision threshold, which should be chosen large enough so that unmarked models are not incorrectly verified. The challenge lies in estimating the cumulative probability distribution that an unmarked model has a watermarking accuracy larger than some decision threshold. Such an estimation enables determining a decision threshold so that an incorrect verification (i.e., falsely claiming that a watermark is retained in a model) has a given probability.

Modeling the Decision Threshold. We empirically estimate an unmarked model's watermark accuracy given two random variables: the unmarked model and the watermarking key. Our goal is to estimate the cumulative probability that the watermark accuracy of a randomly generated watermarking key and a randomly sampled unmarked model is higher than some threshold. We make an i.i.d. assumption for our random variables and randomly generate 100 watermarking keys, each with a bit-length of $N = 100$. Then, we compute the watermark accuracy on a set of 30 unmarked models for CIFAR-10 and 20 unmarked models for ImageNet for every key and model pair. We model the cumulative normal probability distribution for the expected number of matched bits and choose a decision threshold. Table 4.3 summarizes the resulting decision thresholds for CIFAR-10 and ImageNet. We observe that some decision thresholds differ between CIFAR-10 and ImageNet, which requires the defender to derive a threshold specific to the model and dataset they want to protect. For the watermarking methods Content, Noise, Frontier Stitching, and Blackmarks, we observed that the choice of parameters affects their decision thresholds. In these cases, Table 4.3 shows the largest computed decision threshold, and we refer to Appendix B.4 for more information.

Rescaling Watermark Accuracies. Our goal is to compare the robustness of different watermarking methods. Relating watermark accuracies from different methods with each other is difficult because their decision threshold may differ. In such cases, the watermark accuracy alone does not indicate whether a method is robust without knowledge of the method's decision threshold. We avoid this issue by linearly rescaling the watermark accuracy by the method's decision threshold T so that a watermark is retained if the *rescaled* watermark accuracy is at least equal to some fixed value $T' = 0.5$ and removed otherwise. This allows us to plot the watermark accuracies for different methods into the same graph. We define a linear scaling function $S(x; T)$ that rescales the watermark accuracy so that (i) $S(T; T) = T'$ and (ii) $S(1; T) = 1$. The rescaling

	[255]	[255]	[255]	[3]	[109]	[128]	[34]	[33]	[192]	[227]	[216]
CIFAR-10	7%	49%	15%	15%	5%	53%	62%	40%	53%	58%	16%
ImageNet	1%	2%	1%	7%	16%	72%	81%	32%	58%	58%	6%

Table 4.3: This table shows the empirically determined, unscaled decision thresholds for each watermarking scheme on two datasets with a p-value of 0.05. We obtain these decision thresholds by generating 100 watermarking keys with a key length of $N = 100$ each and compute the mean watermark accuracy on a set of unmarked models. We use 30 unmarked models for CIFAR-10 and 20 for ImageNet. We refer to Appendix B.4 for details on the computation of the decision thresholds.

function uses the method's (unscaled) decision threshold T as a parameter and returns the scaled watermark accuracy.

$$S(x;T) = \max(0, \frac{1-T'}{1-T}x + \frac{T'-T}{1-T}) \tag{4.7}$$

We clip the output to avoid negative watermark accuracies. From this point forward, unless stated otherwise, we only refer to the rescaled watermark accuracy and decision threshold.

Runtime. The runtime helps assess the practicality of a watermarking method or evasion attack. We measure the runtime to (i) embed the watermark and (ii) run an evasion attack. Since runtimes depend on the hardware, we report all runtimes measured on (single) Tesla P100 GPUs.

Attack Success Criterion. A success criterion determines whether an evasion attack successfully removed a watermark. We consider the watermark accuracy and the stealing loss of the surrogate model. We say an evasion attack was successful when the surrogate model's watermark accuracy is lower than the method's decision threshold and the surrogate model is well-trained. In this thesis, we consider a maximum stealing loss of *five* percentage points for a surrogate model to be considered well-trained. We refer to Section 4.4.1 for a security game that formalizes our success criterion.

4.5.2 Nash Equilibrium

Our empirical analysis performs an ablation study over multiple sets of parameters for each watermarking method and evasion attack. We now describe a method to measure the robustness of a watermarking method against one or more evasion attacks under the consideration that the defender and attacker can choose from a set of parameters. For every watermarking method and evasion attack, we ablate over multiple parameters (see Section 4.2 and Appendix B.2) from which

the defender and attacker can choose. We define a zero-sum game between the defender and attacker, where both players want to choose optimal parameters to maximize their gains.

We construct a *payoff* matrix $V \in \mathbb{R}^{m \times n}$ for n watermarking method parameters $\{d_0, .., d_n\}$ and m evasion attack parameters $\{a_0, .., a_m\}$. The defender and attacker have full knowledge of this payoff matrix. An entry in this matrix is computed by applying a payoff function on the outcome of running an attack with the row's parameters against a watermarking method with the column's parameters. We define the following payoff function. The payoff is zero for non-successful attacks, and otherwise, the payoff is equal to the surrogate model's test accuracy. At the start of the game, both players choose their strategy from the payoff matrix. We observe that the defender maximizes their gain if they minimize the payoff, whereas the attacker wants to maximize the payoff. A *Nash equilibrium* is found when neither player gains from changing their chosen parameters. Optimal parameters for both players can be derived as follows.

$$(d^*, a^*) = (d_i, a_j) = \arg\min_i(\arg\max_j V[i, j]) \tag{4.8}$$

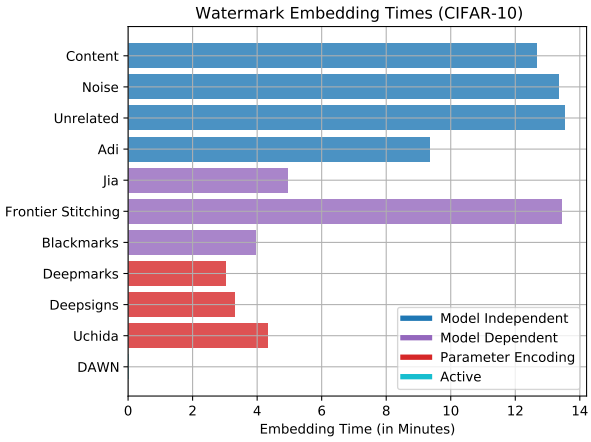
Using the Nash equilibrium to present our results, we demonstrate that successful watermark evasion attacks exist due to the watermarking method's vulnerability rather than a wrong choice of parameters.

4.6 Experiments

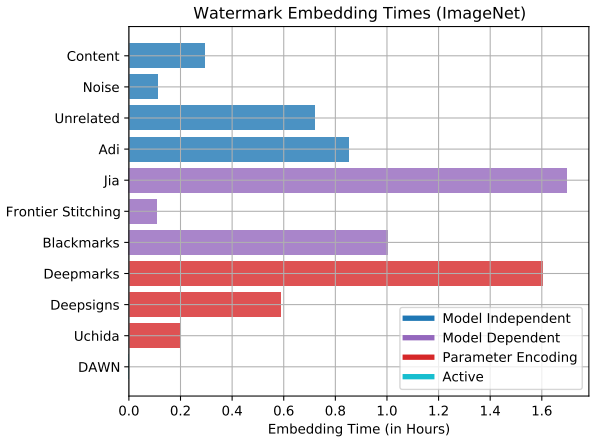
In this section, we present the results of our experiments. We describe our experimental setup, a methodology for splitting data between the attacker and defender, and the model architectures. Then, we report measured quantities of the attacks and methods, such as their runtimes or the embedding loss.

We analyze the robustness of each watermarking method against (i) all attacks, (ii) categories of attacks, and (iii) individual attacks. The first experiment validates whether a method is robust if the attacker knows which method the defender has chosen (but not its parameters). The second experiment analyzes which attack categories are most effective against each watermarking method. The third experiment focuses on finding *dominant* attacks, i.e., successful evasion attacks that remove any watermark. Our results show that none of the single attacks themselves removes all watermarks. Still, we can find *combined* dominant attacks. We cannot depict all evaluation results in this thesis. Hence, we will make our results publicly available via an interactive graph that shows the Nash equilibrium for a set of attacks against a set of watermarking methods[5].

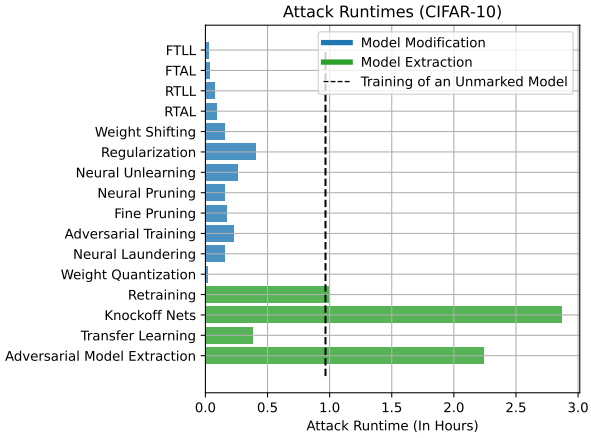
[5] https://crysp.uwaterloo.ca/research/mlsec/wrt



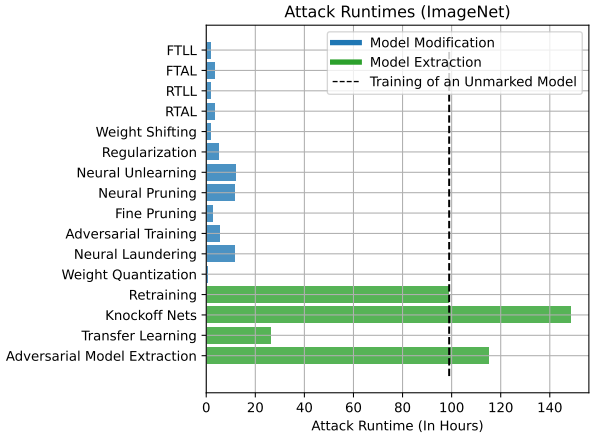
(a) CIFAR-10



(b) ImageNet

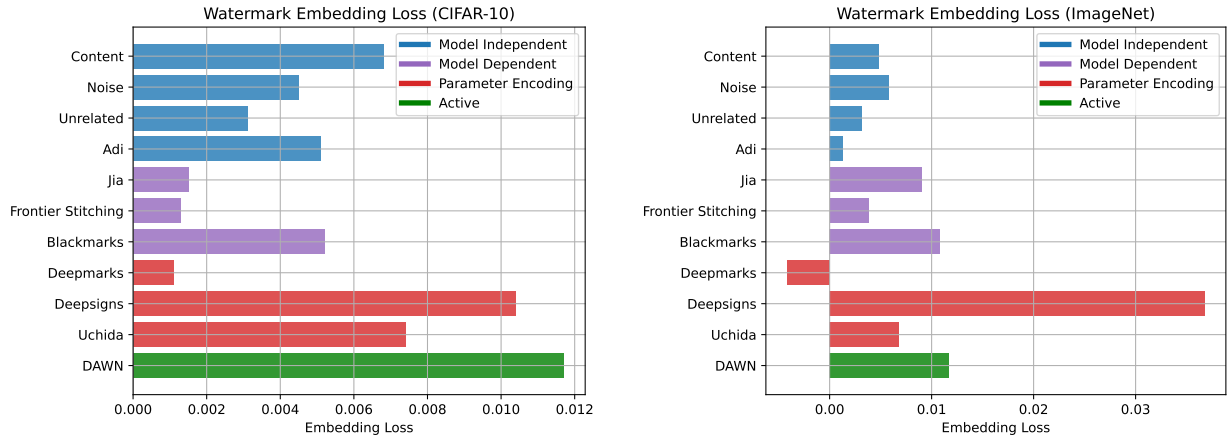


(c) CIFAR-10



(d) ImageNet

Figure 4.1: The runtimes for embedding (top) and attacking (bottom) a watermark on CIFAR-10 and ImageNet measured on Tesla P100 GPUs.



(a) CIFAR-10 (b) ImageNet

Figure 4.2: The embedding losses after watermarking compared to the unmarked model.

4.6.1 Setup

We implement all watermark methods and evasion attacks in our novel Watermark-Robustness-Toolbox (WRT) with PyTorch [173] running as its backend. WRT will be made available as open-source code, which allows independent verification of our empirical results. All reported runtimes in this chapter were obtained using Tesla P100 GPUs.

4.6.2 Datasets

We embed watermarks into source models trained on the image classification datasets CIFAR-10 [121] and ImageNet [47]. Our method of splitting the dataset between the attacker and defender differs depending on the attack's category. For model modification attacks, the attacker can access a third of the dataset, and the defender can access the remaining two-thirds. Model extraction attacks require more data to achieve a high test accuracy, and thus, we give the attacker and defender access to the entire training dataset. We refer to Appendix B.3 for a description of the datasets and details on our method of splitting the training dataset.

4.6.3 Model Architectures

All of our experiments assume that the attacker knows the source model's architecture. For CIFAR-10, we use the wide ResNet 28x10 [251], and for ImageNet, the ResNet-50 [82] architectures. We

also perform cross-architecture retraining using a DenseNet-121 [94] for CIFAR-10 and ImageNet.

4.6.4 Runtimes and Embedding Losses

We report the runtimes for the evasion attacks and watermark embeddings. Since the choice of parameters influences the runtimes, the results can only show general trends. We ensured choosing parameters and training configurations that an attacker or defender would likely choose in practice, such as using early stopping for the embedding. Section 4.2 and appendix B.2 contains detailed descriptions of the chosen parameters and implementation details. Figures 4.1a, 4.1c and 4.2a show results for CIFAR-10 and Figures 4.1b, 4.1d and 4.2b for ImageNet. All graphs are shown as bar charts with the watermarking method or evasion attack on the y-axis and the runtime or embedding loss on the x-axis. The coloring indicates categories.

Embedding Runtimes. Figures 4.1a and 4.1b show the embedding runtimes for CIFAR-10 and ImageNet. We refer to the *training time* as the time to train an unmarked model from scratch. This training time is a reference point to assess the practicality of evasion attacks and watermarking methods. We observe a training time of 1h and 100h for CIFAR-10 and ImageNet.

On CIFAR-10, model-independent methods have the highest embedding time of about 20% of the total training time. Parameter encoding methods are more efficient and require only about 9% of the training time. We do not consider the runtime for the active methods but point out that deploying them incurs computational costs for each inference. On ImageNet, we observe that methods such as Jia and Deepmarks require considerably more time than on CIFAR-10, whereas model-independent methods are relatively fast. Jia has the highest embedding time, exceeding 1.6% of the total training time. All embedding times are low compared to the total training times on both datasets, and we conclude that all surveyed methods are efficient.

Attack Runtimes. Figures 4.1c and 4.1d show the attack runtimes for CIFAR-10 and ImageNet. Input Preprocessing attacks are not shown because they run only during inference. We observe that the runtimes of all attacks are proportionally similar on CIFAR-10 and ImageNet. On both datasets, model extraction attacks require significantly longer than model modification attacks. Transfer learning is an exception for a model extraction attack that is relatively fast as it requires about 40% of the training time on CIFAR-10 and roughly 25% of the training time on ImageNet. Knockoff is the slowest attack, which takes considerably longer than retraining due to the larger size of the training dataset.

Embedding Losses. Figures 4.2a and 4.2b show the embedding losses, which is the drop in test accuracy due to embedding the watermark into the source model (see Section 4.5.1). Embedding losses for CIFAR-10 and ImageNet are about one percentage point, with the exception of Deepsigns on ImageNet, which has an embedding loss of more than three percentage points. The parameter encoding method Deepmarks incurs the lowest embedding loss on both datasets.

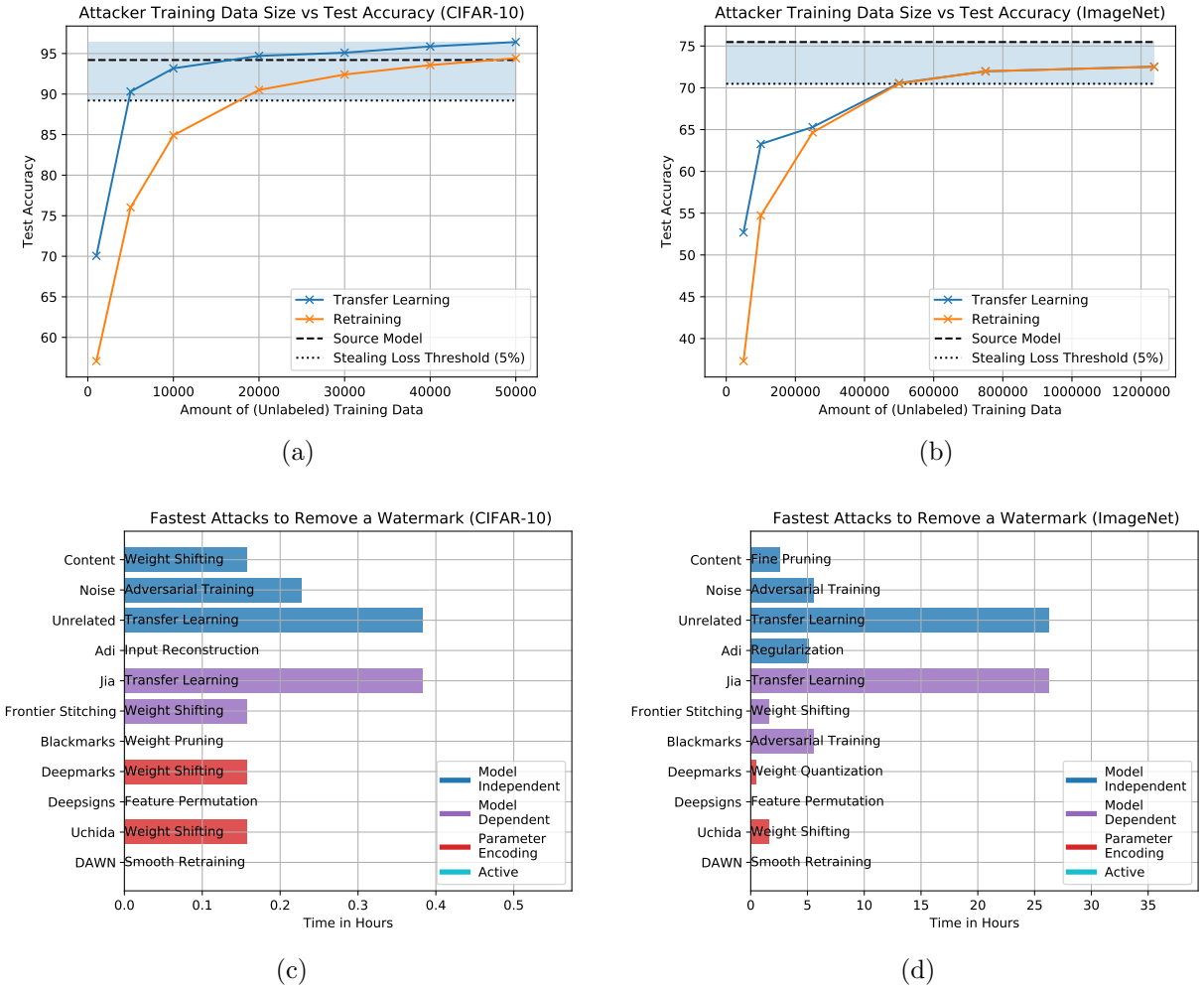
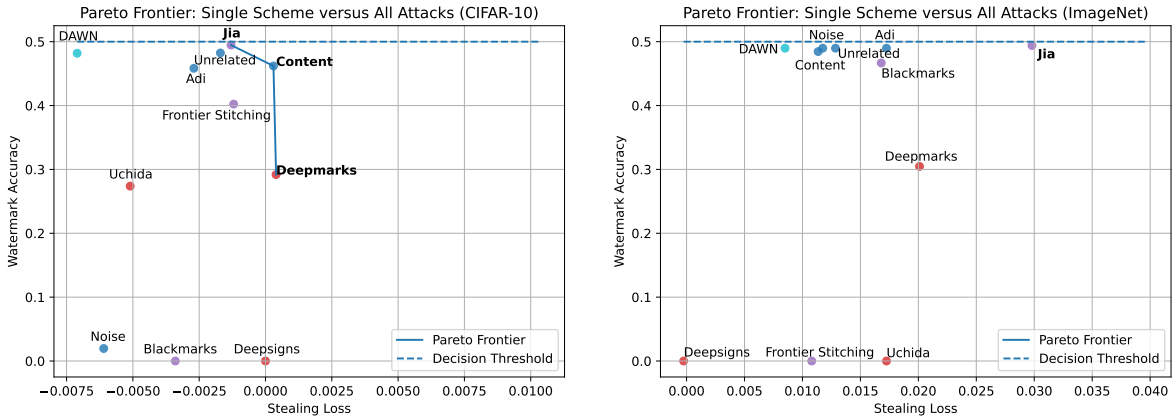


Figure 4.3: Figures 4.3a and 4.3b show the minimum amount of training data required for the transfer learning and retraining attacks to achieve a given test accuracy. Figures (c, f) show the fastest attack that removes each watermark. With DAWN [216], the attacker has to obtain white-box access by extracting the source model before using other attacks. For a fair comparison with other methods, we do not consider this extraction runtime.



(a) CIFAR-10 (b) ImageNet

Figure 4.4: The Pareto frontier for all watermarking methods with respect to the stealing loss (defined in Section 4.5.1) and watermark accuracy of the best attack. A watermark accuracy lower than $T' = 0.5$ means that the watermark is not robust.

4.6.5 Robustness of Watermarking Methods

This section describes our analysis of the robustness of each watermarking method against all attacks. The defender can choose from a set of parameters for a single watermarking method, whereas an attacker can choose from all parameters for all evasion attacks. The goal of this analysis is to evaluate whether any watermarking method can be considered robust against an adaptive adversary. We assume that the attacker knows the watermarking method chosen by the defender but not its parameters.

Robustness. The results are illustrated in Figures 4.4a and 4.4b in the form of a scatter plot. The x-axis shows the stealing loss, which is the drop in test accuracy in the surrogate model compared to the source model, and the y-axis shows the rescaled watermark accuracy (see Section 4.5.1). A watermark accuracy lower than $T' = 0.5$ means that the watermark has been removed. We highlight T' by a dashed line in the graph. We draw the *Pareto frontier*, which is the set of watermarking methods with a watermark accuracy or stealing loss so that no other watermarking method improves upon both metrics. Jia, Content, and Deepmarks are members of the Pareto frontier for CIFAR-10 and only Jia for ImageNet.

We observe that none of the watermarking methods is robust. For CIFAR-10, the marked source models can be stolen with a stealing loss of less than one percentage point, i.e., without a considerable loss of utility. For ImageNet, we observe that evasion attacks incur a higher stealing loss overall. Jia has the highest stealing loss of three percentage points, whereas the remaining

watermarking methods have a stealing loss of at most two percentage points. We designed a set of adaptive attacks against a subset of watermarking methods and feature their results separately as follows.

- **Smooth Retraining**: The smooth retraining attack is adapted to the active watermarking method DAWN. The idea is to query DAWN multiple times with the same image, using a different affine transformation (e.g., cropping, horizontal flipping) for each query. The label for each image is the mean across all collected labels for each image. Smooth retraining is the only attack that removes DAWN on CIFAR-10.
- **Feature Permutation**: Hidden layer neurons are permutation invariant, meaning that we can apply a random permutation on the features without losing any utility of the model. Deepsigns is the only method that is not robust against feature permutation attacks.
- **Weight Shifting**: Weight shifting perturbs the filter weights of each convolutional layer by the negative mean over all its filters, adds a small amount of noise and fine-tunes the model. We observe that weight shifting is the only model modification attack that removes Uchida on CIFAR-10 and ImageNet.

Fastest Attacks. Figures 4.3c and 4.3d show the fastest attacks that successfully remove a watermark. On CIFAR-10, we observe that some methods, such as Deepsigns, Blackmarks, and Adi, can be removed with a negligible runtime, whereas Jia and Unrelated require the highest runtime. On ImageNet, we observe that evading the watermarks from Unrelated and Jia requires the highest runtime, whereas parameter encoding methods can be removed in the shortest amount of time. For both datasets, we observe that the fastest attacks depend on the watermarking method, i.e., there is no single fastest attack or attack category against all watermarking methods.

Dataset Availability. We stated that the dataset available for a model extraction attack is larger than for model modification attacks. We ablate over the amount of data available to the attacker to achieve a given test accuracy. This is relevant to discuss the practicality of model extraction attacks because the attacker wants to minimize both (i) the training time and (ii) the amount of data required to perform an attack.

Figures 4.3a and 4.3b show the amount of unlabeled data in relation to the surrogate model's test accuracy for CIFAR-10 and ImageNet. The attacker trains their surrogate model on data labeled by source models with a test accuracy of 94.20% on CIFAR-10 and 75.48% on ImageNet. On CIFAR-10, we observe that transfer learning achieves a significantly higher test accuracy than retraining from scratch using the same amount of data. Retraining requires at least about 20k samples to perform a successful attack, whereas transfer learning needs only about 5k samples. On ImageNet, the difference between retraining and transfer learning goes to zero when more than 250k samples are available to the attacker. Performing a successful evasion attack requires at least 500k samples. While transfer learning still requires the same amount of data as retraining from scratch, we point out that transfer learning requires significantly less computation time.

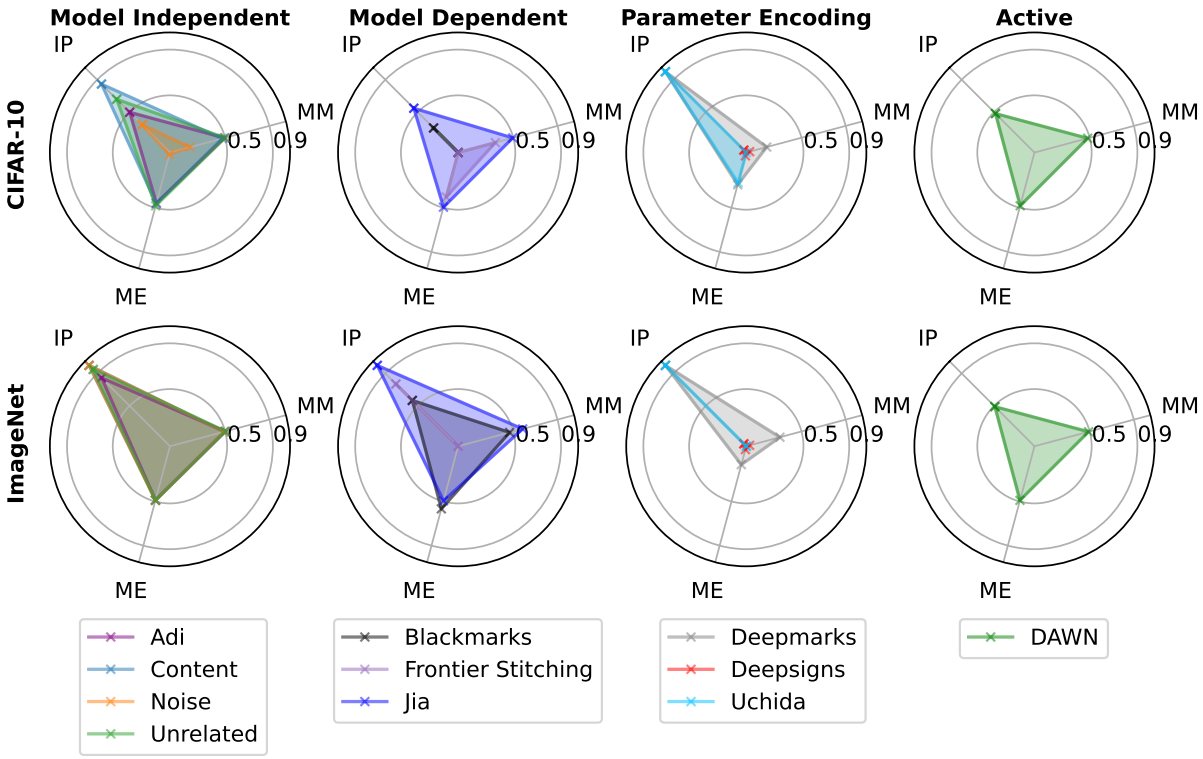


Figure 4.5: An illustration of the robustness of each surveyed watermarking method against categories of attacks for CIFAR-10 (top) and ImageNet (bottom). The axes show the (scaled) watermark accuracy of a method against the best attack from each category. A watermark is robust against a category if the watermark accuracy is at least $T' = 0.5$. The method and attack parameters are chosen using the Nash Equilibrium, and we ignore attacks when their stealing loss exceeds five percentage points. The attack categories are Input Preprocessing (IP), Model Modification (MM), and Model Extraction (ME).

4.6.6 Robustness against Attack Categories

In the previous section, we showed that none of the watermarking methods is robust against all attacks. We further analyze the robustness of each watermarking method against categories of evasion attacks. The defender can choose from the set of parameters for each watermarking method, and the attacker can choose from the set of parameters for attacks of only one category. This analysis provides insights into the vulnerability of watermarking methods to certain attack categories. We refer the reader to Table 4.2 for a list of all attacks and their categories.

Figure 4.5 shows a radar plot of our result for CIFAR-10 and ImageNet. The radar plot axis shows the watermark accuracy of each method against the most successful attack from each attack category. A larger covered area in the plot illustrates higher robustness to multiple attack categories. A method is robust against the attack category if the watermark accuracy is at least $T' = 0.5$ (see Section 4.5.1). We analyze the results for each category.

Input Preprocessing. We observe that input preprocessing attacks often do not remove a watermark on either CIFAR-10 or ImageNet, but these attacks often impact the watermark accuracy. Input smoothing and input reconstruction are effective against Adi and Noise on CIFAR-10 but not on ImageNet. We always apply feature permutation because it does not impact the model's utility and requires negligible computational costs. For this reason, Deepsigns, which is vulnerable to feature permutation, is removed by input preprocessing attacks for both CIFAR-10 and ImageNet. Similarly, DAWN is not robust because it requires extracting a surrogate model prior to running an input preprocessing or model modification attack. We extract a surrogate model for DAWN using smooth retraining, which already removes the watermark.

Model Modification. Model modification attacks successfully remove all watermarks for CIFAR-10 and ImageNet, except for Jia on ImageNet. Many surveyed watermarking methods are vulnerable to multiple model modification attacks, whereas other methods, such as Uchida, are only vulnerable to our adaptive weight-shifting attack. Similar to input preprocessing attacks, we observe that model modification attacks that do not remove the watermark can still significantly lower the watermark accuracy.

Model Extraction. We observe that almost none of the methods is robust to model extraction attacks on CIFAR-10 and ImageNet. The most effective attack is transfer learning for both CIFAR-10 and ImageNet because it requires a fraction of the training time for an unmarked model, and it removes almost all of the surveyed watermarks. Notable exceptions are Noise and Blackmarks, which are robust against transfer learning on ImageNet, but Noise is not robust against retraining on ImageNet and Blackmarks is not robust against adversarial training. Retraining, distillation, and adversarial training from scratch yield similar results as transfer learning, but they require (i) at least as much data and (ii) have a significantly longer runtime. Therefore we do not evaluate distillation and adversarial model extraction on ImageNet if a model is already vulnerable to transfer learning or retraining.

Attack \ Watermark	[255]	[255]	[255]	[3]	[109]	[127]	[34]	[33]	[192]	[227]	[216]
Input Preprocessing											
Input Smoothing [241] (Gaussian Kernel)	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✗/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓
Model Modification											
Regularization [202]	✗/✓	✓/✓	✓/✓	✗/✗	✗/✓	✗/✓	✗/✓	✓/✓	✓/✓	✓/✓	✓/✗
Neural Cleanse [232] (Unlearning)	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓
Feature Permutation (Ours)	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✗/✗	✓/✓	✓/✓
Weight Shifting (Ours)	✗/✓	✓/✓	✗/✓	✗/✓	✓/✓	✓/✗	✗/✓	✗/✗	✗/✗	✗/✗	✓/✓
Model Extraction											
Knockoff Nets [169]	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✓/✓	✗/✓	✗/✓	✗/✓	-
Retraining [224]	✗/✗	✗/✗	✗/✗	✗/✗	✓/✓	✗/✓	✗/✗	✗/✗	✗/✗	✗/✗	✓/✗
Smooth Retraining (Ours)	-	-	-	-	-	-	-	-	-	-	✗/✗
Cross-Architecture Retraining	✗/✗	✗/✗	✗/✗	✗/✗	✗/✓	✗/✓	✗/✓	✗/✗	✗/✗	✗/✗	✓/✗
Transfer Learning [223]	✗/✗	✗/✓	✗/✗	✗/✗	✗/✗	✗/✗	✗/✓	✗/✗	✗/✗	✗/✗	✓/✗

Table 4.4: A summary of the robustness for each watermarking method against selected attacks. A checkmark (✓) indicates that the method is robust, and a cross (✗) indicates a lack of robustness. A dash (-) indicates that the attack has not been performed against the watermarking method (e.g., it is an adaptive attack designed against a subset of methods). With two consecutive marks, we indicate the robustness of CIFAR-10 and ImageNet.

In summary, we conclude that model extraction attacks are the most effective evasion attacks against most watermarks. Jia and Blackmarks are robust against retraining, but Jia is not robust against transfer learning, and Blackmarks is not robust against adversarial training. Even when a method is robust to retraining with the same architecture, the attacker can obtain a well-trained surrogate model by switching to a different architecture. We believe that transfer learning is more effective at removing some watermarks because the model re-uses low-level features learned from another task. Hence, watermarks encoded into low-level features are less likely to be robust against transfer learning. None of the parameter encoding methods is robust to transfer learning, also because extraction of such a watermark is not defined for a different model architecture. For example, Uchida defines a secret watermarking key that expects a layer's weights to be in the same shape as the source model's layer used for the embedding. Input preprocessing attacks are often not successful at removing a watermark, but they can reduce the watermark's accuracy. Model modification attacks, especially our novel adaptive attacks, successfully remove the watermark of a subset of watermarking methods and require (i) significantly fewer data and (ii) computational resources than model extraction attacks.

4.6.7 Attack's Effectiveness.

Table 4.4 shows whether a method is robust against an attack on CIFAR-10 and ImageNet for a subset of attacks. We observe that (i) attacks designed against one category of watermarks are not necessarily effective against watermarks from this category, and (ii) no method is robust against all model extraction attacks. Neural Cleanse [5] and Regularization [202] were designed against model-independent watermarks, but they often only decrease the watermark accuracy instead of removing the watermark. Jia is robust against retraining but not against transfer learning, suggesting that it is encoded into the low-level features of the source model. Transfer learning does not re-learn these low-level features from scratch, which could explain why transfer learning is more effective than retraining at removing the Jia watermark.

4.6.8 Dominant Attacks

This section analyzes whether a *dominant* attack exists that removes all watermarks. The existence of a dominant attack would mean that an attacker does not require knowledge about the method used by the defender to remove their watermark. The attacker can choose from the set of parameters for a single attack, whereas the defender can choose from the set of parameters for all watermarking methods. We observe that transfer learning is dominant for CIFAR-10, but there exists no dominant attack for ImageNet.

Creating Dominant Attacks. We now evaluate whether it is possible to find *combined* attacks that are dominant for source models trained on ImageNet. A combined attack performs many attacks in sequence. Our empirical results show that transfer learning combined with label smoothing is a dominant attack that removes all eleven watermarks on CIFAR-10 and ImageNet. The threat of combined attacks to the robustness of watermarking methods has not yet been explored, and we show that combined attacks can pose a significant threat.

4.7 Discussion

In this section, we discuss the practicality of the evaluated evasion attacks and argue that they are real-world threats to DNN watermarking. We identify three requirements for the attacker: (1) computational resources, (2) dataset availability, and (3) pre-trained models for transfer learning. Then we present guidelines for designing future watermarking methods and discuss the implications of our work for future research.

Computational Resources. Related work often restricts the availability of computational resources to the attacker in their threat model [227, 3, 33] and claims robustness against attackers with limited computational resources. While it may be the adversary's objective to minimize

computational resources, there is no theoretical guarantee that the adversary's learning problem will be a hard instance and require infeasible resources in some security parameters. Quite to the contrary, for the classification problems considered in this chapter, the adversary's costs are very feasible. Using shared GPUs in the cloud, the monetary costs are proportional to the attack's runtime. All runtimes were obtained on (single) Tesla P100 GPUs, which incur a cost of 0.43$ per on-demand hour of GPU-time[6]. Training a ResNet-50 model from scratch on ImageNet, consisting of 1.28 million images, takes about 100 hours and costs 43$. Transfer learning a model takes only 23 hours and brings down the costs to about 10$. There are even more optimized implementations [39] than ours, which achieve lower costs through various optimizations, e.g., by training on multiple GPUs, utilizing TPUs, or choosing more efficient model architectures. We conclude that, in absolute terms, the price for computational resources is almost insignificant and is likely not a deterrent for the attacker.

Dataset Availability. Related work often does not restrict the dataset available to the attacker, except for limiting the amount of ground-truth labels. We find that the attacker's dataset significantly influences the effectiveness of the evasion attacks. Increasing the amount of (unlabeled) domain data is sufficient to perform successful evasion attacks, and predicted labels can substitute ground-truth labels.

Using a transfer dataset (labeled data from a different domain) to train a model from scratch, such as in the Knockoff attack [169], does not lead to successful evasion attacks. For CIFAR-10, almost all watermarks are retained, and for ImageNet, we could not train a surrogate model with high test accuracy. We observe that access to domain data is crucial to perform these attacks.

Availability of Pre-Trained Models. Related work has not used transfer learning to remove watermarks, but transfer learning is a known method for training models in the visual domain [223]. We show that transfer learning is highly effective at removing watermarks; it is computationally efficient, and it can leverage access to less data than other model extraction attacks. Related work has shown that access to larger transfer sets can reduce the amount of domain data required for transfer learning [118]. Specifically, the authors use models that have been pre-trained on up to 300 million images and show that they can transfer learn this model for ImageNet with a test accuracy of 87.5% using as few as ten examples per class. We argue that it should not be problematic for attackers to obtain access to a pre-trained model from a different domain in practice. There exist many platforms to share pre-trained models with various model architectures, such as ONNX[7] or Model Zoo[8], without charging the user.

[6] https://cloud.google.com/compute/gpus-pricing
[7] https://onnx.ai/
[8] https://modelzoo.co/

4.7.1 Guidelines

In this section, we propose guidelines for evaluating the robustness of watermarking methods. These guidelines incorporate many of our findings and provide a minimal checklist to claim robustness for a watermarking method.

Attacker's Dataset. Our experiments have shown that robustness on CIFAR-10 does not imply robustness on ImageNet and vice versa. In general, we observed that it is more difficult to remove watermarks from models trained on ImageNet than from models trained on CIFAR-10. We believe that is because (i) the model and task are more complex and (ii) attacks have a greater impact on the model's utility (measured by the test accuracy). Our recommendation for image classification models is to experiment on (i) a small dataset, (ii) a dataset with large input image dimensions, and (iii) a dataset with a large number of classes. We use ImageNet to cover the last two requirements within one dataset. Furthermore, we recommend listing the amount of data and ground-truth labels used during the attack for evasion attacks.

Decision Threshold. We noticed that a method to derive a watermarking method's decision threshold is missing from many papers in related work. Disproving the robustness claim of a method requires a method of deriving the decision threshold. This method affects the method's usability. For example, for the watermarking method Adi, we could theoretically derive the decision threshold because the input images and target labels are drawn randomly. However, Blackmarks requires an empirical method to derive a decision threshold because it relies on adversarial examples for which it is difficult to theoretically quantify the transferability of these examples to unmarked models. Our work proposes a general method to empirically determine this decision threshold, which involves training many unmarked models on CIFAR-10 and ImageNet (hence the usability is limited).

Parameter Ablation. We recommend stating all parameters for an evasion attack and watermarking method that can be included in an ablation study. In our work, we manually selected parameters for our ablation study. For multiple parameters, the robustness should be evaluated at the Nash equilibrium. This enhances (i) the reproducibility of robustness claims and (ii) allows for a fair evaluation of a method's robustness and an attack's effectiveness.

Class Accuracies. For some watermarking methods, such as Content or Jia, we observed that the source model might unlearn a single class during the embedding process. On ImageNet, the test accuracy drops only by about 0.1% when the model unlearns a single class, but we argue that in such cases, the impact of the watermark is greater than the drop in overall test accuracy suggests. We recommend evaluating the drop in test accuracy for single classes.

Runtime. We suggest that a watermarking method or evasion attack should show their runtimes for the embedding or evasion procedure in relation to retraining a model from scratch. While the runtime of all surveyed watermarking methods is small, we believe the runtime is still a distinguishing factor for the proposed method's practicality.

4.7.2 Implications for Future Research

We show with our systematic, empirical study that a well-defined attacker can break all surveyed watermarking methods. We argue that DNN watermarking robustness needs to be defined and evaluated more rigorously. Many previous works evaluate against a relatively weak attacker that does not adapt their attacks. In other cases, the attacker is limited by their computational resources or the non-availability of other pre-trained models. We present a well-defined attacker model, and our Watermark-Robustness-Toolbox[9] is publicly available. Authors of future watermarking methods can evaluate robustness against the attacker presented in this chapter.

4.8 Conclusion

We evaluate eleven watermarking methods from related work and empirically determine their decision thresholds for the CIFAR-10 and ImageNet datasets. Then, we measured the performance of a large set of evasion attacks against all watermarking methods and ablate over multiple parameters for each method and evasion attack. We use the Nash equilibrium to evaluate a method's robustness against (i) all attacks, (ii) categories of attacks, and (iii) single attacks. Our results show that none of the methods is robust against all attacks. We analyze these results by analyzing each attack category's effectiveness and find that the most effective evasion attack categories are model extraction attacks, followed by model modification attacks. We show that transfer learning removes all watermarks on CIFAR-10, but there exists no such dominant attack for ImageNet. We create a combined attack composed of (i) transfer learning and (ii) label smoothing that removes all eleven watermarks. Finally, we discuss the practicality of the evasion attacks, e.g., their monetary costs and the dataset availability of the attacker, and propose guidelines for evaluating the robustness of DNN watermarking. We hope that our work will improve future evaluations of black-box or white-box DNN watermarking methods.

[9] https://github.com/dnn-security/Watermark-Robustness-Toolbox

Chapter 5

Deep Neural Network Fingerprinting By Conferrable Adversarial Examples

This chapter is based on work [146] that was published at the 2021 International Conference on Learning Representations (ICLR). I investigate whether it is possible to extract an identifying code from a source model to detect its unauthorized redistribution against model extraction attacks. The threat is an attacker who tries to corrupt the identifying code and use a model against its fair and intended use. I propose a fingerprint composed of samples from a new subclass of transferable adversarial examples, which are called *conferrable* examples. Conferrable examples can be used as a source model's fingerprinting key to detect surrogate models while withstanding model extraction attacks.

5.1 Motivation

Deep neural network (DNN) classifiers have become indispensable tools for addressing many practically relevant problems, such as autonomous driving [220], natural language processing [245], and health care predictions [59]. While a DNN provides substantial utility, training a DNN is costly because of data preparation (collection, organization, and cleaning) and computational resources required to validate a model [179]. For this reason, DNNs are often provided by a single entity and consumed by many, such as in the context of Machine Learning as a Service (MLaaS). A threat to the provider is *model stealing*, in which an adversary derives a surrogate model from only API access to the source model.

Consider an MLaaS provider that wants to protect its service and hence restrict its redistribution, e.g., through a contractual usage agreement because trained models constitute their

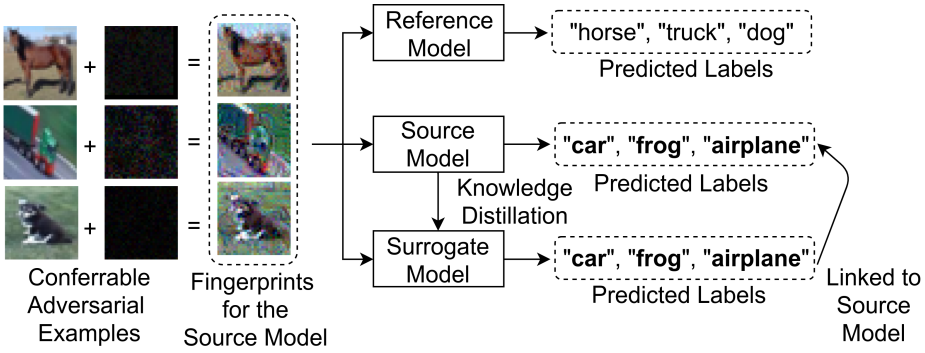


Figure 5.1: Conferrable adversarial examples used as a fingerprint to identify surrogates.

intellectual property. A threat to the model provider is an attacker who derives surrogate models and publicly deploys them. Since access to the source model has to be provided, users cannot be prevented from deriving surrogate models. Krishna et al. [120] have shown that model stealing is effective because even high-fidelity surrogates of large models can be stolen and efficient because surrogate models can be derived for a fraction of the costs with limited access to domain data.

This chapter proposes a DNN fingerprinting method to predict whether a model is a (stolen) surrogate or a (benign) reference model relative to a source model. DNN fingerprinting is a new area of research that extracts a persistent, identifying code (fingerprint) from an already trained model. Model stealing can be categorized into *model modification*, such as weight pruning [264], or *model extraction* that uses some form of knowledge distillation [86] to derive a surrogate from scratch. Claimed security properties of existing defenses ([3, 255]), have been broken by model extraction attacks [202]. Our fingerprint is the first defense specifically designed to withstand model extraction attacks, which extends to robustness against model modification attacks.

Our research provides new insight into the transferability of adversarial examples. In this chapter, we hypothesize that there exists a subclass of targeted, transferable, adversarial examples that transfer exclusively to surrogate models but not to reference models. We call this subclass *conferrable*. Any conferrable example found in the source model should have the same misclassification in a surrogate model but a different one in reference models. We propose metrics to measure conferrability and an ensemble adversarial attack that optimizes these new metrics. Our fingerprint is made of conferrable adversarial examples.

Retrained CIFAR-10 surrogate models can be verified with a perfect ROC AUC of 1.0 using our fingerprint, compared to an ROC AUC of 0.63 for related work [23]. While our fingerprint is robust to almost all derivation and extraction attacks, we show that some adapted attacks may remove our fingerprint. Specifically, our fingerprint lacks robustness against transfer learning attacks, where a similar, pre-trained model and limited domain data are accessible to the attacker.

Moreover, our fingerprint is not robust against adversarial training [151] from scratch. Adversarial training is an adapted model extraction attack specifically designed to limit the transferability of adversarial examples. We hypothesize that incorporating adversarial training into the generation process of conferrable adversarial examples may lead to higher robustness against this attack.

5.2 Related Work

In black-box adversarial attacks [171, 224, 151], access to the target model is limited, meaning that the target architecture is unknown and computing gradients directly is not possible. Transfer-based adversarial attacks [170, 171] exploit the ability of an adversarial example to *transfer* across models with similar decision boundaries. *Targeted* transferability additionally specifies the target class of the adversarial example. Our proposed adversarial attack is a targeted, transfer-based attack with white-box access to a source model (that should be defended) but black-box access to the stolen model derived by the attacker.

Liu et al. [141] and Tramèr et al. [226] show that (targeted) transferability can be boosted by optimizing over an ensemble of models. Our attack also optimizes over an ensemble of models to maximize transferability to stolen surrogate models while minimizing transferability to independently trained models, called reference models. We refer to this special subclass of targeted transferability as *conferrable*. Tramèr et al. [226] empirically studies transferability and finds that transferable adversarial examples intersect with high-dimensional "adversarial subspaces" across models. We further their studies and show that (i) stolen models apprehend adversarial vulnerabilities from the source model and (ii) parts of these subspaces, in which conferrable examples are located, can be used in practice to predict whether a model has been stolen.

Watermarking of DNNs is a related method to DNN fingerprinting, where an identifying code is embedded into a DNN, thereby potentially impacting the model's utility. Uchida et al. [227] embed a secret message into the source model's weight parameters but requires white-box access to the model's parameters for the watermark verification. Adi et al. [3] and Zhang et al. [255] propose backdooring the source model on a set of unrelated or slightly modified images. Their approaches allow black-box verification that only requires API access to the watermarked model. Frontier-Stitching [128] and BlackMarks [54] use (targeted) adversarial examples as watermarks. These watermarks have been evaluated only against model modification attacks but not against model extraction attacks that train a surrogate model from scratch. At least two of these watermarking schemes [3, 255] are not robust to model extraction attacks [202]. Cao et al. [23] recently proposed a fingerprinting method with adversarial examples close to the source model's decision boundary. We show that their fingerprint does not withstand retraining as a model extraction attack and propose a fingerprint with improved robustness to model extraction attacks.

5.3 Threat Model

We consider an attacker who wants to derive a surrogate model from access to the defender's source model that (i) has a similar test accuracy and (ii) cannot be verified by the defender as one of the source model's surrogate models. We consider two attacker models: (i) an attacker with *white-box* access to the source model, meaning access to the source model's parameters, and (ii) a less informed, *black-box* attacker with only API access to the source model. A more informed attacker can drop information and invoke all attacks of a less informed attacker. Robustness against a more informed attacker implies robustness against a less informed attacker. Our attacker has access to many unlabeled images but is limited in their access to ground-truth labels; otherwise, they could train their own model, and there would be no need to steal a model.

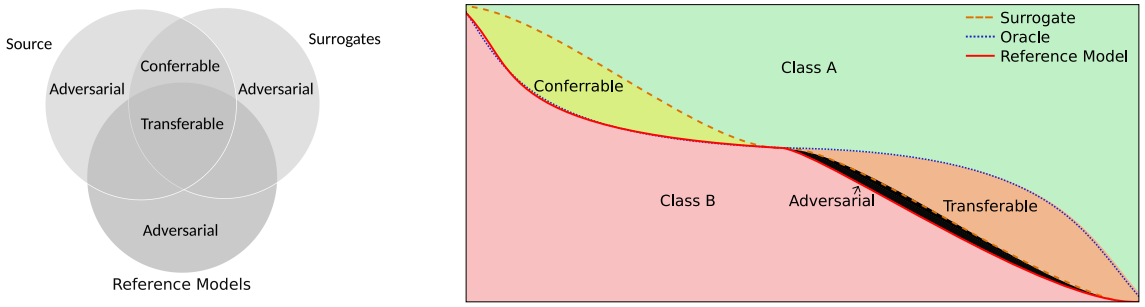
In our evaluation, we experiment with attackers on CIFAR-10 [121] with up to 80% of ground-truth labels. A plausible rationale for this constraint on label accessibility could be attributed to the absence of a reliable oracle, such as in specialized domains like healthcare, where the accuracy and confidentiality of data are important. Additionally, the procurement of labels could be cost-prohibitive, as it may necessitate the use of human annotation services, for instance, Amazon Mechanical Turk[1], which can incur significant monetary expenses.

5.4 Conferrable Adversarial Examples

Conferrability is a new property for adversarial examples, in which targeted transferability occurs only from a source model to its surrogates but not to independently trained reference models. Intuitively, surrogate models are expected to be more similar to the source model than any reference model, but quantifying this similarity is non-trivial. Conferrable examples are an attempt to quantify this similarity by shared adversarial vulnerabilities of the source model and its surrogates. Surrogate models differ from reference models in the objective function that they optimize. Reference models maximize fidelity to the ground truth labels, whereas surrogate models maximize fidelity to the source model's labels, which do not always coincide.

Figure 5.2a shows the relation between targeted adversarial, transferable, and conferrable examples for the source model, its surrogates, and all reference models. An example is transferable if it is adversarial to any model. It is conferrable when it is adversarial only to surrogate and source models. In that sense, conferrable adversarial examples are a subclass of targeted transferable adversarial examples. Figure 5.2b shows the relation between transferable and conferrable adversarial examples in a model's decision space at the example of binary classification. Transferable adversarial examples occur in those adversarial subspaces where the decision boundary

[1] https://www.mturk.com/



(a) (b)

Figure 5.2: (a) The relationship between transferable and conferrable adversarial examples. (b) A conceptual illustration of transferable and conferrable examples in a model's decision space relative to the ground truth provided by human annotators.

of surrogate and reference models coincide [225]. Conferrable adversarial examples occur in those adversarial subspaces where the decision boundaries of surrogate and reference models differ.

Targeted transferability for a class t and a set of models Θ can be computed as follows.

$$\text{Transfer}(\Theta, x; t) = \underset{\theta \in \Theta}{Pr}[\textsc{Classify}(x, \theta) = t] \tag{5.1}$$

Objective. The defender aims to find adversarial examples that maximize the difference in predicted labels between surrogate models $\mathcal{S} \subseteq \Theta$ and reference models $\mathcal{R} \subseteq \Theta$. This difference can be quantified by our *conferrability* score, which measures an example's transferability.

$$\text{Confer}(\mathcal{S}, \mathcal{R}, x; t) = \text{Transfer}(\mathcal{S}, x; t)(1 - \text{Transfer}(\mathcal{R}, x; t)) \tag{5.2}$$

The central challenge for optimizing conferrability is that we require access to the function *Transfer* that estimates an example's transferability score. To the best of our knowledge, the only known method to evaluate transferability is to train a representative set of DNNs and estimate the example's transferability itself. We use this method to evaluate transferability. In the case of conferrable examples, we evaluate transferability on a set of surrogate and reference models trained locally by the defender and use Equation 5.2 to obtain the conferrability score. We hypothesize that conferrability generalizes, i.e., examples that are conferrable to a representative set of surrogate and reference DNNs are also conferrable to other, unseen DNNs.

Conferrable Ensemble Method. This Section describes our adversarial attack to generate conferrable adversarial examples, called the *Conferrable Ensemble Method* (CEM). CEM operates on the source model θ, a set of its surrogates $\mathcal{S}_\theta$, and a set of reference models $\mathcal{R}$. Our attack constructs an ensemble model M_θ with a shared input layer that outputs per-class conferrability

scores for each $y \in \mathcal{Y}$. CEM generates highly conferrable adversarial examples by finding a perturbation δ so that $x' = x_0 + \delta$ maximizes the output of the ensemble model for a target class t.

We now present the construction of the ensemble model M_θ. The ensemble model produces two intermediate outputs for an input $x \in \mathcal{X}$, representing the mean logits of all surrogate and all reference models on x.

$$\text{Surr}(\mathcal{S}_\theta, x) = \frac{1}{|\mathcal{S}_\theta|} \sum_{\theta \in \mathcal{S}_\theta} \text{Dropout}(\textsc{Classify}(x, \theta); d) \tag{5.3}$$

$$\text{Ref}(\mathcal{R}, x) = \frac{1}{|\mathcal{R}|} \sum_{\theta \in \mathcal{R}} \text{Dropout}(\textsc{Classify}(x, \theta); d) \tag{5.4}$$

To enhance transferability, we optimize with Dropout and a drop ratio of $d = 0.3$ [211]. The ensemble model outputs a per-class conferrability score given an image. We refer to Appendix A.4 for details about the optimization.

$$M_\theta(x; \mathcal{S}_M, \mathcal{R}) = \texttt{softmax}(\text{Surr}(\mathcal{S}_M, x)(\mathbf{1} - \text{Ref}(\mathcal{R}, x))) \tag{5.5}$$

The loss in Equation (5.6) consists of three summands and is computed over the benign, initial example x_0, and the example at an intermediate iteration step $x' = x_0 + \delta$. We denote the cross-entropy loss by H. The first summand maximizes the output of the ensemble model for some target class t. The second summand maximizes the categorical cross-entropy between the current and initial prediction for the source model. The third summand minimizes the categorical cross-entropy between the source model's prediction and the predictions of its surrogates, and the total loss $\mathcal{L}$ is the weighted sum over all individual losses. We use shorthand notation to refer to the source model's classification of an image by $M(x) = \textsc{Classify}(x, \theta)$.

$$\mathcal{L}(x_0, x') = \alpha\ H(1, \max_t [M_\theta(x')_t]) - \beta\ H(M(x_0), M(x')) + \gamma\ H(M(x'), \text{Surr}(\mathcal{S}_\theta, x')) \tag{5.6}$$

In all our experiments, we use weights $\alpha = \beta = \gamma = 1$ and refer to Appendix A.6 for an empirical sensitivity analysis of the hyperparameters. We address the box constraint, i.e., $||\delta|| \leq \epsilon$, similarly to PGD [151] by clipping intermediate outputs around the ϵ ball of the original input x_0 in the L_∞-norm. For optimizing an input with respect to the loss, we use the Adam optimizer [116]. Note that we use an untargeted attack to generate targeted adversarial examples. Targeted attacks are harder to optimize than their untargeted counterparts [238]. We assign the source model's predicted label for the generated adversarial example as the target label and ensure that target classes are balanced in the fingerprint verification key.

Fingerprinting Algorithms. We now describe our fingerprinting generation and verification algorithms. For the generation, the defender locally trains a set of c_1 surrogate and c_2 reference models ($c_1 = c_2 = 18$) on their training data before executing CEM. Surrogate models are trained on data labeled by the source model, whereas reference models are trained on ground-truth

labels. The defender composes the ensemble model M_E as described in the previous paragraph and optimizes for a perturbation δ given an input x_0, so that $\mathcal{L}(x_0, x_0 + \delta)$ is minimized. The optimization returns a set of adversarial examples that are filtered by their conferrability scores on the locally trained models. If their conferrability score (see Equation 5.2) exceeds a minimum threshold (≥ 0.95), they are appended to the fingerprinting key. For fingerprint verification, we compute the error rate between the source model's prediction of the fingerprint and the target model's predictions. If the error rate exceeds $1 - \rho$, which we refer to as the *decision threshold*, the target model is predicted to be a reference model and a surrogate model otherwise.

5.4.1 Experiments

Setup. We evaluate the robustness and undetectability of our fingerprint. Then, we show that using our proposed method of deriving fingerprints (CEM), we can create adversarial examples with substantially higher conferrability scores compared to known adversarial attacks such as FGM [77], BIM [123], PGD [151], and CW-L_∞ [25]. We demonstrate the undetectability of our fingerprint by evaluating it against the detection algorithm proposed by Hitaj et al. [88]. Our study on robustness is the most extensive study conducted on DNN fingerprints or DNN watermarks compared to related work [23, 227, 255, 3, 128, 54, 216]. We compare our DNN fingerprint to another proposed DNN fingerprint called IPGuard [23]. Appendix A.1 and Appendix A.2 contain more details on the CIFAR-10 and ImageNet32 experiments.

Metrics. We measure the fingerprint *retention* as the success rate of conferrable examples in a target model $\hat{M}$. An example is successful if the target model predicts the target label given in $\mathcal{F}_y$. We measure the accuracy with which the predicted label matches the expected label from the fingerprinting key and refer to it as Conferrable Adversarial Example (CAE) accuracy.

CIFAR-10 models. We ablate over popular CNN architectures on CIFAR-10 without any modification to the standard training process[2]. The defender's source model has a ResNet20 [83] architecture. All surrogate and reference models required by the defender for CEM are ResNet20 models trained on CIFAR-10 using retraining. The attacker has access to various model architectures, such as DenseNet [99], VGG-16/VGG-19 [208] and ResNet20 [83], to show that conferrability is maintained across model architectures.

Removal Attacks. Removal attacks are successful if (i) the surrogate model has a high test accuracy (at least 85.55% for CIFAR-10, see Figure 5.4e) and (ii) the stolen surrogate's CAEAcc is lower than the verification threshold ρ. Stolen models are derived with a wide range of fingerprint removal attacks, which we categorize into (i) model modification, (ii) model extraction, and (iii) adapted model extraction attacks. Model modification attacks modify the source model directly, such as fine-tuning or parameter pruning. Model extraction distills knowledge from a source

[2] https://keras.io/examples/cifar10_resnet

ϵ	0.01	0.025	0.05	0.075	0.1	0.125	0.15
T_ϵ	63.00%	75.00%	84.00%	86.00%	87.00%	87.00%	87.00%

Table 5.1: Experimentally derived values for the decision threshold T_ϵ for CIFAR-10.

model into a fresh surrogate model, such as retraining or model extraction attacks from related work [171, 104, 169]. A defense against model extraction attacks is restricting the source model's output to the top-1 predicted label. We refer to an extraction attack that retrains on the top-1 predicted label by the source model as *Black-Box* attack. We also evaluate transfer learning as a model extraction attack, where a pre-trained model from a different domain (ImageNet32 [38]) is transfer learned onto the source model's domain using the source model's labels.

Adapted model extraction attacks limit the transferability of adversarial examples in the surrogate model, such as adversarial training [151]. In our experiment, we use multi-step adversarial training with PGD. We design another adapted attack against our fingerprint called the *Ground-Truth* attack, where the attacker has a fraction of $p \in [0.6, 0.7, 0.8]$ ground-truth labels. We trained surrogate models with Differential Privacy (DP), using DP-SGD [2], but even for large ϵ, the surrogate models had unacceptably low CIFAR-10 test accuracy of about 76%. The test accuracies of all surrogate models can be seen in Table 5.2. We repeat all removal attacks three times and report the mean values and the standard deviation as error bars. Our empirically chosen CIFAR-10 decision thresholds T_ϵ (see Table 5.1) are chosen relative to the perturbation threshold ϵ with which our adversarial attacks are instantiated.

Evasiveness Attack. Hitaj et al. [88] present an evasion attack against black-box watermark verification. Their evasion attack trains a binary DNN classifier to distinguish between benign and out-of-distribution images and reject queries by the defender. Our attacker trains a binary classifier to classify benign images and adversarial examples generated by FGM, CW-L_∞, and PGD adversarial attacks. The evasion attack is evaluated by the area under the curve (AUC) of the receiver operating characteristic (ROC) curve for varying perturbation thresholds.

Attacker Datasets. We distinguish between attackers with access to different datasets, which are CIFAR-10 [121], CINIC [44] and ImageNet32 [38]. CINIC is an extension of CIFAR-10 with downsampled images from ImageNet [47] for classes that are also defined in CIFAR-10. ImageNet32 is a downsampled version of images from all classes defined in ImageNet, i.e., ImageNet32 is most dissimilar to CIFAR-10. We observe that the similarity of the attacker's to the defender's dataset positively correlates with the surrogate model's test accuracy and thus boosts the effectiveness of the model stealing attack.

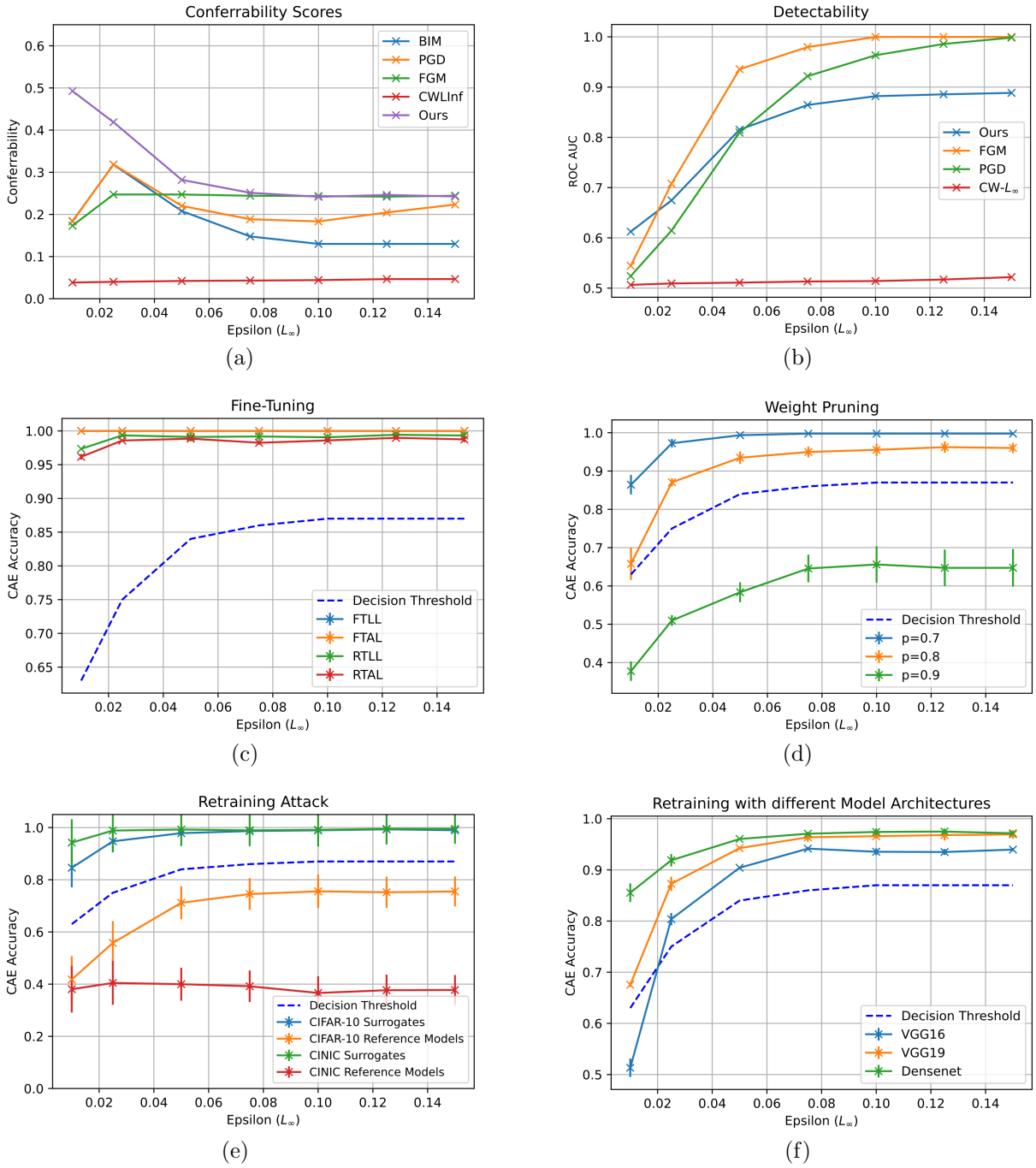


Figure 5.3: The (a) conferrability scores, (b) non-evasiveness, and (c-d) irremovability to model modification attacks. $\epsilon = 0.01$ corresponds to $^{2.5}/_{255}$ and $\epsilon = 0.15$ equals $^{38.25}/_{255}$

5.4.2 Results

Conferrability Scores. Our empirical results show support in favor of the existence of conferrable adversarial examples. CEM produces adversarial examples with the highest conferrability scores compared to classical adversarial attacks such as FGM, PGD, and CW-L_∞. We generate $\lambda = 100$ adversarial examples from the same set of CIFAR-10 test samples, filter those that are non-successful to the source model, and evaluate conferrability scores as specified by Equation (5.2) across five unseen CIFAR-10 surrogate and reference models trained from scratch. The results in Figure 5.3a show that CEM outputs adversarial examples with significantly higher conferrability scores than other attacks for small perturbation thresholds ϵ, which worsen as ϵ increases. For $\epsilon = {}^{2.5}/_{255}$, we measure a mean conferrability score of 0.49, which amounts to a mean CAEAcc of 0.42 in reference models and 0.85 in surrogate models (see Figure 5.3d). For $\epsilon = {}^{38.25}/_{255}$, we measure a conferrability score of only 0.24, which translates to a mean CAEAcc of 0.98 in surrogate models and 0.75 in reference models. These results show that transferability can be increased with ϵ at the expense of lower conferrability. We observe that the conferrability scores measured on surrogate and reference models used in CEM are nearly perfect. This indicates that CEM could benefit from access to an even larger ensemble of models.

Undetectability. We evaluate the undetectability against attacks proposed by Hitaj et al. [88]. Figure 5.3b show that (i) detectability increases with larger perturbation thresholds ϵ and (ii) small perturbation thresholds $\epsilon \leq {}^{6}/_{255}$ yield a ROC AUC of only 0.67. For $\epsilon \leq {}^{6}/_{255}$, the detection is too unreliable to be deployed in practice considering that only a small fraction of requests would contain the fingerprint. Sufficiently high recall against our fingerprint comes at the cost of more false positives, which diminishes the utility of the attacker's model to other users.

Model Modification Attacks. Figures 5.3c and 5.3d illustrate the robustness of our fingerprint against model modification attacks. We evaluate four types of fine-tuning attacks tried by Uchida et al. [227], which are implemented as follows.

1. Fine-Tune Last Layer (**FTLL**): Freezes all layers except for the penultimate layer.
2. Fine-Tune All Layers (**FTAL**): Fine-tune of all layers.
3. Retrain Last Layer (**RTLL**): Re-initializes the penultimate layer's weights and updates only the penultimate layer while all other layers are frozen.
4. Retrain All Layers (**RTAL**): Re-initializes the penultimate layer's weights, but all layers are updated during fine-tuning.

Our results with iterative weight pruning [264] show that larger pruning rates $p \in [0.7, 0.8, 0.9]$ can remove our fingerprint, but only when it degrades the model's test accuracy by 7% (see Table 5.2). Note that we show robustness to much higher pruning rates than related work [23]. The attack for $p = 0.9$ is unsuccessful because the surrogate test accuracy is lower than 85.55%, as described in Section 5.4.1. Our fingerprint is robust for all remaining pruning configurations.

Model Extraction Attacks. Figures 5.3 and 5.4 show that our fingerprint is also robust to model extraction attacks, except for transfer learning when the attacker can access in-domain data. Surprisingly, we find that surrogates extracted using out-of-domain data such as CINIC [44] have higher mean CAEAcc values than CIFAR-10 surrogates at a lower test accuracy (-2.16%). Similarly, we measure significantly lower CAEAcc values for reference models trained on CINIC than those trained on CIFAR-10. This means that it is increasingly difficult for the attacker to remove our fingerprints the more dissimilar the attacker's and defender's datasets become.

Figure 5.3f shows that our fingerprint is robust even when the attacker extracts the source model using a different surrogate model architecture. Figure 5.4b shows that transfer learning removes our fingerprint, but the attacker requires access to an in-domain dataset (e.g., CIFAR-10). Transfer learning does not evade detection when the attacker only has access to out-of-domain data such as CINIC. We observed that the pre-trained ImageNet32 model exceeds 80% test accuracy after only a single epoch with CIFAR-10, which may be too few updates for our fingerprint to transfer to the surrogate. The robustness of our fingerprint to a wide range of model extraction attacks shows that certain adversarial vulnerabilities (that enable conferrable examples) are consistently carried over from the source model to its surrogates. Our data supports that the class of transferable examples can be broken down further into conferrable examples and that known findings for transferable examples extend to conferrable examples.

Adapted Model Extraction Attacks. We now show the limitations of our fingerprint when the attacker uses *adaptive* attacks that limit the transferability of adversarial examples between the source and surrogate models. Figure 5.4c shows our fingerprint is not robust to adversarial re-training from scratch. In CIFAR-10 surrogates, we measure a mean CAEAcc of only 15% for $\epsilon = 0.025$, which supports the claims by Madry et al. [151] that adversarial training increases robustness to transfer attacks. We believe that incorporating adversarial training into the generation process of conferrable adversarial examples could enhance robustness. The results from our Ground-Truth attack in Figure 5.4d show that access to more ground-truth labels decreases the CAEAcc in surrogate models. The results show that our fingerprint is robust unless the attacker has more than 50% ground-truth labels for CIFAR-10.

Confidence Analysis. In Figure 5.4f, we show the ROC curve (false positive vs true positive rate) of our fingerprint verification on ten unseen, retrained CIFAR-10 surrogate and reference models. We compare our work with IPGuard [23] and generate $n = 100$ fingerprints in both cases. Our ROC AUC is 1.0, whereas IPGuard has an ROC AUC of only 0.63. These results show that IPGuard is not robust to retraining as a model extraction attack. Inspecting adversarial examples generated by IPGuard further, we measure a mean adversarial success rate of 17.61% for a surrogate and 16.29% for reference models. Figure 5.4e visualizes the difference in CAEAcc for a well-trained surrogate and reference models at $\epsilon = 6/255$. The plot shows that CAEAcc positively correlates with the surrogate's test accuracy. We measure a mean difference in CAEAcc of about 30% between a well-trained surrogate and reference models with our fingerprint.

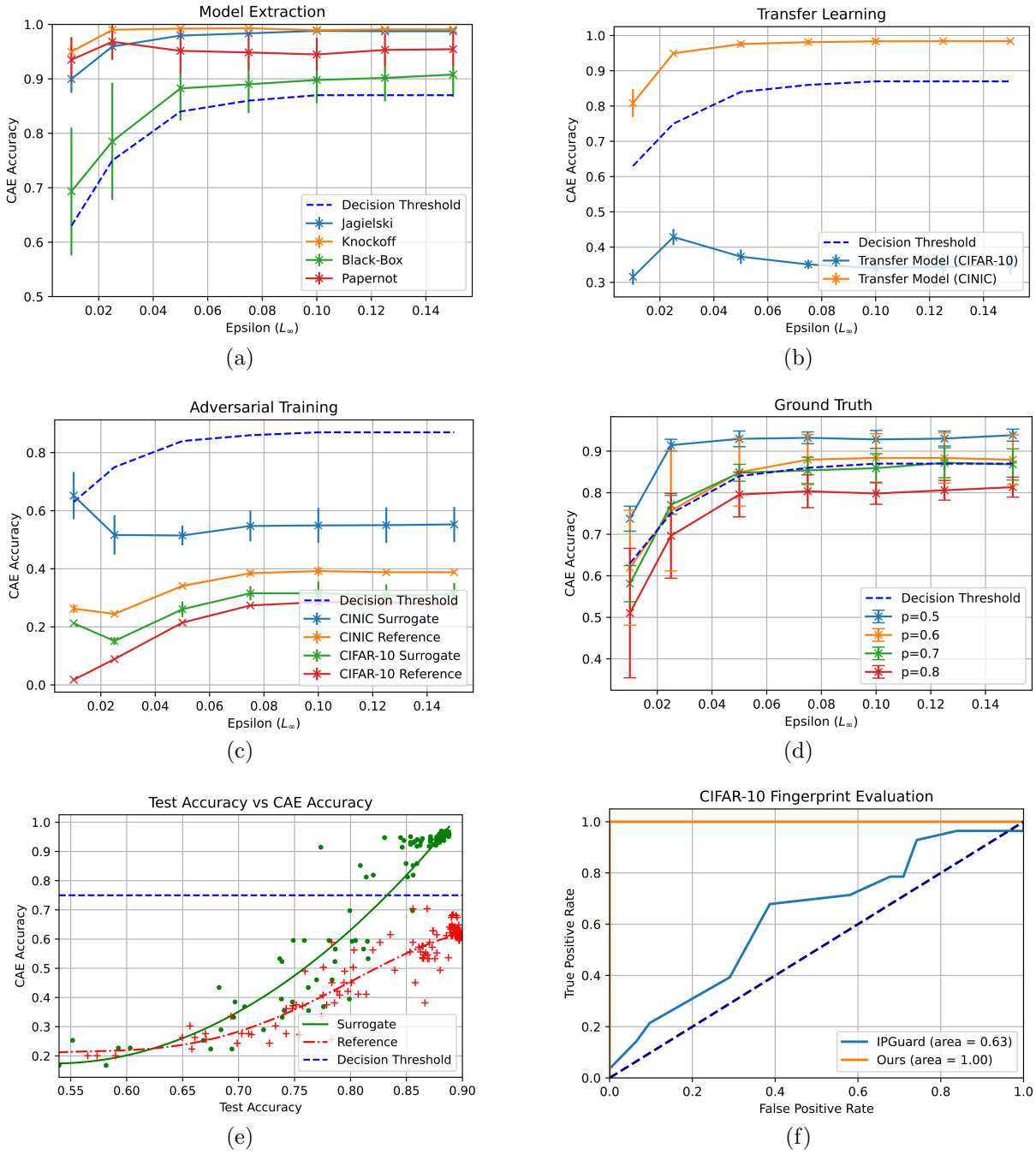


Figure 5.4: The robustness of our fingerprint against many different attacks.

	Fine-Tuning				Pruning		
Source	FTLL	FTAL	RTLL	RTAL	p=0.7	p=0.8	p=0.9
89.30	89.59	89.57	87.90	87.25	88.00	86.97	83.36
	Retrain				**Extraction**		
ResNet20	Densenet	VGG16	VGG19	CINIC	Jagielski	Papernot	Knockoff
89.22	90.88	90.91	90.05	87.06	88.74	87.34	84.55
AdvTrain		**Transfer Learning**		**Ground-Truth**			
CIFAR-10	CINIC	CIFAR-10	CINIC	p=0.5	p=0.6	p=0.7	p=0.8
89.75	88.15	89.59	88.15	89.54	89.13	89.43	89.33

Table 5.2: Mean CIFAR-10 test accuracies for surrogate models derived by threefold repetition of each evasion attack.

5.5 Conclusion

We empirically show the existence of conferrable adversarial examples. Our ensemble adversarial attack CEM outperforms existing adversarial attacks such as FGM [77], PGD [151] and CW-L_∞ [25] in producing highly conferrable adversarial examples. We formally define fingerprinting for DNN classifiers and use the generated conferrable adversarial examples as our fingerprint. Our experiments on the robustness of our fingerprint show increased robustness to model modification attacks and model extraction attacks. Transfer learning is a successful removal attack when the attacker has access to CIFAR-10 data but not when the attacker only has access to CINIC. Adversarial training from scratch is the most effective removal attack and successfully removes our fingerprint. We hypothesize that adding adversarial training into the generation process of conferrable adversarial examples may increase robustness against adversarial training. Our experiments confirm the non-evasiveness of our fingerprint against a detection method proposed by Hitaj et al. [88]. We empirically find that our fingerprint is the first to perfectly verify retrained CIFAR-10 surrogates with an ROC AUC of 1.0.

Chapter 6

PTW: Pivotal Tuning Watermarking for Pre-Trained Image Generators

This chapter is based on work that appeared at the 2023 USENIX Security Symposium [145]. We study whether watermarking methods can reliably control model misuse of deep image generators. The previous chapters focused on black-box and white-box watermarking methods to detect stolen models and deter their unauthorized redistribution. To control misuse, the provider can access only the generated content for attacker-chosen inputs. We refer to this as the *no-box* setting and propose learnable watermarks for pre-trained image generators that can be trained and embedded efficiently. Our results highlight that our watermarking methods for pre-trained image generators outperform existing watermarking methods in both effectiveness and robustness.

6.1 Motivation

Deepfakes, a term that describes synthetic media generated using deep image generators, have received widespread attention in recent years. While deepfakes offer many beneficial use cases, for example, in scientific research [42, 210] or education [180, 207, 58], they have also raised ethical concerns because of their potential to be *misused* which can lead to an erosion of trust in digital media. Deepfakes have been scrutinized for their use in disinformation campaigns [4, 85], impersonation attacks [158, 53] or when used to create non-consensual media of an individual violating their privacy [81, 46]. These threats highlight the need to control the misuse of deepfakes.

While some deepfakes can be recreated using traditional computer graphics, deep learning such as the Generative Adversarial Network (GAN) [75] can reduce the time and effort needed to create deepfakes. However, training GANs requires a significant investment in terms of computational

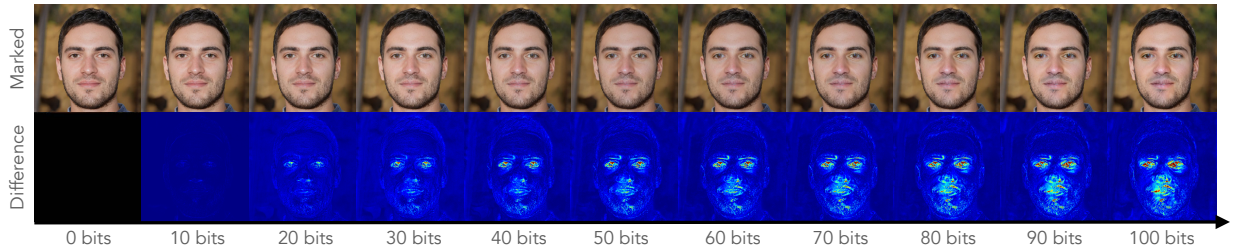


Figure 6.1: A demonstration of our watermark and the impact of the number of embedded bits on the visual image quality. The top row shows the watermarked, synthetic image, and the bottom row shows its difference from the same image without a watermark.

resources [114] and data preparation, including collection, organization, and cleaning. These costs make training image generators a prohibitive endeavor for many. Consequently, generators are often trained by one *provider* and made available to many users through Machine-Learning-as-a-Service [22]. The provider wants to disclose their model responsibly and deter *model misuse*, which is the unethical use of their model to generate harmful or misleading content [161].

Problem. Consider a provider who wants to make their image generator publicly accessible under a contractual usage agreement that serves to prevent misuse of the model. The threat is a user who breaks this agreement and uses the generator to synthesize and distribute harmful deepfakes without detection. To mitigate this threat, companies like OpenAI have deployed invasive prevention measures by providing only monitored access to their models through a black-box API. Users that synthesize deepfakes are detectable when they break the usage agreement if the provider matches the deepfake with their database. This helps deter misuse of the model, but it can also lead to a lack of transparency and limit researchers and individuals from using their technology [218, 50]. For example, query monitoring, which is used in practice by companies such as OpenAI, raises privacy concerns as it involves collecting and potentially storing sensitive information about the user's queries. A better solution would be to implement methods that deter model misuse without the need for query monitoring.

A potential solution is to rely on deepfake detection methods [131, 108, 32]. The idea guiding such methods is to exploit artifacts from synthetic images that separate fake and real content. While these detectors protect well against some deepfakes, it has been demonstrated that they can be bypassed by unseen, improved generators that adapt to existing detectors [52]. As technology advances, it is possible that generators that synthesize virtually indistinguishable images will be developed, rendering passive deepfake detection methods ineffective in the long term.

A different approach to deepfake detection is watermarking [249] when the detection method can access and modify the *target generator*. This is a kind of watermarking that modifies the generator to embed an identifiable message that is later extractable from access to the generated

content using a secret key. Methods such as generator watermarking [248, 249] remain applicable to unseen image generators that could be developed in the future. For deepfake detection, the provider needs a watermark that is extractable from any image synthesized by the generator. We refer to this setting as a *no-box* verification because the verifier only requires access to the generated content but not the generator model. However, there are still several challenges in designing no-box watermarks. These include (i) the ability to embed long messages with limited impact on the model's utility, (ii) the undetectability of the watermark without the secret key, (iii) robustness against removal, and (iv) the method should be efficient.

Solution Overview. Existing watermarking methods are challenging to scale to high-resolution models because they require re-training the generator from scratch, which is computationally intensive [248, 249]. Moreover, while some existing watermarks claim a good capacity/utility trade-off [249], these claims have been limited to relatively small generators. We propose an efficient watermark embedding called Pivotal Tuning Watermarking (PTW) to address these challenges. PTW is the first method to embed watermarks into pre-trained generators and speeds up the embedding process by three orders of magnitude from more than one GPU month when watermarking from scratch [248, 249] to less than one hour. We identify modifications to existing watermarks that speed up their embedding significantly and propose our watermark that can be embedded using PTW with an improved capacity/utility trade-off compared to related work. Figure 6.1 visualizes this trade-off for our watermark.

We propose game-based definitions for robustness and undetectability and evaluate our watermark in two threat models, where the adversary either has access to the generator only through an API (*black-box*) or has control over its parameters (*white-box*). Our results confirm that existing watermarking methods [249, 248] are robust and undetectable in the black-box threat model using existing attacks. We propose three new attacks: a black-box attack called Super-Resolution and two white-box attacks called (1) Overwriting and (2) Reverse Pivotal Tuning. Our experiments indicate that watermarking can be robust in the black-box setting when the attacker has limited access to a high-quality image generator. However, it cannot withstand a white-box attacker with access to only 200 images ($\approx 0.3\%$ of the generator's training dataset) who can remove watermarks at a negligible loss in the generator's image quality.

6.1.1 Contributions

- We propose a watermarking method for pre-trained generators called Pivotal Tuning Watermarking (PTW). PTW does not require any training data, and (ii) is three orders of magnitude computationally less intensive than existing methods [248, 249]
- We modify existing watermarking methods [249, 248] for GANs to be compatible with pre-trained generators, enabling their replication efficiently for large models.
- We provide game-based definitions for robustness and undetectability.

- We propose experiment with three generator architectures (StyleGAN2 [112], StyleGAN3 [114] and StyleGAN-XL [198]) on multiple high-quality image generation datasets.
- We propose black-box and white-box watermark removal attacks. Our results show that watermarking is not robust in practice against our white-box attacks.
- We make our source code publicly available that implements all presented experiments[1].

6.2 Background & Related Work

This section provides a background on generative models and Pivotal Tuning [189], followed by a description of related work on the detection and attribution of deepfakes.

6.2.1 Background

Pivotal Tuning [189]. Pivotal Tuning is a known method to regularize a pre-trained generator while preserving a high fidelity to the generator before tuning. The idea is to preserve the mapping from latent codes to images by cloning and freezing the generator's parameters, referred to as the *Pivot* with parameters θ_G, and then fine-tuning a trainable, second generator θ_G^* with some regularization term $R(\cdot)$. It has been demonstrated that Pivotal Tuning achieves near-perfect image inversion while enabling latent-based image editing [189]. The Pivotal Tuning loss $\mathcal{L}_{\text{PT}}$ is written as follows.

$$\mathcal{L}_{\text{PT}} = \mathcal{L}_{\text{LPIPS}}(x_0, x) + \lambda_{\text{R}} R(x) \tag{6.1}$$

where $x_0 = G(z, \theta_0)$ is an image synthesized by the frozen Pivotal generator using a latent code $z \in \mathcal{Z}$ and $x = G(z, \theta_1)$ is the image generated with the cloned weights θ_1 for the same latent code. The Learned Perceptual Image Patch Similarity (LPIPS) [257] loss $\mathcal{L}_{\text{LPIPS}}$ quantifies a perceptual similarity between images extracted using deep feature extractors.

6.2.2 Related Work

This section summarizes related work on deepfake detection for deep image generators.

Deepfake Detection and Attribution. Deepfake detection and attribution are the tasks of identifying fake images that have been generated or manipulated using deep image generators. Detection focuses only on detecting whether an image is fake, whereas attribution focuses on determining the image's origin. For a given *target generator* that is used to synthesize a deepfake,

[1] https://github.com/nilslukas/gan-watermark

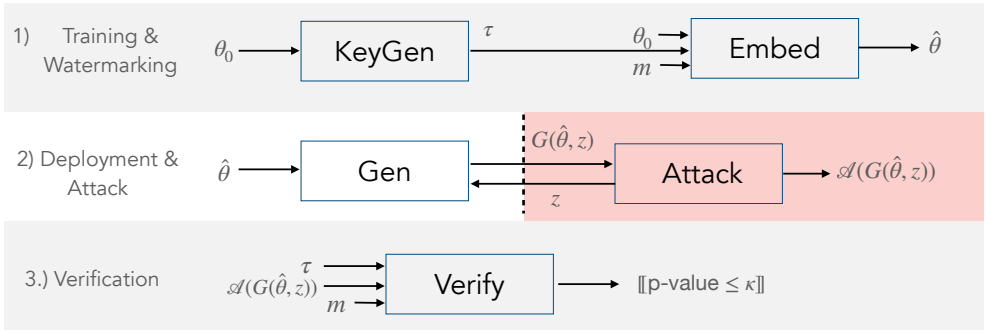


Figure 6.2: Deepfake detection in a *no-box* setting by watermarking the generator.

we taxonomize existing work by (i) the level of access to this target generator and (ii) whether the target generator's parameters can be modified by the detection or attribution method before the deployment of the generator.

(A) Without Generator Access: In this setting, the detection algorithm does not have access to the target generator. Existing work on detecting deepfakes trains classifiers on a public set of deepfakes with known labels for fake/real images [191, 131, 51], exploit semantic incoherence such as asymmetries [153, 93] or low-level artifacts from the generation process [182, 108, 152, 32, 69, 239]. Deepfake attribution methods without access to the target generator apply unsupervised learning methods [248, 73] or only attribute deepfakes to an architecture (and not a generator instance) [243]. Although these methods have proven effective in detecting some deepfakes, it has been shown that they can be evaded by an adversary who adapts to these detectors [52].

(B) With Generator Access: The detection method can have some level of access to the generator, including black-box API or white-box access to its parameters. **(Fingerprinting)** Methods that do not modify the generator's parameters are referred to as *fingerprinting* methods. Recently, attribution methods have been proposed for Latent Diffusion models [201] based on training classifiers on the model's generated data. For GANs, fingerprinting methods all rely on training classifiers on the target generator's data [21, 242, 248]. **(Watermarking)** Methods that modify the generator before deployment are referred to as *watermarking* methods. Yu et al. [249] modify the generator's training data and re-train a watermarked generator. Another approach is to modify the generator's training procedure [250]. All existing methods require training the generator from scratch to embed a watermark.

6.3 Threat Model

Our threat model consists of a defender, who we also refer to as the model provider, and an adversary, who controls a malicious user. Figure 6.2 illustrates attribution through watermarking in

three steps. First, the provider trains a generator and embeds a watermark. Second, the generator is deployed, and a malicious user generates harmful deepfakes at any time after deployment. Finally, deepfakes are attributed to the generator by verifying their watermark.

Adversary's Capabilities. We consider two adversaries that differ in their level of access to the target generator. Our first adversary, the *black-box* adversary, has only API access to the target generator. This means they can query the generator on any latent code $z \in \mathcal{Z}$, but do not have knowledge of its parameters or intermediate activations. The black-box adversary is limited in the number of queries to the generator since queries usually incur a monetary cost to the user. Our second adversary, the *white-box* adversary, has full access to the (watermarked) target generator's parameters, meaning they can tune the generator's weights in an attempt to remove watermarks and generate any number of watermarked images. We use the same definitions for adversaries with black-box API access and white-box access as in Chapter 4.

Both adversaries can access a limited set of R real, non-watermarked images from the same distribution as the defender's training data. An adversary can have limited access to real images without a watermark, which is a commonly made assumption to evaluate the robustness of watermarking [148]. We assume limited availability of real, non-watermarked data in our threat model, as an attacker with sufficient data and computational resources could train and deploy their own generator without a watermark. A black-box attacker could attempt to *extract* the generator [217] by training a surrogate on generated data, but this is out of the scope of our work since model extraction (i) requires massive computational resources and (ii) likely results in surrogate models with a low image quality [148].

Notation	**Description**
$\mathcal{T}$	Stochastic training algorithm
$\mathcal{M}$	Distribution over messages
$\mathcal{D}$	Distribution over images
$\mathcal{D}^n$	Distribution over n images
$D \sim \mathcal{D}$	Draw images D uniformly from $\mathcal{D}$
$y \leftarrow \mathcal{P}(\vec{x})$	Call $\mathcal{P}$ with $\vec{x}$ and assign result to y
$\mathcal{A}$	A procedure denoting an adversary
θ	Parameters of a generator
m	A watermarking message
τ	A secret watermarking key
z	A latent code as input for a generator

Table 6.1: A summary of this chapter's notation.

Adversary's Objective. The common goal of our adversaries is to synthesize images (i) with high visual fidelity to real images and (ii) to generate images that do not retain the watermark.

An image does not retain its watermark if the verification mistakenly outputs zero.

Defender's Capabilities. The defender has access to their own generator's parameters and the secret watermarking key. Their objective is to verify any given image whether it originated from their generator. We refer to the defender's access during verification as *no-box* because, unlike black-box watermarking [3] that can verify the watermark using many queries to the model, with no-box access, the watermark needs to be verified using only one generated image and without control over the query used the generate the image.

6.3.1 Robustness

Algorithm 11 encodes the watermark robustness game given a pre-trained generator θ_0, a confidence threshold for the verification $\kappa \in [0,1]$, a number R of non-watermarked images available to the attacker and a challenge size $K \in \mathbb{N}^+$. The challenge size is the number of images that the adversary has to synthesize to win the game. Note that an adversary with auxiliary access to at least K non-watermarked, real images can always trivially win our security game by returning these real images in their attack. For this reason, we make the assumption that the adversary's auxiliary dataset size $R \ll K$ is much smaller than the challenge size. We choose K=50 000, which allows comparing the quality of the generated images with related work [113, 114].

Algorithm 11 Watermark Robustness Game

1: **procedure** ACCESS(θ) ▷ *Determines adversary's access*
2: **return** θ if white-box else $G(\cdot;\theta)$ ▷ *Returns full model or black-box generator*
1: **experiment** ROBUSTNESS(θ_0, κ, R, K)
2: $\tau \leftarrow \text{KEYGEN}(\theta_0)$ ▷ *Generate watermark key*
3: $m \sim \mathcal{M}$ ▷ *Sample watermark message*
4: $\theta_1 \leftarrow \text{EMBED}(\theta_0, \tau, m)$ ▷ *Embed watermark into model*
5: $D_{\mathcal{A}} \sim \mathcal{D}^R$ ▷ *Attacker's dataset of R images*
6: $X^0, X^1, Y \leftarrow \varnothing$ ▷ *Initialize sets for adversary and verification*
7: $B \sim \{0,1\}^K$ ▷ *K random coin flips*
8: **for** $i \in \{1..K\}$ **do**
9: $X^0 \leftarrow X^0 \cup \{\mathcal{A}(\text{ACCESS}(\theta_1), D_{\mathcal{A}})\}$ ▷ *Adversary generates images*
10: $X^1 \leftarrow X^1 \cup \{\text{GEN}(\theta_0, z) | z \sim \mathcal{Z}\}$ ▷ *Generate control images*
11: $Y \leftarrow Y \cup \left\{ \begin{array}{ll} B_i & \text{if } \text{VERIFY}(X_i^{B_i}, \tau, m) \leq \kappa \\ 1 - B_i & \text{otherwise} \end{array} \right\}$ ▷ *Verification step*
12: $e \leftarrow \frac{1}{K} \sum_{i \in \{1..K\}} (1 - Y_i)$ ▷ *Compute evasion rate*
13: **return** X^0, e ▷ *Return images after evasion and evasion rate*

In Algorithm 11, the game generates a watermark key τ and a watermarking message $m \in \mathcal{M}$ uniformly at random. (lines 2-3). The defender then embeds the watermark into the generator θ_0 to

obtain the watermarked generator θ_1 (line 4). The adversary obtains access to R non-watermarked images (line 5), and K unbiased coin flips dictate the sequence of whether the verifier gets access to the watermarked image after evasion or the non-watermarked image (line 7). A correct verification means that the p-value returned by the verification function is at most κ when a watermark should be present in the image, which we denote with $Y_i = 1$ for the i-th verified image. An incorrect prediction ($Y_i = 0$) means that either (i) the p-value was smaller than κ in a non-watermarked image or (ii) it exceeded κ for a watermarked image after evasion (line 11). Finally, the game returns the images after evasion and the evasion rate e (lines 12-13).

The adversary's success is its expected evasion rate e and the visual quality of its images X^0, which is commonly measured by the Fréchet Inception Distance (FID) [84]. FID is a perceptual distance computed between two sets of images, and a low FID score indicates a high visual similarity between both sets. Similar to the evaluation by Karras et al. [112], we measure the FID between the adversary's images X and the defender's training dataset D. The adversary's success is a trade-off between the expected evasion rate e and the quality of the images X^0.

$$\text{Succ}_{\text{Evasion}} = \mathbb{E}\left[(e - \text{FID}(X^0, D))\right] \tag{6.2}$$

6.3.2 Detectability

A detectable watermark poses a threat because it could facilitate an adversary in locating and removing the watermark or spoofing it. Algorithm 12 presents the watermark detectability game for a generator θ_0 and the attacker's access to R_1 non-watermarked and R_2 watermarked images from the generator. Our game challenges the adversary to determine the presence of a watermark.

The defender generates a watermarking key (line 2) and embeds a randomly sampled watermarking message into their pre-trained generator θ_0 (lines 3-4). The attacker then gets access to R_1 images without a watermark and R_2 images with a watermark that are samples given random latent codes not controlled by the attacker (lines 7-9). A fair coin is flipped, deciding whether the detection attack operates on images containing a watermark, and the attacker has to predict the coin flip (lines 7-8). The success of the detectability attack is its expected classification accuracy.

$$\text{Succ}_{\text{Detection}} = \mathbb{E}\left[a\right] \tag{6.3}$$

Contrasting with Related Work. Related work has proposed other methods to measure detectability that sample synthetic images from generators trained on the same dataset with different seeds [248]. However, in these approaches, it is unclear whether the detection was successful because the watermark has been detected or because some other patterns make each generator instance identifiable (e.g., a fingerprint). Our notion of detectability can be attributed solely to the impact of the watermark in the synthesized image.

Algorithm 12 Watermark Detection Game

experiment $\textsc{Detectability}(\theta_0, R_1, R_2)$
  $\tau \leftarrow \textsc{KeyGen}(\theta_0)$
  $m \sim \mathcal{M}$
  $\theta_1 \leftarrow \textsc{Embed}(\theta_0, \tau, m)$
  $D_1 \leftarrow \cup_{i=1..R_1}\{\textsc{Gen}(z, \theta_0) | z \sim \mathcal{Z}\}$ ▷ *Non-watermarked images*
  $D_2 \leftarrow \cup_{i=1..R_2}\{\textsc{Gen}(z, \theta_1) | z \sim \mathcal{Z}\}$ ▷ *Watermarked images*
  $b \sim \{0, 1\}$ ▷ *Unbiased coin flip*
  $\hat{b} \leftarrow \mathcal{A}_{\text{Detect}}(\{G(z, \theta_b) | z \sim \mathcal{Z}\}, D_1, D_2)$ ▷ *Detection attack*
  $a \leftarrow [\![\hat{b} = b]\!]$ ▷ *Determine correctness*
  return a

6.4 Conceptual Approach

This section describes our proposed embedding method for watermarking image generators. We describe improvements of our embedding method over existing methods. Then, we modify two existing watermarks for GANs to embed into pre-trained generators and propose our improved GAN watermark. Finally, we propose three attacks against the robustness of watermarking.

6.4.1 Pivotal Tuning Watermarking

Pivotal Tuning Watermarking (PTW) is a method for watermarking a pre-trained generator to preserve latent similarity with the Pivot. On input of the same latent code $z \in \mathcal{Z}$, both generators should produce a similar image with imperceptible modifications. Given a pre-trained generator θ_0 and watermark decoder θ_D, a watermarking message $m \in \mathcal{M}$, a number of iterations N, a regularization parameter λ_R and a learning rate α, PTW returns a watermarked generator $\hat{\theta}$ with high output fidelity to the generator before watermarking for the same latent codes.

The watermark decoder neural network extracts messages from images. Let θ_D be a decoder neural network that extracts messages from images, and m is a message that should be embedded. Let λ_R be the strength of the watermark regularization term, α be the learning rate, and N be the number of steps to optimize the generator θ.

Algorithm 13 creates a copy of the pre-trained generator's parameters, called the *pivot*, and enters a loop (lines 2-3). A random latent code is drawn and passed through both variants of the generators to produce two images x_0, x (lines 4-6). The loss is computed according to Equation (6.1) where we compute a binary cross-entropy H on the extracted and target messages (line 7). The optimization tries to encode as many bits of the message as possible into x while minimizing the LPIPS loss to x_0 generated by the pivot. The model parameters are updated

Algorithm 13 Pivotal Tuning Watermarking

1: **procedure** EMBED($\theta_0, \tau, m, N, \lambda_R, \alpha$)
2: $\tau \leftarrow$ KEYGEN(θ_0) ▷ *Requires a key where the verification is efficiently optimizable*
3: $\hat{\theta} \leftarrow$ copy parameters from θ_0 ▷ *Freeze the Pivot* θ_0
4: **for** $i \in \{1..N\}$ **do** ▷ *Embedding loop*
5: $z \sim \mathcal{Z}$
6: $x_0 \leftarrow$ GEN(z, θ_0) ▷ *Frozen Pivot*
7: $x \leftarrow$ GEN($z, \hat{\theta}$) ▷ *Watermarked image*
8: $g_{\hat{\theta}} \leftarrow \nabla_{\hat{\theta}} \mathcal{L}_{\text{LPIPS}}(x_0, x) + \lambda_R \text{VERIFY}(x, \tau, m)$ ▷ *Balance quality and message error rate*
9: $\hat{\theta} \leftarrow \hat{\theta} - \alpha \cdot \text{Adam}(\hat{\theta}, g_{\hat{\theta}})$ ▷ *Update the generator*
10: **return** $\hat{\theta}$ ▷ *Watermarked generator*

iteratively (line 8), and the tunable generator's parameters and the trained decoder are returned. The decoder represents the secret watermarking key (line 9).

Overview. Any image generator that maps from a latent space to images (see Section 6.2.1) is compatible with PTW. PTW can also be used to embed any image watermarking method [8, 262] for that can be learned by a watermark decoder network. We highlight three advantages of PTW over existing embedding methods for GANs [248, 249].

1. **Speed**: PTW enables watermarking a pre-trained generator up to three orders of magnitude faster than watermarking from scratch.
2. **No Training Data**: PTW requires access only to the generator but not to any training data or the discriminator.
3. **Post-Hoc**: PTW allows embedding watermarks as a post-processing step into any pre-trained generator.

As stated in Algorithm 13, PTW requires access to a differentiable process to extract a message from an image. We use a binary message space $\mathcal{M}$ and a deep image classifier θ_D to extract multi-bit messages from images. We propose a KEYGEN procedure that generates watermarking keys by leveraging optimization criteria. Then, we show how existing watermarking methods can be modified to embed them into pre-trained generators efficiently as a baseline for our watermarks.

6.4.2 Generating a Watermarking Key through Optimization

We now present the implementation of KEYGEN that leverages an optimization procedure to derive a watermarking key τ. The watermarking key consists of parameters of a deep image classifier θ_D that can extract multi-bit messages from images and allow backpropagating through

the extraction process because the classifier is efficiently differentiable The ability to efficiently optimize the watermark decoder is central to our approach.

We generate a key with knowledge of the generator's parameters, which is useful for learning which pixels can be modified easily to hide messages with minimal impact on the visual image quality of the generator. For example, a generator trained to synthesize faces likely allows encoding more bits per pixel for central pixels in the image that belong to a face, as opposed to pixels at the edge that encode the background. The challenge is that none of the existing pre-trained GAN architectures allow the input of a message; hence, the problem becomes how to modulate a message to a generator to subsequently encode this message into its generated images.

Overview. We use three methods to modulate a message to a generator without modifying its architecture or re-training the generator from scratch. All three options are based on training *mapper* neural networks, which map a message to a perturbation of the generator's parameters [250] or (ii) its inputs [174] (i.e., the latent codes). We implement the parameter mapper by adding the perturbation to a subset of the GAN's weights. In our experiments, we modulate $\leq 1\%$ of its total parameters in practice. Our parameter-mapper is similar to parameter-efficient fine-tuning (PEFT) [92] techniques and universally applies to any generator architecture. Our implementation focuses on modulating convolutional layers of GANs [254].

Algorithm 14 Generating Keys through an Optimization Procedure

procedure KEYGEN($\theta_0, n, N, \lambda_R$)
  $\theta_P, \theta_Z, \theta_D \leftarrow$ random initialization ▷ *Parameter and latent mappers and message decoder*
  $\theta \leftarrow \{\theta_P, \theta_Z, \theta_D\}$ ▷ *θ denotes all trainable parameters*
  for $i \in \{1..N\}$ **do**
    $m \sim \mathcal{M}$
    $\theta_P \leftarrow$ MAPPER$(\theta_P, \theta_0, m, z)$ ▷ *Perturbation of parameters*
    $\hat{z} \leftarrow$ MAPPER$(\theta_Z, \theta_0, m, z)$ ▷ *Perturbation of latents*
    $x \leftarrow$ GEN$(\theta_0 + \theta_P, \hat{z} + z)$ ▷ *Image containing a watermark*
    $x_0 \leftarrow$ GEN(z, θ_G) ▷ *No watermark*
    $g_\theta \leftarrow \nabla_\theta \mathcal{L}_{\text{LPIPS}}(x_0, x) + \lambda_R$VERIFY$(x, \theta_D, m)$
    $\theta \leftarrow \theta - \alpha \cdot \text{Adam}(\theta, g_\theta)$ ▷ *joint update of all trainable parameters*
  return θ_D

We now describe our key generation algorithm that trains a deep image classifier to extract messages. Let MAPPER$(\theta_0, \theta_1, m, z)$ be a function that takes the parameters of a mapper θ_0, a GAN generator θ_1, a message m and a latent input $z \in \mathcal{Z}$ to predict a perturbation. Parameter mappers predict parameter perturbations, whereas latent mappers predict a perturbation to latents.

Training the Decoder. Algorithm 14 presents pseudocode for the key generation procedure. First, we randomly initialize a parameter mapper θ_P, a latent mapper θ_Z, and a message decoder

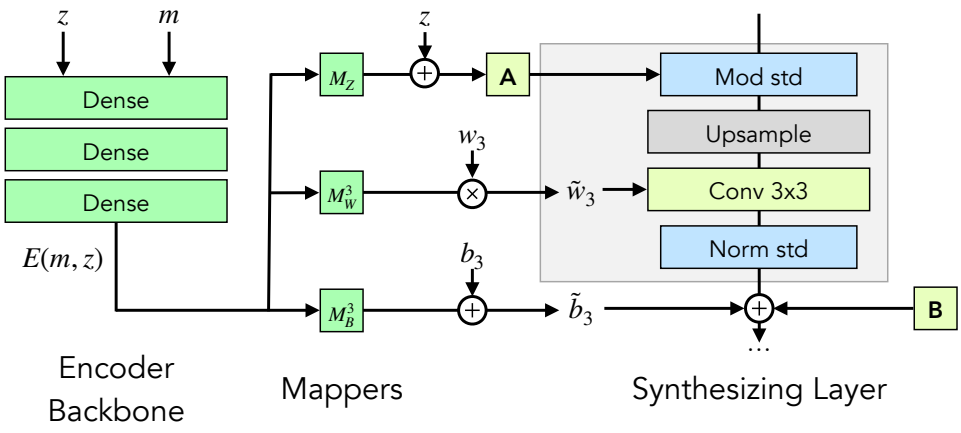


Figure 6.3: An exemplary illustration of the mappers for a single generator synthesis layer (adapted from [113]). On input of a latent code z and a message m, the mappers (in green) modulate the generator's weights and inputs. A is an affine transform and B is random noise sampled during inference.

θ_D (lines 2-3). Then, we iteratively optimize all trainable parameters by randomly sampling a message and predicting a parameter and a latent perturbation (lines 5-7). We generate a watermarked and non-watermarked image and calculate the gradient on the loss between the quality difference and the verification loss (lines 8-10). Finally, we update all trainable parameters (line 11), and at the end of the optimization, we return the parameters of the decoder (line 12). Figure 6.3 illustrates the modulation of a message to a synthesizing layer in a generator.

The returned decoder can extract messages from any image. Notably, the decoder learns an encoding of the message that causes the least degradation in visual image quality (according to the LPIPS loss). We refer to Figure 6.1, where the most perturbed pixels are pixels with high semantic value, such as the eyes and nose of a face. Since the decoder depends on the generator instance, a different decoder should be trained for each instance for the best visual quality, but decoders can be re-used if the model architectures are similar. Next, we describe modifications to existing watermarks [248, 250] that allow embedding them into pre-trained generators.

6.4.3 Modifying Existing Watermarks

Two existing watermarks for GANs require re-training the generator from scratch to embed a watermark [248, 249]. This section describes modifications to both methods for watermarking pre-trained generators as a baseline comparison to our method. First, we briefly summarize both watermarking methods and then describe our modifications to enhance their efficiency.

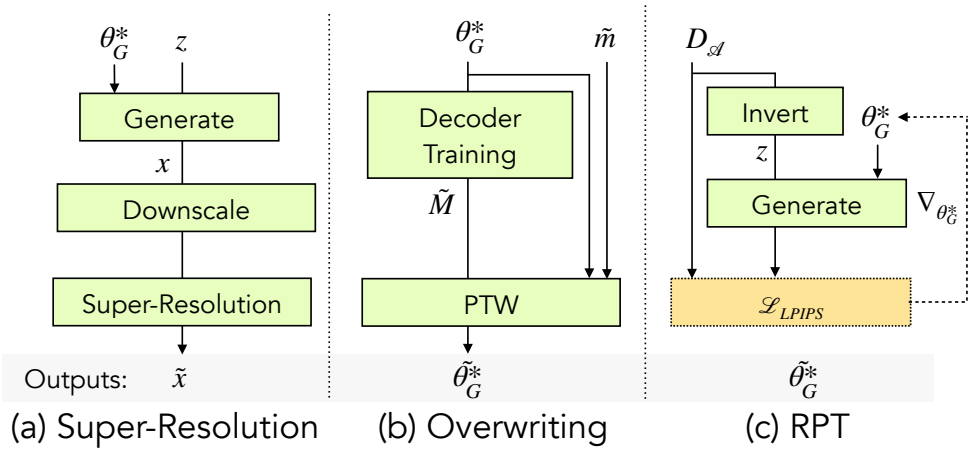


Figure 6.4: An illustration of our three attacks against the robustness of generator watermarking. RPT stands for Reverse Pivotal Tuning. The function INVERT maps images to latent codes, and DOWNSCALE reduces the resolution of an image.

Summary. The first watermark, which we call `Yu1` [248], trains an encoder-decoder network on real images by marking the images with imperceptible patterns. The GAN is trained from scratch on the marked training data. The second watermark, which we call `Yu2` [249], modifies the GAN's training objective and is trained from scratch. Some of their modifications were invented to mitigate training instabilities such as mode collapse [219] or consistency losses, which are problems that do not appear during fine-tuning. The authors modulate the watermark with a parameter mapper θ_P and embed the watermark by adding the predicted parameter perturbation to the generator's weights. We find that embedding without PTW has a substantial impact on the generator's image quality and that using PTW can preserve a much higher image quality. We refer to the authors' papers for a more detailed description of their works.

Modifications. The required modifications for `Yu1` are straightforward: Instead of training on real training data, we stamp synthetic data and use their decoder network for embedding a watermark with PTW. For `Yu2`, we ignore all additional losses that address training instabilities or consistency losses and train their weight mapping network instead via fine-tuning on synthetic data while freezing the generator's weights. We embed the `Yu2` watermark using the author's approach of summing the weight mapper's prediction to the generator's parameters.

6.4.4 Attacks to Evade Watermark Detection

This Section proposes three novel adaptive attacks against the robustness of model watermarking for image generators. Recall from Section 6.3 that we consider two adversaries: a black-box and a

white-box adversary. Previous work [248, 249] assumes only a black-box attacker who can execute any of these five attacks: blurring, cropping, image noising, JPEG compression, or quantization. We refer to Appendix C for a detailed description of all attacks and parameters, including those from previous work. Figure 6.4 illustrates our proposed black box and two white-box attacks.

Black-box Attacks

Our black-box attacker first scales the resolution of an image down by a factor ρ and then uses *super-resolution* models [190] to upscale the image to its previous resolution. A super-resolution model can upscale images by interpolating details that are not sharp in the low-resolution image. Such super-resolution models enable an attacker to apply stronger perturbations to images in an attempt to remove their watermark with a smaller impact on the image quality. Super-resolution models have been demonstrated to generalize well to out-of-domain data, meaning the attacker does not need access to the generator's training data. While our attacks use pre-trained models from related work [190] to achieve super-resolution, we are the first to apply super-resolution models to undermine watermarking in image generators. Previous attacks [248, 250, 249] use image augmentation techniques such as blurring or noising to remove the watermark.

White-box Attacks

We propose two adaptive white-box attacks called *Overwriting* and *Reverse Pivotal Tuning* (RPT). In the overwriting attack, the attacker trains their own watermark decoder (see Algorithm 14) and then uses PTW for watermarking the generator using a random message. The success of the overwriting attack depends on the similarity between the defender's watermarking key τ and the attacker's watermarking key $\tilde{\tau}$. The overwriting attack can successfully remove the watermark if both keys modulate similar pixels in the input.

Algorithm 15 (White-box) Reverse Pivotal Tuning (RPT)

procedure $\textsc{Invert}(x, \hat{\theta})$
  return $\arg\min_{z \in \mathcal{Z}} \mathcal{L}_{\text{LPIPS}}(\textsc{Gen}(\hat{\theta}, z), x)$
procedure $\text{RPT}(\hat{\theta}, D, N, \alpha)$
  $Z \leftarrow \{\textsc{invert}(x, \hat{\theta}) | x \in D\}$
  for $i \in \{1..N\}$ **do**
    $g_{\hat{\theta}} \leftarrow \nabla_{\hat{\theta}} \mathcal{L}_{\text{LPIPS}}(\textsc{Gen}(\hat{\theta}, Z_i), D_i)$
    $\hat{\theta} \leftarrow \hat{\theta} - \alpha \cdot \text{Adam}(\theta, g_{\hat{\theta}})$
  return $\hat{\theta}$

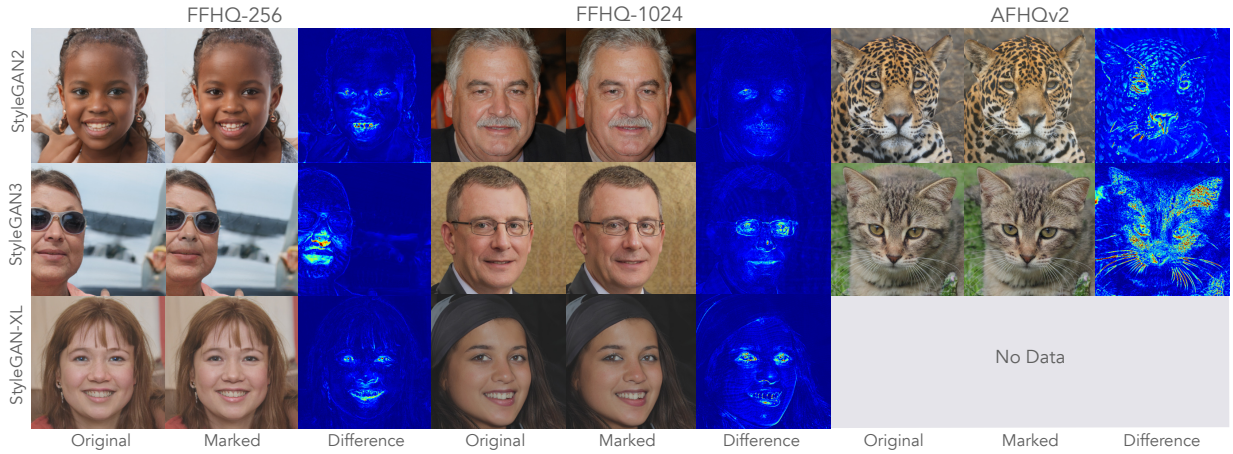


Figure 6.5: Images synthesized using our watermarked generators on different datasets and model architectures. We show the image synthesized by the generator (i) before and (ii) after watermarking and (iii) the difference between the watermarked and non-watermarked images. StyleGAN-XL does not provide a pre-trained model checkpoint for AFHQv2.

Algorithm 15 implements our RPT attack. The attacker has access to a watermarked generator θ_G^*, a limited set of R non-watermarked, real images D^R, and performs the RPT attack for N steps with a learning rate α. Their goal is to regularize the generator to synthesize images that are visually similar to their real images (i.e., they have high visual quality) but do not retain a watermark. The RPT attack consists of two stages: (1) inversion of the real images (line 5) and (2) Pivotal Tuning so that the inverted images have a high visual similarity to the real images (lines 5-6). RPT should be successful with the availability of many non-watermarked images.

6.5 Experiments

We describe our experimental setup and specify measured quantities, namely capacity, utility, detectability, and robustness. Then, we measure detectability and robustness (see Section 6.3) and compare our watermarking method to the modified methods from related work (see Section 6.4.3).

6.5.1 Experimental Setup

Datasets. We experiment with three datasets for which pre-trained, high-quality generators have been made publicly available. FFHQ [112] consists of 70k human faces in various poses. We experiment with a lower-resolution version of the dataset at 256^2 pixels, which we refer to as

FFHQ-256, and the high-quality version FFHQ at $1\,024^2$ pixels. In addition, we experiment with AFHQv2 [36], which consists of roughly 16k animal images and has a resolution of 512^2 pixels.

Pre-Trained Generators. We experiment with three StyleGAN-based architectures: StyleGAN2 [112], StyleGAN3 [114] and StyleGAN-XL [198]. We select the StyleGAN architectures because (1) this architecture achieves state-of-the-art FID values on the surveyed datasets and (2) many high-quality model checkpoints are publicly available that have been trained with different seeds[2][3]. State-of-the-art image generator architectures are evaluated using the same checkpoints that we are using. Therefore, the image quality of our watermarked generated images can be compared to the image quality in ongoing research on image generation models.

Framework. We implement all watermarking methods from scratch in PyTorch 1.13. While implementations for `Yu1`[4] and `Yu2`[5] exist, we could not reproduce their results with the provided implementation. `Yu1` never converges, and `Yu2` is implemented in Tensorflow version 1, which is no longer supported by modern GPUs, meaning we cannot reuse their source code or load the provided generator checkpoints.

6.5.2 Metrics

Utility. Similar to existing work [112], we measure the utility of a generator by its Fréchet Inception Distance (FID) [84]. Lower FID indicates a higher utility. Like Karras et al. [112], we measure FID between $50,000$ generated and real images. For AFHQv2, we use only $16\,000$ real images due to the limited dataset size.

Capacity. We measure the capacity of a watermark in bits by the difference in the expected number of correctly extracted bits from watermarked and non-watermarked images. The expected rate of correctly extracted bits equals 0.5 for non-watermarked images, assuming messages are randomly sampled. Let $m \in \{0,1\}^n$ be a message, τ the secret watermarking key, and θ are the parameters of a generator. The capacity of the generator is computed as follows.

$$C_\theta = n \cdot \mathop{\mathbb{E}}_{z \in \mathcal{Z}} \Big[\mathrm{BER}(\textsc{Extract}(G(z;\theta),\tau), m) - 0.5 \Big] \tag{6.4}$$

The function BER computes the bit error rate between messages. It is straightforward to achieve a high capacity by overwriting a significant portion of the image. However, doing so also decreases the image's quality, which can be measured and visualized as the capacity/utility trade-off.

Decision Threshold. We consider a watermark to be *removed*, if we can reject the null hypothesis H_0 with a p-value less than 0.05. The null hypothesis states that k matching bits

[2] https://github.com/NVlabs/stylegan3
[3] https://github.com/autonomousvision/stylegan-xl
[4] https://github.com/ningyu1991/ArtificialGANFingerprints
[5] https://github.com/ningyu1991/ScalableGANFingerprints

Model	StyleGAN2	StyleGAN3	StyleGAN-XL
FFHQ-256	158h	482h	552h
FFHQ-512	384h	662h	1285h
FFHQ-1024	929h	1161h	1456h

Table 6.2: Time measured in GPU hours required for training generators *without* watermarking (from scratch) on FFHQ [112] on 8xA100 GPUs.

were extracted from the synthetic images by random chance. Quantitatively, the probability of this event is calculated as $\Pr(X > k|H_0) = \sum_{i=k}^{n} \binom{n}{i} 0.5^n$. In practice, for a watermark with $n = 40$ bits, we need to extract at least 26 bits correctly, meaning that we verify the presence of a watermark by correctly extracting $C_\theta \geq 6$ in bits.

6.5.3 Runtime Analysis

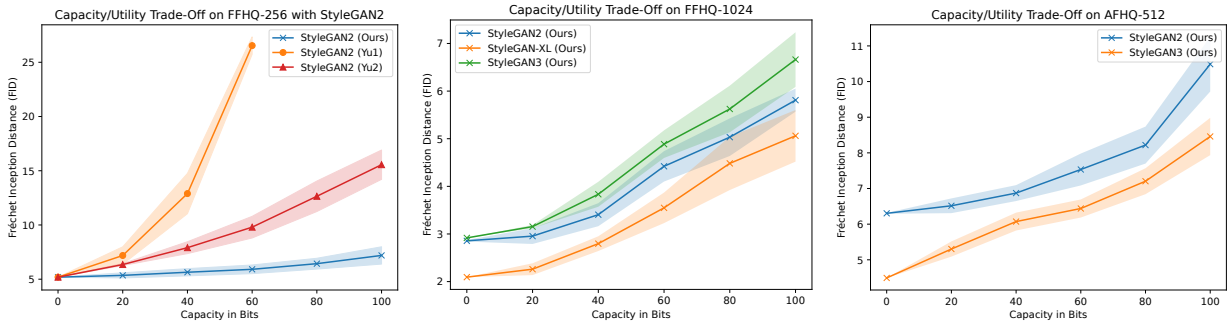
To calculate the speed-up of PTW over other methods [250, 249], we compare it with training non-watermarked generators from scratch. This comparison is fair, as watermarking is not expected to decrease a generator's training time. We estimate the total runtimes in GPU hours using the suggested parameters in the relevant GAN papers [113, 114, 198] on 8xA100 GPUs.

Table 6.2 shows the estimated training runtimes from scratch for each generator on FFHQ [113] at varying pixel resolutions. For instance, training a StyleGAN-XL model on FFHQ at a resolution of 256^2 pixels requires 552 GPU hours. With PTW, watermarking a pre-trained generator on FFHQ requires only about 0.5 GPU hours, which is three-orders of magnitude improvement for high-resolution generators. Our approach also requires training the watermarking decoder (see Algorithm 14), which is a one-time upfront cost of about 2 GPU hours.

6.5.4 Capacity/Utility Trade-off

This section summarizes our results on the capacity/utility trade-off on various datasets and model architectures and compares them to two existing, modified watermarks: `Yu1` and `Yu2`.

Visual Inspection. Figure 6.5 shows images synthesized by our watermarked generators on all three surveyed datasets. The columns show the original image synthesized before and after watermarking and their differences as a heatmap. Heavily modified regions are highlighted in yellow and red. In both versions of the facial image datasets, we observe that our watermark focuses on pixels located on the face of the generated person, most prominently its eyes. Upon closer inspection, the network modifies the eyes and mouth area of a face strongest and is invariant



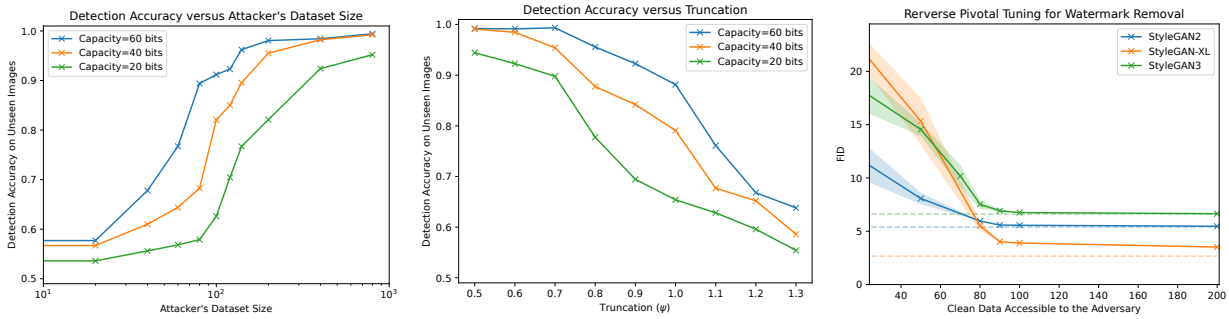
(a) FFHQ-256. (b) FFHQ-1024. (c) AFHQv2-512.

Figure 6.6: The capacity/utility trade-off. Figure 6.6a shows our watermark compared to two existing, modified watermarks for StyleGAN2. Figure 6.6b shows our watermarking for on FFHQ-1024 using three different generator architectures. Figure 6.6c shows our watermark on a domain other than faces (wild animals) using different generator architectures. The shaded area illustrates the standard deviation for three repetitions.

to the location of the face in the image. For AFHQv2, we observe that the pixels are more spread out onto the entire image. In the next subsection, we compare our watermark quantitatively to other existing watermarks.

Comparison to Existing Watermarks. Figure 6.6a shows the trade-off on FFHQ-256 for different watermarking methods using a StyleGAN2 architecture. We plot the embedded bits C_θ against the FID. Our method outperforms the other two approaches substantially, as we can embed 100 bits with a similar loss in utility as embedding 20 bits using `Yu2`. In contrast, `Yu1` is not competitive, even though it employs PTW as its embedding strategy. Upon analyzing the generated images, we observe that `Yu1` is not sensitive to the capacity per pixel and attempts to encode many bits of the message into background pixels, which noticeably deteriorates the generator's quality. We believe this method works better when the entire generator is re-trained from scratch, as the generator can learn to allocate capacity to arbitrary pixels. For FFHQ-256, using our watermark, encoding 40 bits only worsens the FID by approximately 0.3 points.

Watermarking High-Quality Generators. Figure 6.6b shows the trade-off using our watermark across three generator architectures on FFHQ-1024. Compared to FFHQ-256, which has a FID of over 5, the generators trained on FFHQ-1024 have a much lower FID of less than 3. Our watermark embeds up to 40 bits with little loss in utility, but the FID deteriorates quickly when embedding more than 40 bits. For StyleGAN2, we measure a FID deterioration of almost 3 points when embedding 100 bits. We believe the effect on the FID is greater for the high-quality dataset due to two factors: (1) high-quality images with a low FID may be more sensitive to modifications, and (2) our watermark decoder downscales images to 224^2 pixels, which means that



(a) (b) (c)

Figure 6.7: (a) The detection accuracy in relation to the adversary's dataset size for different capacities without truncation. (b) The detection accuracy plotted against truncation when fixing the adversary's dataset size to ≤ 100 labeled images. (c) An ablation study for the number of real, non-watermarked data used during our adaptive Reverse Pivotal Tuning attack. The shaded areas denote the standard deviation (N=3), and the dashed horizontal lines show the generator's FID before the attack.

our decoder cannot extract more information from larger images. Our decoder is a ResNet18 [82] model designed for this resolution. Nonetheless, we demonstrate that watermarking high-quality generators is possible using our method.

Watermarking Different Domains. Figure 6.6c shows the capacity/utility trade-off across two generator architectures for the domain of animal images. We cannot evaluate StyleGAN-XL on AFHQv2 because no pre-trained checkpoint was available for this dataset. We aim to demonstrate that our watermark is not restricted to just the facial image domain. Figure 6.6c shows that our approach can embed watermarks up to 100 bits, although we observe a strong deterioration in FID of more than 4 points, at which point the watermark is (barely) visually perceptible. While it is possible to embed 100 bits, given our results, we believe that 40 bits are more practically relevant as the deterioration in FID is less than one point for StyleGAN2, and the watermark is not easily perceptible. Interestingly, the deterioration in FID is stronger for the animal domain, which we attribute to a larger output diversity. AFHQv2 contains images of multiple different animal species in diverse poses.

6.5.5 Detectability

Our first experiment measures the detectability of our watermark at different capacities. We recall from our security game in Algorithm 12 that the attacker can access R_1, R_2 non-watermarked and watermarked images. In our experiments, we set $R_1 = R_2$ and evaluate the detectability of

images synthesized by generators that have been watermarked with varying capacities. We use a standard, pre-trained ResNet-18 and fine-tune it for the detection task using an Adam optimizer.

Detectability versus Dataset Size. Figure 6.7a shows the detection accuracy p of our attack against the number of labeled images available to the adversary. As expected, a higher watermark capacity results in a higher detectability of watermarked images. We observe that an adversary with access to ≥ 400 labeled images has a classification accuracy of over 90% in classifying watermarked/non-watermarked images for any capacity. An adversary with access to ≤ 100 images cannot reliably detect watermarked images if the capacity is at most 40 bits. Our results show that the detection algorithm cannot detect our watermark unless the attacker can access a relatively large set of labeled, non-watermarked images. Next, we evaluate the influence of the latent code's sampling strategy on the watermark detectability.

Truncation Trick. Generating an image from a GAN requires sampling a latent code. The *truncation trick* [16] is a technique used for generative models to limit the range of values for a latent code and allows for controlling the diversity and quality of the generated images. If the truncation threshold ψ is low, samples will have a high similarity to the real training data but limited diversity, meaning they may appear similar to each other. We identify that truncation plays a significant role in the ability of an adversary to detect watermarks. Figure 6.7b shows that the detection accuracy decreases with an increasing diversity of the generator's output when fixing the number of samples available to the adversary.

6.5.6 Robustness

We evaluate the watermark's robustness against several types of attacks, including (1) black-box attacks like cropping, blurring, quantization, noising, JPEG compression, and our super-resolution attack, and (2) two white-box attacks, overwriting, and Reverse Pivotal Tuning (see Section 6.4.4). We refer to Appendix C for describing the attacks and parameters we use during our evaluation. All attacks are evaluated against generators that have been watermarked with a capacity of at least $C_\theta \geq 40$ bits. Embedding 40 bits only deteriorates the generator's FID by about 0.3 points on average for StyleGAN2 on FFHQ.

Black-box Attacks

Latent Space Analysis. We examine whether there are points in the generator's latent space that synthesize high-quality images without a watermark. If such points exist, an attacker could attempt to find them and sample the generator on these points. We test for such latent subspaces using three sampling methods: (i) truncation, (ii) latent interpolation, and (iii) style-mixing (for the StyleGAN architectures). Truncation restricts the distance of a latent code to the global average. Latent interpolation samples point on a line between latent codes, and style-mixing

combines intermediate latent codes $w \in \mathcal{W}$ and feeds them to the generator [112]. Our results show that the mean capacity remains unaffected by the sampling method, meaning we were able to successfully extract the watermark message in all cases. We conclude from these results that the watermark generalizes to the generator's entire latent space.

Removal Attacks. Next, we perform all surveyed removal attacks against all watermarked generators and measure the evasion rate and FID with $K = 50,000$ synthetic images. A summary of all black-box attacks is shown as a scatter plot in Figure 6.8. The Figure shows the remaining capacity after an attack on the x-axis and utility (measured by the FID) on the y-axis. The Pareto front, which represents the optimal trade-off between capacity and utility, is highlighted and represents the best attack out of all surveyed attacks that an adversary could choose. We find that none of the black-box attacks are effective at removing a watermark, but our super-resolution attack is always part of the Pareto Front.

The Pareto front represents the best capacity/utility trade-off a black-box attacker can achieve using these attacks. For example, a black-box attacker can reduce the capacity by 10 bits from 50 to 40 but, in doing so, reduces the FID by over 6 points. Our super-resolution attack is on the Pareto front but it cannot remove the watermark. Removal is only possible when the FID drops to 30, at which point the image quality has been compromised. Table 6.3 summarizes the best-performing black-box attacks for the three evaluated generator architectures. Each attack has a single parameter that we ablate over using grid search. We refer to Appendix C for a detailed description of all attacks and parameters we used in this ablation. Table 6.3 lists those data points that either remove the watermark ($C_\theta < 5$) or, if the watermark cannot be removed, the data point with the lowest FID. None of the black-box attacks, including our super-resolution attack, successfully remove the watermark while preserving the generator's utility.

White-box Attacks

Overwriting. Table 6.3 shows that overwriting can remove watermarks but deteriorates the generator's image quality, measured using FID, by approximately 3 points for StyleGAN2 and 6 points for StyleGAN-XL. Such a deterioration in FID likely prevents attacks in practice because low-quality deepfakes are more easily detectable. Overwriting implicitly assumes knowledge of the defender's watermarking method, which may not be true in practice. Overwriting could cause a noticeable decline in FID if the attacker's and defender's watermarking methods differ.

Reverse Pivotal Tuning is substantially more effective than the overwriting attack as it preserves the FID of the generator to a greater extent. We found that an attacker accessing 200 real, non-watermarked images can remove any watermark without causing a noticeable deterioration in FID. This means that with access to less than 0.3% of the training dataset, a white-box adversary can remove any watermark. In the case of StyleGAN-XL, using 200 images leads to a decrease in FID of less than one point (from 2.67 to 3.52).

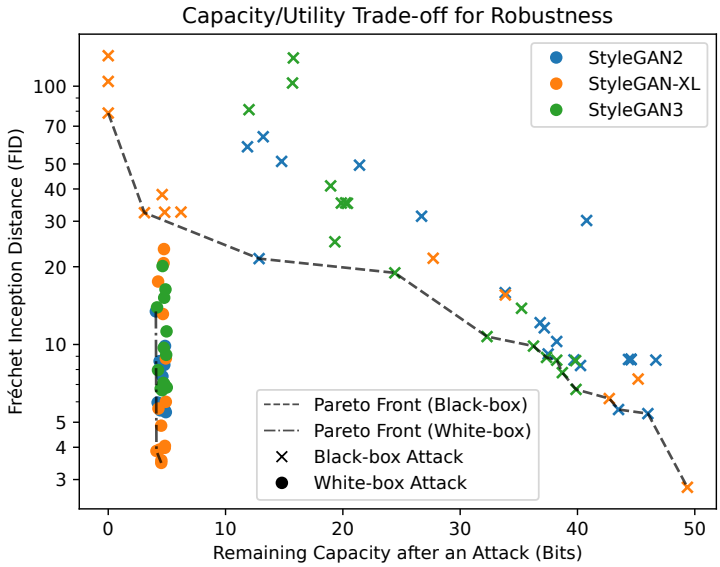


Figure 6.8: This Figure shows the robustness of our watermark against all surveyed attacks. We highlight black-box and white-box attacks that are in the Pareto front.

	StyleGAN2		StyleGAN-XL		StyleGAN3	
	C_θ	FID	C_θ	FID	C_θ	FID
Attacks	43.05	5.4	48.79	2.67	40.33	6.61
Black-box Attacks						
Crop	39.73	8.72	42.71	6.18	38.23	8.69
Blur	38.82	36.84	12.12	10.32	35.12	11.73
JPEG	42.12	8.70	38.43	9.12	38.23	9.33
Noise	40.26	8.29	45.17	7.35	32.29	10.73
Quantize	37.17	11.60	43.27	5.61	39.72	8.71
SR	32.86	11.51	34.52	11.62	30.12	11.34
White-box Attacks						
Overwrite	4.78	8.34	4.91	8.83	4.73	9.71
RPT_{200}	4.91	5.47	4.52	3.52	4.59	6.65
RPT_{100}	4.44	5.56	4.21	3.90	4.47	6.75
RPT_{50}	4.38	8.07	4.38	15.32	4.16	14.47

Table 6.3: The capacity and FID of all surveyed attacks. We ablate over multiple parameters for each attack and this table shows the points with the best (i.e., lowest) FID. RPT_R stands for the Reverse Pivotal Tuning attack using R real samples.

Ablation Study for RPT. Figure 6.7c shows an ablation study over the amount of real, non-watermarked training data an attacker requires to remove a watermark. We measured these curves as follows: We randomly sample a set of R real images and run the RPT attack encoded by Algorithm 15 with gradually increasing weight λ_{LPIPS} on the LPIPS loss until the watermark is removed. Then, we compute the FID on $K = 50,000$ images. In all experiments, the watermark is eventually removed, but access to more data has a significant impact on the FID that is retained in the generator after the attack. For StyleGAN2, we find that 80 images ($\approx 0.1\%$ of the training data) can remove the watermark at less than 0.3 points of deterioration in FID, representing a visually imperceptible quality degradation. Our results demonstrate that an adaptive attacker with access to the generator's parameters can remove any watermark using only a small number of clean, non-watermarked images and can threaten the trustworthiness of watermarking.

6.6 Discussion

This section discusses the limitations of watermarking and our study, the extension of our work to other image generators, and ethical considerations from releasing our attacks.

Non-Cooperative Providers. Our study indicates that watermarking for image generators can be robust under restrictive threat models (see Section 6.3), where the attacker is limited in their access to capable image generators. For watermarking to effectively control misuse, every provider has to cooperate and watermark their generators before disclosing them. However, this is unlikely to occur in practice [237]. A capable adversary with access to sufficient training data and computational resources can always train their own generator without a watermark and provide it to others. It is important to acknowledge this inherent limitation of watermarking methods, as they cannot prevent this scenario. Nonetheless, they can act as a deterrent to adversaries who cannot train their own generators from synthesizing harmful deepfakes.

Watermarking from Scratch. We focus on the robustness and undetectability of watermarking methods on pre-trained image generators due to the substantially higher scalability than watermarking from scratch. Further research is needed to explore the potential impact of watermarking from scratch on robustness and undetectability.

Watermarking for Intellectual Property Protection. Watermarking has also been used for Intellectual Protection (IP) protection of neural networks [168, 147, 227]. The IP protection threat model typically assumes white-box access of the adversary to the target generator and black-box API access of the defender to the adversary's generator to verify a watermark. For example, Ong et al. [168] embed a *backdoor* into a generator that synthesizes images containing a watermark when the generator is queried with certain latent codes. All watermarks evaluated in this chapter assume no-box access to the target generator, meaning that our watermarks can be extracted from any of the generator's synthetic images. We show that existing no-box watermarks

are not robust in the white-box setting, meaning that they are likely not suitable candidates for IP protection of generators in practice.

Other Generator Models. Recently, image generators such as DALL·E 2 [187] based on the Transformer architecture [230], and latent diffusion models [190] have been shown to also synthesize high-quality images. While OpenAI's DALL·E 2 generator is only accessible through a black-box API[6], model checkpoints for latent diffusion are publicly available as a white-box[7]. The utility of these checkpoints on FFHQ is comparable to that of the StyleGAN model checkpoints used in this chapter (e.g., latent diffusion reports a FID score of 4.98). DALL·E 2 and Latent Diffusion models also map from a latent space to images, but they accept auxiliary input such as text that controls the synthesis. While PTW is compatible with any image generator that maps latent codes to images, extending our work to other models requires developing a watermarking method to map to their parameters (see Section 6.4.2), which we leave to future work. Our work demonstrates for one state-of-the-art generator model architecture (StyleGAN-based generators with a FID score of 2.1) that they can be watermarked effectively using our method.

Summary of Deepfake Detection. Our research reveals that existing watermarks are not robust against an adversary with white-box access to the generator but can withstand a black-box adversary. This means that watermarking can be a viable solution for deterring deepfakes if the provider acts responsibly and the generator is provided through a black-box deployment. The provider's goal is to deter model misuse, which can be accomplished through several means. These include (1) monitoring and restricting queries, (2) relying on passive deepfake detection methods, or (3) implementing proactive methods such as watermarking. Monitoring lacks transparency and can deter usage of the model if the user does not trust the provider [218, 50]. Passive detection methods may not detect deepfakes as the quality of synthesized images improves or the adversary adapts to existing detectors [52]. Active methods such as watermarking enable a different type of deployment that (1) does not require query monitoring and (2) remains applicable to future, higher-quality generators. The provider and the user agree on a mutually trusted third party to deploy the generator, who does not tamper with the watermark nor monitor the queries. Our research suggests that such a black-box deployment represents a viable option in practice to prevent model misuse using existing watermarks.

Ethical Consideration. Deep image generators can have potential negative societal impacts when misused, for instance, when generating harmful deepfakes. Our contributions are intended to raise awareness about the limited trustworthiness of watermarking in potential future deployments of image generators rather than to undermine real systems. While the attacks presented in our work could be used to evade watermarking, thereby enabling misuse, we believe that sharing our attacks does not cause harm at this time since there are no known deployments of the presented watermarking methods. We aim to advance the development of watermarking methods that our attacks cannot break.

[6] https://openai.com/dall-e-2/

[7] https://github.com/CompVis/latent-diffusion

6.7 Conclusion

We propose Pivotal Tuning Watermarking (PTW), which is a scalable method for watermarking pre-trained image generators. PTW is three orders of magnitude faster than related work and enables watermarking large generators efficiently without any training data. We find that our watermark is not easily detectable without the secret watermarking key unless the attacker can access ≥ 100 labeled non-watermarked and watermarked images. Our watermark shows robustness against attackers who are limited in their availability of similar-quality models. However, we are the first to show that watermarking for image generators cannot withstand white-box attackers. Such an attacker can remove watermarks with almost no impact on image quality using less than 0.3% of the training data with our adaptive Reverse Pivotal Tuning (RPT) attack. Our results challenge that watermarking controls misuse when the parameters of a generator are provided, showing that open-source watermarking in the *no-box* setting remains a challenging problem.

Chapter 7

Leveraging Optimization for Adaptive Attacks on Image Watermarks

The previous chapter proposed a watermarking method for pre-trained image generators that could be optimized, which we refer to as a *learnable* watermarking method. A challenge when evaluating watermarking methods and their (adaptive) attacks is to determine whether an adaptive attack is optimal, i.e., it is the best possible attack. In this chapter, we solve this problem by defining an objective function and then approach adaptive attacks as an optimization problem. The core idea of our adaptive attacks is to replicate secret watermarking keys locally by creating surrogate keys that are differentiable and can be used to optimize the attack's parameters. We demonstrate for Stable Diffusion models that such an attacker can break all five surveyed watermarking methods at negligible degradation in image quality. Our findings highlight that the robustness of image watermarks must be tested with learnable, adaptive attackers in the future.

7.1 Motivation

Deepfakes are images synthesized with deep image generators that can undermine the authenticity of real images. While generated content can serve many beneficial purposes if used ethically, for example, in medical imaging [7] or education [176], they also have the potential to be *misused* and erode trust in digital media. Deepfakes have already been used in disinformation campaigns [14, 12] and social engineering attacks [161], highlighting the need for methods that control the misuse of deep image generators.

Watermarking offers a solution to controlling misuse by embedding hidden messages into all generated images that are later detectable using a secret watermarking key. Images detected

as deepfakes can be flagged by social media platforms or news agencies, mitigating potential harm [79]. Providers of large image generators such as Google have announced the deployment of their own watermarking methods [78] to enable the detection of deepfakes and promote the ethical use of their models, which was also declared as one of the main goals in the US government's "AI Executive Order" [65].

A core security property of watermarking is *robustness*, which states that an attacker can evade detection only by substantially degrading the image's quality. While several watermarking methods have been proposed for image generators [236, 261, 68], none of them are certifiably robust [11] and instead, robustness is tested empirically using a limited set of known attacks. Claimed security properties of previous watermarking methods have been broken by novel attacks [148], and no comprehensive method exists to validate robustness, which causes difficulty in trusting the deployment of watermarking in practice.

We propose testing the robustness of watermarking by defining robustness using objective function and approaching adaptive attacks as an optimization problem. Adaptive attacks are specific to the watermarking algorithm used by the defender but have no access to the secret watermarking key. Knowledge of the watermarking algorithm enables the attacker to consider a range of *surrogate* keys similar to the defender's key. This is also a challenge for optimization since the attacker only has imperfect information about the optimization problem.

Adaptive attackers had previously been shown to break the robustness of watermarking for image classifiers [148], but attacks had to be handcrafted against each watermarking method. Finding attack parameters through an optimization process can be challenging when the watermarking method is not easily optimizable, for instance, when it is not differentiable. Our attacks leverage optimization by approximating watermark verification through a differentiable process. Figure 7.1 shows that our adaptive attacker can prepare their attacks before the provider deploys their watermark. We show that adaptive, *learnable* attackers, whose parameters can be optimized efficiently, can evade watermark detection for 1 billion parameter Stable Diffusion models at a negligible degradation in image quality.

7.2 No-box Watermarking

Watermarking embeds a hidden signal into a medium, such as images, using a secret watermarking key that is later extractable using the same secret key. In the context of deep learning, watermarking can be characterized by the medium used by the defender to verify the presence of the hidden signal. White-box and black-box watermarking methods assume access to the model's parameters or query access via an API and have been used[1] primarily for Intellectual Property protection [227].

[1]Uchida et al. [227] study watermarking image classifiers. Our categorization is independent of the task.

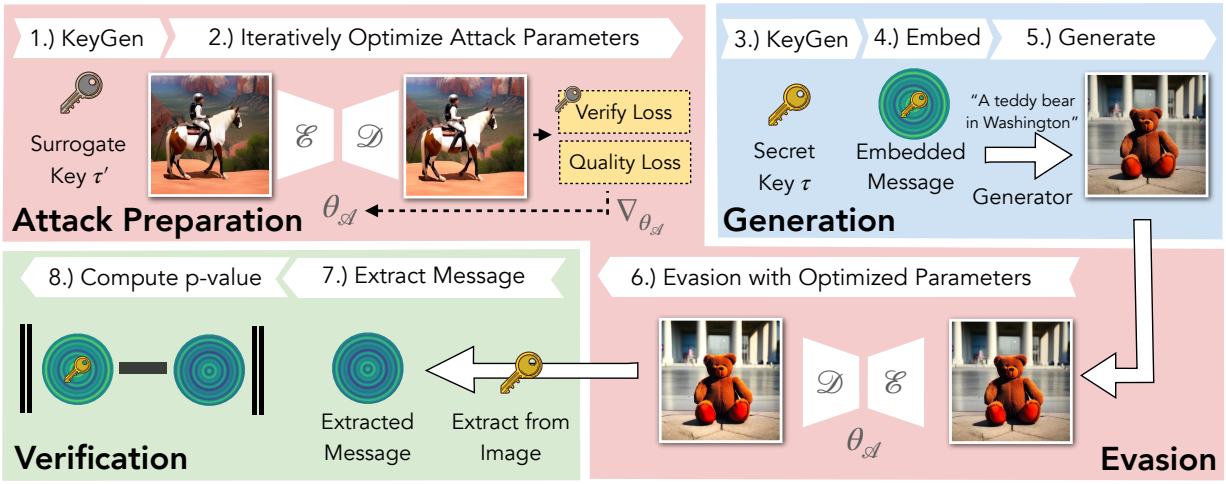


Figure 7.1: The attacker prepares their attack by generating a surrogate key and leveraging optimization to find optimal attack parameters $\theta_{\mathcal{A}}$ (illustrated here as an encoder $\mathcal{E}$ and decoder $\mathcal{D}$) for any message. Then, the attacker generates watermarked images and applies a modification using their optimized attack to evade detection. The attacker succeeds if the verification procedure cannot detect the watermark in high-quality images.

No-box watermarking [145] assumes a more restrictive setting where the defender only knows the generated content but does not know the query used to generate the image. This type of watermarking has been used to control misuse by having the ability to detect any image generated by the provided image generator [78]. Given a generator's parameters θ_G, a no-box watermarking method defines the following three procedures.

- $\tau \leftarrow \textsc{KeyGen}(\theta_G)$: A randomized function to generate a watermarking key τ.
- $\theta_G^* \leftarrow \textsc{Embed}(\theta_G, \tau, m)$: Given a generator θ_G, a watermarking key τ and a message m, return parameters θ_G^* of a *watermarked* generator[2] that only generates watermarked images.
- $p \leftarrow \textsc{Verify}(x, \tau, m)$: This function (i) extracts a message m' from x using τ and (ii) returns the p-value to reject the null hypothesis that m and m' match by random chance.

A watermarking method is a set of algorithms that specify (KeyGen, Embed, Verify). A watermark is a hidden signal in an image that can be mapped to a message m using a secret key τ. The key refers to secret random bits of information used in the randomized verification algorithm to detect a message. Adaptive attackers know the watermarking method but not the key message pair. [24] first studied adaptive attacks in the context of adversarial attacks.

[2]Embedding can alter the entire generation process, including adding pre- and post-processors.

In this chapter, we denote the similarity between two messages by their L_1-norm difference. We use more meaningful similarity measures when $\mathcal{M}$ allows it, such as the Bit-Error-Rate (BER) when the messages consist of bits. A watermark is *retained* in an image if the verification procedure returns $p < 0.01$, following [236]. Adi et al. [3] specify the requirements for trustworthy watermarking, and we focus on two properties: Effectiveness and robustness. Effectiveness states that a watermarked generator has a high image quality while retaining the watermark, and robustness means that a watermark is retained in an image unless the image's quality is substantially degraded. We refer to Chapter 6 for security games encoding effectiveness and robustness.

7.2.1 Watermarking for Image Generators

Several works propose no-box watermarking methods to prevent misuse for two types of image generators: Generative Adversarial Networks (GANs) [76] and Latent Diffusion Models (LDMs) [190]. We distinguish between *post-hoc* watermarking methods that apply an imperceptible modification to an image and *semantic* watermarks that modify the output distribution of an image generator and are truly "invisible" [236].

For post-hoc watermarking, traditional methods hide messages using the Discrete Wavelet Transform (DWT) and Discrete Wavelet Transform with Singular Value Decomposition (DWT-SVD) [41] and are currently used for Stable Diffusion[3]. RivaGAN [256] watermarks by training a deep neural network adversarially to stamp a pattern on an image. Yu et al. [250, 249] propose two methods that modify the generator's training procedure but require expensive re-training from scratch. Lukas and Kerschbaum [145] propose a watermarking method for GANs that can be embedded into a pre-trained generator. Zhao et al. [261] propose a general method to watermark diffusion models (WDM) that uses a method similar to Yu et al. [250], which tunes an autoencoder to stamp a watermark on all training data before also re-training the generator from scratch. Fernandez et al. [68] pre-train an autoencoder to encode hidden messages into the training data and embed the watermark by fine-tuning the decoder $\mathcal{D}$ component of the LDM.

Wen et al. [236] are the first to propose a semantic watermarking method for LDMs they call Tree-Rings Watermarks (TRW). The idea is to mark the initial noise x_T with a tree-ring-like pattern m in the frequency domain before generating an image. During detection, they leverage the reversibility of the diffusion process in LDM's, which allows mapping an image back to its original noise. The verification extracts a message m' by spectral analysis and tests whether the same tree-ring patterns m are retained in the frequency domain of the reconstructed noise.

Surveyed Watermarking Methods. In this chapter, we evaluate the robustness of five watermarking methods: TRW, WDM, DWT, DWT-SVD, and RivaGAN. DWT, DWT-SVD, and

[3] https://github.com/Stability-AI/stablediffusion

RivaGAN are default choices when using StabilityAI's Stable Diffusion repository, and WDM and TRW are two recently proposed methods for Stable Diffusion models. However, WDM requires re-training a Stable Diffusion model from scratch, which can require 150-1000 GPU days [49] and is not replicable with limited resources. For this reason, instead of using the autoencoder on the input data, we apply their autoencoder as a post-processor after generating images.

7.3 Threat Model

We consider a provider capable of training large image generators who make their generators accessible to many users via a black-box API, such as OpenAI with DALL·E 2. Users can query the generator by including a textual prompt that controls the content of the generated image. We consider an untrustworthy user who wants to misuse the provided generator without detection.

Provider's Capabilities and Goals *(Model Capabilities)* The provider fully controls image generation, including the ability to post-process generated images. *(Watermark Verification)* In a no-box setting, the defender must verify their watermark using a single generated image. The defender aims for an effective watermark that preserves generator quality while preventing the attacker from evading detection without significant image quality degradation.

Attacker's Capabilities. *(Model Capabilities)* The user has black-box query access to the provider's watermarked model and also has white-box access to less capable, open-source *surrogate* generators, such as Stable Diffusion on Huggingface. We assume the surrogate model's image quality is inferior to the provided model; otherwise, there would be no need to use the watermarked model. Our attacker does not require access to image generators from other providers, but, of course, such access may imply access to surrogate models as our attack does require. *(Data Access)* The attacker has unrestricted access to real-world image and caption data available online, such as LAION-5B [199]. *(Resources)* Computational resources are limited, preventing the attacker from training their own image generator from scratch. *(Queries)* The provider charges the attacker per image query, limiting the number of queries they can make. The attacker can generate images either unconditionally or with textual prompts. *(Adaptive)* The attacker knows the watermarking method but lacks access to the secret watermarking key τ and chosen message m.

Attacker's Goal. The attacker wants to use the provided, watermarked generator to synthesize images (i) without an extractable watermark that (ii) have a high quality. We measure quality using a perceptual similarity function $\mathcal{Q} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ between the generated, watermarked image and a perturbed image after the attacker evades watermark detection. We require that the defender verifies the presence of a watermark correctly with a p-value of at most $p < 0.01$ [236].

7.4 Conceptual Approach

In this Section, we define robustness as an objective function and show that we can optimize this function efficiently using stochastic gradient descent when the watermark algorithm is known (i.e., in adaptive attacks) for all five surveyed watermarking methods. Then, we propose *learnable* attacks and find optimal parameters through empirical loss minimization.

7.4.1 Robustness as an Objective Function

As described in Section 7.2, a watermarking method defines three procedures (`KeyGen`, `Embed`, `Verify`). The provider generates a secret watermarking key $\tau \leftarrow \textsc{KeyGen}(\theta_G)$ that allows them to watermark their generator so that all its generated images retain the watermark. To embed a watermark, the provider chooses a message (we sample a message $m \sim \mathcal{M}$ uniformly at random) and modifies their generator's parameters $\theta_G^* \leftarrow \textsc{Embed}(\theta_G, \tau, m)$. Any image generated by θ_G^* should retain the watermark. For any $x \leftarrow \textsc{Generate}(\theta_G^*)$ we call a watermark *effective* if (i) the watermark is retained, i.e., $\textsc{Verify}(x, \tau, m) < 0.01$ and (ii) the watermarked images have a high perceptual quality. The attacker generates images $x \leftarrow \textsc{Generate}(\theta_G^*)$ and applies an image-to-image transformation, $\mathcal{A} : \mathcal{D} \to \mathcal{D}$ with parameters $\theta_\mathcal{A}$ to evade watermark detection by perturbing $\hat{x} \leftarrow \mathcal{A}(x)$. Finally, the defender verifies the presence of their watermark in $\hat{x}$.

Let $W(\theta_G, \tau', m) = \textsc{Embed}(\theta_G, \tau', m)$ be the watermarked generator after embedding with key τ' and message m and $G_W = \textsc{Generate}(W(\theta_G, \tau', m))$ denotes the generation of an image using the watermarked generator. For any high-quality G_W, the attacker's objective becomes:

$$\max_{\theta_\mathcal{A}} \mathop{\mathbb{E}}_{\substack{\tau' \leftarrow \textsc{Keygen}(\theta_G) \\ m \in \mathcal{M}}} \left[\textsc{Verify}(\mathcal{A}(G_W), \tau', m) + \mathcal{Q}(\mathcal{A}(G_W), G_W) \right] \tag{7.1}$$

This objective seeks to maximize (i) the expectation of successful watermark evasion over all potential watermarking keys τ' and messages m (since the attacker does not know which key-message pair was chosen) and (ii) the perceptual similarity of the images before and after the attack. Note that in this chapter, we define image quality as the perceptual similarity to the watermarked image before the attack. There are two obstacles for an attacker to optimize this objective: (1) The attacker has imperfect information about the optimization problem and must substitute the defender's image generator with a less capable, open-source surrogate generator. When KeyGen depends on θ_G, then the distribution of keys differs, and the attack's effectiveness must transfer to keys generated using θ_G. (2) The optimization problem might be hard to approximate, even when perfect information is available, e.g., when the watermark verification procedure is not differentiable.

7.4.2 Making Watermarking Keys Differentiable

We overcome the two aforementioned limitations by (1) giving the attacker access to a similar (but less capable) surrogate generator $\hat{\theta}_G$, enabling them to generate surrogate watermarking keys, and (2) by creating a method GKeyGen($\hat{\theta}_G$) that creates a surrogate watermarking key θ_K through which we can backpropagate gradients. A simple but computationally expensive method of creating differentiable keys θ_D is using Algorithm 16 to train a watermark extraction neural network with parameters θ_D to predict the message m from an image.

Algorithm 16 GKeyGen: A Simple Method to Generate Differentiable Keys

Require: Surrogate generator $\hat{\theta}_G$, Watermarking method (`KeyGen`, `Embed`, `Verify`), N

1: $\tau \leftarrow$ KeyGen($\hat{\theta}_G$) ▷ *The non-differentiable surrogate key*
2: **for** $j \leftarrow 1$ **to** N **do**
3: $m \sim \mathcal{M}$ ▷ *Sample a random message*
4: $\hat{\theta}^*_G \leftarrow$ Embed($\hat{\theta}_G, \tau, m$) ▷ *Embed the watermark*
5: $x \leftarrow$ Generate($\hat{\theta}^*_G$)
6: $m' \leftarrow$ Extract($x; \theta_D$)
7: $g_{\theta_D} \leftarrow \nabla_{\theta_D} ||m - m'||_1$ ▷ *Compute gradients using distance between messages*
8: $\theta_D \leftarrow \theta_D - \text{Adam}(\theta_D, g_{\theta_D})$
9:
10: **return** θ_D ▷ *The differentiable surrogate key*

Algorithm 16 generates a surrogate key (line 1) to embed a watermark into the surrogate generator and use it to generate watermarked images (lines 3-5). The attacker extracts the message (line 6) and updates the parameters of the (differentiable) watermark decoder using an Adam optimizer [116]. The attacker subsequently uses the decoder's parameters θ_D as inputs to Verify. Our adaptive attacker must invoke Algorithm 16 only for the non-differentiable watermarks DCT and DCT-SVD [41]. The remaining three watermarking methods TRW [236], WDM [261] and RivaGAN [256] do not require invoking GKeyGen. In our work, we tune the parameters θ_D of a ResNet-50 decoder (see Appendix D.3 for details).

7.4.3 Leveraging Optimization against Watermarks

Equation (7.1) requires finding attack parameters $\theta_{\mathcal{A}}$ against any watermarking key $\tau' \leftarrow$ KeyGen(θ_G), which can be computationally expensive if the attacker has to invocate GKeyGen many times. We find empirically that generating many keys is unnecessary, and the attacker can find effective attacks using only a single surrogate watermarking key $\theta_D \leftarrow$ GKeyGen($\hat{\theta}_G$).

We propose two learnable attacks $\mathcal{A}_1, \mathcal{A}_2$ whose parameters $\theta_{\mathcal{A}_1}, \theta_{\mathcal{A}_2}$ can be optimized efficiently. The first attack called *Adversarial Noising*, finds adversarial examples given an image x using

Algorithm 17 Adversarial Noising

Require: surrogate $\hat{\theta}_G$, budget ϵ, image x

1: $\theta_{\mathcal{A}} \leftarrow 0$ ▷ *adversarial perturbation*
2: $\theta_D \leftarrow \text{GKeyGen}(\hat{\theta}_G)$
3: $m \leftarrow \text{Extract}(x; \theta_D)$
4: **for** $j \leftarrow 1$ **to** N **do**
5: $\quad m' \leftarrow \text{Extract}(x + \theta_{\mathcal{A}}, \theta_D)$
6: $\quad g_{\theta_{\mathcal{A}}} \leftarrow -\nabla_{\theta_{\mathcal{A}}} ||m - m'||_1$
7: $\quad \theta_{\mathcal{A}} \leftarrow P_\epsilon(\theta_{\mathcal{A}} - \text{Adam}(\theta_{\mathcal{A}}, g_{\theta_{\mathcal{A}}}))$
8: **return** $x + \theta_{\mathcal{A}}$

Algorithm 18 Adversarial Compression

Require: surrogate $\hat{\theta}_G$, strength α, image x

1: $\theta_{\mathcal{A}} \leftarrow [\theta_{\mathfrak{E}}, \theta_{\mathfrak{D}}]$ ▷ *Compressor parameters*
2: $\theta_D \leftarrow \text{GKeyGen}(\hat{\theta}_G)$ ▷ *surrogate key*
3: **for** $j \leftarrow 1$ **to** N **do**
4: $\quad m \sim \mathcal{M}$
5: $\quad \hat{\theta}^*_G \leftarrow \text{Embed}(\hat{\theta}_G, \theta_D, m)$
6: $\quad x \leftarrow \text{Generate}(\hat{\theta}^*_G)$
7: $\quad x' \leftarrow \mathfrak{D}(\mathfrak{E}(x; \theta_{\mathcal{A}}))$ ▷ *compression*
8: $\quad m' \leftarrow \text{Extract}(x', \theta_D)$
9: $\quad g_{\theta_{\mathcal{A}}} \leftarrow \nabla_\delta(\mathcal{L}_{\text{LPIPS}}(x', x) - \alpha||m - m'||_1)$
10: $\quad \theta_{\mathcal{A}} \leftarrow \theta_{\mathcal{A}} - \text{Adam}(\theta_{\mathcal{A}}, g_{\theta_{\mathcal{A}}})$
11: **return** $\mathfrak{D}(\mathfrak{E}(x; \theta_{\mathcal{A}}))$

the surrogate key as a reward model. The second attack, called *Adversarial Compression*, first fine-tunes the parameters of a pre-trained autoencoder in a preparation stage and uses the optimized parameters during an attack. The availability of a pre-trained autoencoder is a realistic assumption if the attacker has access to a surrogate Stable Diffusion generator, as the autoencoder is a detachable component of any Stable Diffusion generator. Access to a surrogate generator implies the availability of a pre-trained autoencoder at no additional cost to the attacker.

Adversarial Noising. Algorithm 17 shows the pseudocode of our adversarial noising attack. Given a surrogate generator $\hat{\theta}_G$, a budget $\epsilon \in \mathbb{R}^+$ for the maximum allowed noise perturbation, and a watermarked image x generated using the provider's watermarked model, the attacker wants to compute a perturbation within an ϵ-ball of the L_∞ norm that evades watermark detection. The attacker generates a local surrogate watermarking key (line 2) and extracts a message m from x (line 3). Then, the attacker computes the adversarial perturbation by maximizing the distance to the initially extracted message m and clips the perturbation into an ϵ-ball (line 7).

Adversarial Compression. Algorithm 18 shows the pseudocode of our adversarial compression attack. After generating a surrogate watermarking key (line 2), the attacker generates images containing a random message (lines 4-6) and uses their encoder-decoder pair to compress the images (line 7). The attacker iteratively updates their model's parameters by (i) minimizing a quality loss, which we set to the LPIPS metric [257], and (ii) maximizing the distance between the extracted and embedded messages (line 9). The output $\theta_{\mathcal{A}}$ of the optimization loop between lines 3 and 10 only needs to be run once, and the weights $\theta_{\mathcal{A}}$ can be re-used in subsequent attacks.

We highlight that the attacker optimizes an approximation of Equation (7.1) since they

only have a surrogate generator $\hat{\theta}_G$, but not the provider's generator θ_G. This may lead to a generalization gap of the attack at inference time. Even if an attacker can find optimal attack parameters $\theta_{\mathcal{A}}$ that optimizes Equation (7.1) using $\hat{\theta}_G$, the attacker cannot test whether their attack remains effective when the defender uses a different model θ_G to generate watermarking keys.

7.5 Experiments

Image Generators. We experiment with Stable Diffusion, an open-source, state-of-the-art latent diffusion model. The defender deploys a Stable Diffusion-v2.0 model[4] trained for 1.4m steps in total on a subset of LAION-5B [199]. The attacker uses a less capable Stable Diffusion-v1.1[5] checkpoint, trained for 431k steps in total on LAION-2B and LAION-HD. All experiments were conducted on NVIDIA A100 GPUs.

Images are generated using a DPM solver [144] with 20 inference steps and a default guidance scale of 7.5. We create three different watermarked generators for each surveyed watermarking method by randomly sampling a watermarking key $\tau \leftarrow \textsc{KeyGen}(\theta_G)$ and a message $m \sim \mathcal{M}$, used to embed a watermark. Appendix D.1 contains descriptions of the watermarking keys. All reported values represent the mean value over three independently generated secret keys.

Quantitative Analysis. Similar to Wen et al. [236], we report the True Positive Rate when the False Positive Rate is fixed to 1%, called the TPR@1%FPR. Appendix D.2 describes statistical tests used in the verification procedure of each watermarking method to derive p-values. We report the Fréchet Inception Distance (FID) [84], which measures the similarity between real and generated images. Additionally, we report the CLIP score [185] that measures the similarity of a prompt to an image. We generate 1k images to evaluate TPR@1%FPR and 5k images to evaluate FID and CLIP scores on the training dataset of MS-COCO-2017 [135].

7.5.1 Evaluating Robustness

Figure 7.2 shows the effectiveness of our attacks against all surveyed watermarking methods. We evaluate adaptive and non-adaptive attacks. Similar to [236], for the non-adaptive attacks, we use Blurring, JPEG Compression, Cropping, Gaussian noise, Jittering, Quantization, and Rotation but find these attacks to be ineffective at removing the watermark. Figure 7.2 highlights Pareto optimal attacks for pairs of (i) watermark detection accuracies and (ii) perceptual distances. We find that only adaptive attacks evade watermark detection while preserving image quality.

[4] https://huggingface.co/stabilityai/stable-diffusion-2-base
[5] https://huggingface.co/CompVis/stable-diffusion-v1-1

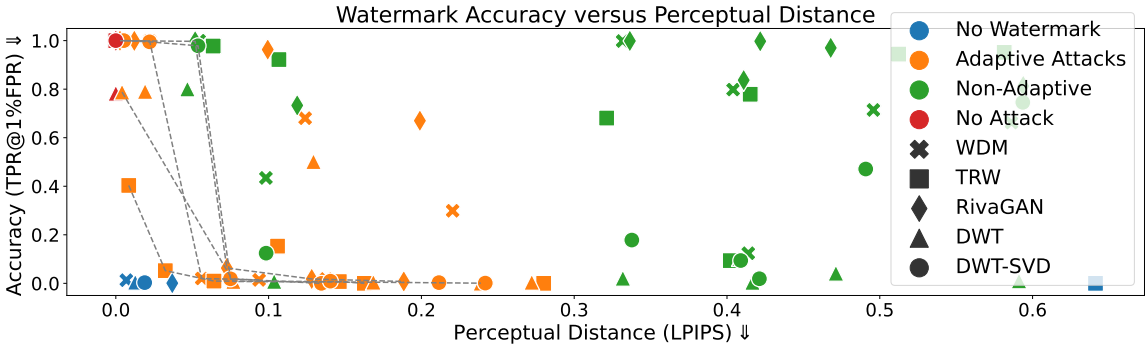


Figure 7.2: The effectiveness of our attacks against all watermarks. We highlight the Pareto front for each watermarking method by dashed lines and indicate adaptive/non-adaptive attacks by colors.

Table 7.1 summarizes the best attacks from Figure 7.2 when we set the lowest acceptable detection accuracy to 10%. When multiple attacks achieve a detection accuracy lower than 10%, we pick the attack with the lowest perceptual distance to the watermarked image. We observe that adversarial compression is an effective attack against all watermarking methods. TRW is also evaded by adversarial compression, but adversarial noising at $\epsilon = 2/255$ preserves a higher image quality.

	TRW	WDM	DWT	DWT-SVD	RivaGAN
Best Attack	Adv. Noising	Compression	Compression	Compression	Compression
Parameters	$\epsilon = 2/255$	$r = 1$	$r = 1$	$r = 1$	$r = 1$
LPIPS ⇓	3.2e-2	5.6e-2	7.7e-2	7.5e-2	7.3e-2
Accuracy ⇓	5.2%	2.0%	0.8%	1.9%	6.3%

Table 7.1: A summary of Pareto optimal attacks with detection accuracies less than 10%. We list the attack's name and parameters, the perceptual distance before and after evasion, and the accuracy (TPR@1%FPR). ϵ is the maximal perturbation in the L_∞ norm and r is the number of compressions.

7.5.2 Image Quality after an Attack

Figure 7.3 shows the perceptual quality after using our adaptive attacks. We show a cutout of the top left image patch with high contrasts on the bottom right to visualize noise artifacts potentially

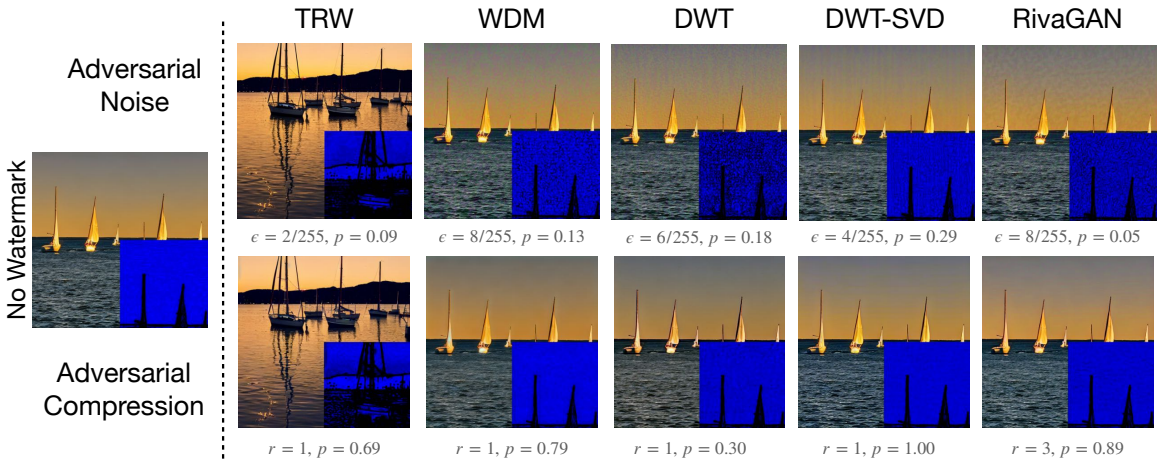


Figure 7.3: A visual analysis of two adaptive attacks. The left image shows the unwatermarked output, including a high-contrast cutout of the top left corner of the image to visualize noise artifacts. On the right are images after evasion with adversarial noising (top) and adversarial compression (bottom).

introduced by our attacks. We observe that, unlike adversarial noising, the compression attack introduces no new visible artifacts. Appendix D.4 displays more visualizations on the perceptual impact of our attacks on the image quality.

Table 7.2 shows the FID and CLIP scores of the watermarked images and the images after using our adaptive attacks. We calculate the quality using the best attack configuration from Figure 7.2 when the detection accuracy is less than 10%. Adversarial Noising is ineffective at removing WDM and RivaGAN for $\epsilon \leq {}^{10}/_{255}$, which we denote by N/A in Table 7.2.

7.5.3 Ablation Study

Figure 7.4 shows ablation studies for our adaptive attacks over the (i) maximum perturbation budget and (ii) the number of compressions applied during the attack. TRW and DWT-SVD are highly vulnerable to adversarial noising, whereas RivaGAN and WDM are substantially more robust to these types of attacks. We believe this is because keys generated by RivaGAN and WDM are sufficiently randomized, which makes our attack (that uses only a single surrogate key) less effective unless the surrogate key uses similar channels as the secret key to hide the watermark.

Figure 7.4 shows that adversarial compression *without* optimization of the parameters $\theta_{\mathcal{A}}$ is ineffective at evading watermark detection against all methods except DWT. Our adaptive optimization is necessary to find effective attacks. After optimization, adversarial compression evades detection from all watermarking methods with only one compression.

	TRW		WDM		DWT		DWT-SVD		RivaGAN	
	FID	CLIP	FID	CLIP	FID	CLIP	FID	CLIP	FID	CLIP
No WM	23.32	31.76	23.48	31.77	23.48	31.77	23.48	31.77	23.48	31.77
WM	24.19	31.78	23.43	31.72	23.16	32.11	23.10	32.15	22.96	31.84
A-Noise	23.67	32.15	N/A	N/A	23.55	32.46	22.89	32.50	N/A	N/A
A-Comp	24.36	31.87	23.27	32.01	23.16	32.17	23.06	31.92	23.25	31.86

Table 7.2: Quality metrics before and after watermark evasion. FID⇓ represents the Fréchet Inception Distance, and CLIP⇑ represents the CLIP score, computed on 5k images from MS-COCO-2017. N/A means the attack could not evade watermark detection, and consequently, we do not report quality measures.

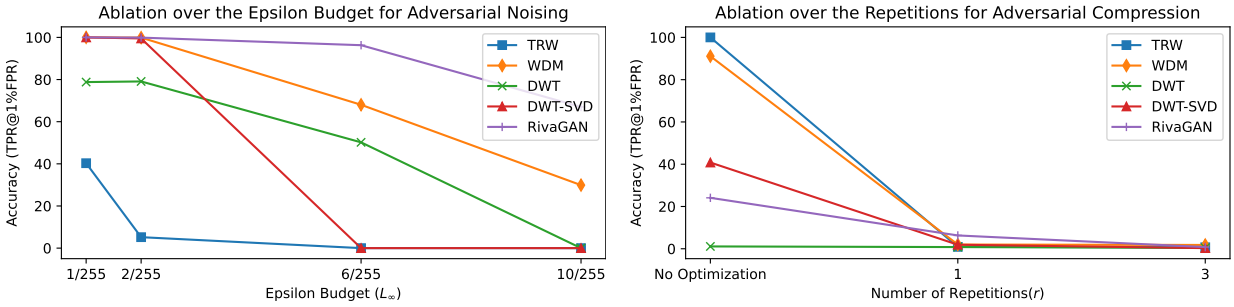


Figure 7.4: Ablation over (left) the maximum perturbation budget ϵ in L_∞ for adversarial noising and (right) the number of adversarial compressions against each watermarking method. "No Optimizations" means we did not optimize the parameters $\theta_{\mathcal{A}}$ of the attack.

7.6 Discussion & Related work

Attack Scalability. The presented findings clearly indicate that even with a less capable surrogate generator, an adaptive attacker can remove all surveyed watermarks with minimal quality degradation. Our attackers generate a single surrogate key and can evade watermark verification, which indicates a design flaw since the key seems to have little impact. If KeyGen were sufficiently randomized, breaking robustness should not be possible using a single key, even if the provided and surrogate generators are the same. An interesting question emerging from our study relates to the maximum difference between the watermarked and surrogate generators for the attacks to remain effective. We used a best-effort approach by using two public checkpoints with the largest reported quality differences: Stable Diffusion v1.1 and v2. More research is needed to study the impact on the effectiveness of our attacks (i) using different models, (ii) or limiting the attacker's knowledge of the method's public parameters. Our attacks evaluate the

best parameters suggested by the authors.

Types of Learnable Attacks. Measuring the robustness against different types of learnable attacks is crucial in assessing the trustworthiness of watermarking. We explored (i) Adversarial Examples, which rely solely on the surrogate key, and (ii) Adversarial Compression, which additionally requires the availability of a pre-trained autoencoder. We believe this requirement is satisfied in practice, given that (i) training autoencoders is computationally less demanding than training Stable Diffusion, and many pre-trained autoencoders have already been made publicly available [178]. Although autoencoders enhance an attacker's ability to modify images, our study did not extend to other learnable attacks such as inpainting [190] or more potent image editing methods [17] which could further enhance an attack's effectiveness.

Enhancing Robustness using Adaptive and Learnable Attacks. Relying on non-adaptive attacks for evaluating a watermark's robustness is inadequate as it underestimates the attacker's capabilities. To claim robustness, the defender could (i) provide a certification of robustness [11], or (ii) showcase empirically that their watermark withstands strong attacks. The issue is that we lack strong attackers. Although [148] demonstrated that adaptive attackers can break watermarks for image classifiers, their attacks were handcrafted and did not scale. Instead, we propose a better method of empirically testing robustness by proposing adaptive *learnable* attackers that require only the specification of a type of learnable attack, followed by an optimization procedure to find parameters that minimize an objective function. We believe that any watermarking method proposed in the future should evaluate robustness using our attacks and expect that future watermarking methods can enhance their robustness by incorporating our attacks.

Limitations. Our attacks are based on the availability of the watermarking algorithm and an open-source surrogate generator to replicate keys. While providers like Stable Diffusion openly share their models, and replicas of OpenAI's DALL·E models are publicly available [45], not all providers release information about their models. To the best of our knowledge, Google has not released their generators [193], but efforts to replicate are ongoing[6]. Providers like Midjourney, who keep their image generation algorithms undisclosed, prevent adaptive attackers altogether but may be vulnerable to these attacks by anyone to whom this information is released.

Outlook. An adaptive attacker can instantiate more effective versions of their attacks with knowledge of the watermarking method's algorithmic descriptions (KeyGen, Embed, Verify). Our attacks require no interaction with the provider. Robustness against adaptive attacks extends to robustness against non-adaptive attacks, which makes studying the former interesting. Adaptive attacks will remain a valuable tool for studying a watermarking method's robustness, even if the watermarking method is kept secret. Previous works did not consider such attacks, and we show that future watermarking methods must consider them if they claim robustness empirically.

[6] https://github.com/lucidrains/imagen-pytorch

7.6.1 Related Work

[110] propose attacks that use (indirect) access to the secret watermarking key via access to the provider's VERIFY method. Our attacks require no access to the provider's secret watermarking key, as our attacks optimize over any key message pair (see Equation (7.1)). These threats are related since both undermine robustness but are orthogonal due to different threat models. [175, 43] propose black-box watermarking methods that protect the Intellectual Property of Diffusion Models. We focus on no-box verifiable watermarking methods that control misuse. [145, 250, 249] propose watermarking methods but only evaluate GANs. We focus on watermarking methods for pre-trained Stable Diffusion models with much higher output diversity and image quality [49].

7.7 Conclusion

We propose testing the robustness of watermarking through adaptive, learnable attacks. Our empirical analysis shows that such attackers can evade watermark detection against all five surveyed image watermarks. Adversarial noising evades TRW [236] with $\epsilon = 2/255$ but needs to add visible noise to evade the remaining four watermarking methods. Adversarial compression evades all five watermarking methods using only a single compression. We encourage using these adaptive attacks to test the robustness of watermarking methods in the future.

Chapter 8

Conclusion and Future Directions

This thesis presented five projects on three threats that emerge when providing large and capable ML models to untrustworthy users. These threats are (i) data leakage, where the model leaks sensitive information about the training data; (ii) model stealing, where an attacker redistributes the provided model without authorization; and (iii) model misuse, where an attacker generates content to deceive or manipulate others. The research question in this thesis was the following.

How can security mechanisms be tested and designed to control unintended use of a provided model in the presence of untrustworthy users?

To provide answers to this question, I have empirically evaluated security mechanisms and tested their reliability in an adversarial setting. Below, I summarize the contributions of this thesis and outline future research directions.

8.1 Summary of Contributions

Chapter 3 shows that differential privacy alone cannot mitigate all privacy risks when fine-tuning language models. The problem is that sensitive data, such as Personally Identifiable Information (PII), can be duplicated within a group across many records, which would have to be accounted for when applying differentially private noise. However, it is difficult to know group sizes a priori, and larger group sizes have a detrimental impact on the model's utility. The main contributions are (i) novel attacks that can extract up to 10× more PII sequences than existing attacks, (ii) showing that sentence-level differential privacy reduces the risk of PII disclosure but still leaks about 3% of PII sequences, and (iii) a subtle connection between record-level membership inference and PII reconstruction.

I study the reliability of fingerprinting and watermarking methods for controlling unauthorized model redistribution in Chapters 4 and 5. The fingerprint presented in Chapter 5 proposes a new subclass of transferable adversarial examples, which I call *conferrable* examples, that extract inputs from a model via optimization and can be used to identify its surrogate models. I conduct a comprehensive set of experiments to demonstrate that the fingerprint is robust against model extraction attacks, which extends to robustness against other attacks, such as fine-tuning or pruning. However, designing defenses with robustness against unknown, adaptive attacks is challenging without knowledge of these attacks. I show that adaptive attacks can undermine the presented fingerprint's robustness. In Chapter 4, I describe work to systematically evaluate the robustness of known watermarking methods using handcrafted adaptive attacks. I propose taxonomies and game-based definitions for robustness. The main contributions of this chapter are taxonomies, new adaptive attacks, and guidelines on evaluating the robustness of watermarking methods via handcrafted attacks.

Chapters 6 and 7 study the reliability of watermarking to monitor and control the potential misuse of image generators by detecting watermarks in generated content. In Chapter 6, I propose Pivotal Tuning Watermarking, which is the first learnable watermarking method for pre-trained generators and is up to (i) three orders of magnitude faster than watermarking from scratch and (ii) does not require any training data. An advantage of learnable watermarking methods is that they can be adapted against known attacks. The evaluation shows that *learnable* watermarks can be robust against attackers restricted to black-box API access to the watermarked model. However, I propose attacks that undermine their robustness when the attacker has white-box access to the parameters of the watermarked model. In the white-box setting, the reverse Pivotal Tuning attack can remove any watermark at negligible degradation in image quality using only 200 non-watermarked images. These results support that open-weight models are unlikely to contain a robust watermark. In Chapter 7, I instantiate learnable, adaptive attacks that can be optimized to find the best possible parameters against any watermarking method given only its algorithmic description and access to a less capable surrogate model. My work shows that under such an assumption, one can instantiate adaptive attacks against five surveyed watermarking methods that break their robustness with imperceptible perturbations to the image. Notably, the presented attacks do not require access to the provider's watermark detector and can remove watermarks from a single watermarked image. I expect that future watermarking methods can incorporate these attacks to enhance their robustness.

8.2 Future Directions

The ongoing development of increasingly capable ML systems underscores the need for security measures to control their misuse and prevent data leakage. Such security mechanisms must be reliable; otherwise, they could provide a false sense of security or privacy. For example,

attackers have recently been shown to break ChatGPT's security mechanisms. They could, as a consequence, obtain potentially sensitive training data from ML systems such as ChatGPT [165] or manipulate the system to produce unintended or potentially harmful content [181, 234]. My research demonstrates that empirically tested security mechanisms are often unreliable because attackers can develop adaptive attacks. Provable defenses like differential privacy can still leak sensitive information from their dataset, as detecting private information correlated or duplicated across multiple records is challenging. A primary challenge in developing empirical defenses is the lack of knowledge about strong, adaptive attacks. To address this challenge, my work proposes treating robustness as an objective function that can be optimized to obtain learnable adaptive attackers that do not require handcrafting. An intriguing future direction involves integrating learnable defenses with learnable attacks and studying the equilibrium of this adversarial min-max optimization game. What restrictions on the adversaries' capabilities make it challenging for them to develop effective attacks, and how can one design security mechanisms that leverage such hardness assumptions? While out of the scope of my thesis, answering these research questions is essential to enhancing the reliability of security mechanisms.

References

[1] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *23rd ACM SIGSAC Conference on Computer and Communications Security, CCS 2016*, pages 308–318. ACM, 2016. doi: 10.1145/2976749.2978318.

[2] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[3] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1615–1631, 2018.

[4] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *CVPR workshops*, volume 1, page 38, 2019.

[5] William Aiken, Hyoungshick Kim, Simon Woo, and Jungwoo Ryoo. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *Computers & Security*, 106:102277, 2021.

[6] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)*, pages 54–59, 2019.

[7] Mohamed Akrout, Bálint Gyepesi, Péter Holló, Adrienn Poór, Blága Kincső, Stephen Solis, Katrina Cirone, Jeremy Kawahara, Dekker Slade, Latif Abid, et al. Diffusion-based data augmentation for skin disease classification: Impact across original medical datasets to fully synthetic images. *arXiv preprint arXiv:2301.04802*, 2023.

[8] Ali Al-Haj. Combined dwt-dct digital image watermarking. *Journal of computer science*, 3 (9):740–746, 2007.

[9] Jamal Kaid Mohammed Ali, Muayad Abdulhalim Ahmad Shamsan, Taha Ahmed Hezam, and Ahmed AQ Mohammed. Impact of chatgpt on learning motivation: teachers and students' voices. *Journal of English Studies in Arabia Felix*, 2(1):41–49, 2023.

[10] Apple. Differential privacy. Online at https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf. Retrieved 28 Nov 2022.

[11] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning*, pages 1450–1465. PMLR, 2022.

[12] Clark Barrett, Brad Boyd, Ellie Burzstein, Nicholas Carlini, Brad Chen, Jihye Choi, Amrita Roy Chowdhury, Mihai Christodorescu, Anupam Datta, Soheil Feizi, et al. Identifying and mitigating the security risks of generative ai. *arXiv preprint arXiv:2308.14840*, 2023.

[13] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[14] Dan Boneh, Andrew J Grotto, Patrick McDaniel, and Nicolas Papernot. How relevant is the turing test in the age of sophisbots? *IEEE Security & Privacy*, 17(6):64–71, 2019.

[15] Thomas Brewster. Huge bank fraud uses deep fake voice tech to steal millions. Forbes, Oct 2021. URL https://www.forbes.com/sites/thomasbrewster/2021/10/14/huge-bank-fraud-uses-deep-fake-voice-tech-to-steal-millions/?sh=37cf48e97559. Accessed: [insert date of access].

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[17] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.

[18] Hannah Brown, Katherine Lee, FatemehSadat Mireshghallah, R. Shokri, and Florian Tramèr. What does it mean for a language model to preserve privacy? *2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language

models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[20] Paweł Budzianowski and Ivan Vulić. Hello, it's GPT-2 - how can I help you? towards the use of pretrained language models for task-oriented dialogue systems. In Alexandra Birch, Andrew Finch, Hiroaki Hayashi, Ioannis Konstas, Thang Luong, Graham Neubig, Yusuke Oda, and Katsuhito Sudoh, editors, *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5602. URL https://aclanthology.org/D19-5602.

[21] Tu Bui, Ning Yu, and John Collomosse. Repmix: Representation mixing for robust attribution of synthesized images. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIV*, pages 146–163. Springer, 2022.

[22] Zhipeng Cai, Zuobin Xiong, Honghui Xu, Peng Wang, Wei Li, and Yi Pan. Generative adversarial networks: A survey toward private and secure applications. *ACM Computing Surveys (CSUR)*, 54(6):1–38, 2021.

[23] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 14–25, 2021.

[24] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[25] 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.

[26] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.

[27] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[28] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *43rd IEEE Symposium on Security and Privacy (S&P)*, pages 1897–1914. IEEE, 2022.

[29] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

[30] Zehra Cataltepe, Yaser S Abu-Mostafa, and Malik Magdon-Ismail. No free lunch for early stopping. *Neural computation*, 11(4):995–1009, 1999.

[31] Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. Neural legal judgment prediction in English. In *57th Annual Meeting of the Association for Computational Linguistics, ACL 2019*, pages 4317–4323. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1424.

[32] Han Chen, Yuezun Li, Dongdong Lin, Bin Li, and Junqiang Wu. Watching the big artifacts: Exposing deepfake videos via bi-granularity artifacts. *Pattern Recognition*, 135:109179, 2023.

[33] Huili Chen, Bita Darvish Rohani, and Farinaz Koushanfar. Deepmarks: A digital fingerprinting framework for deep neural networks. *arXiv preprint arXiv:1804.03648*, 2018.

[34] Huili Chen, Bita Darvish Rouhani, and Farinaz Koushanfar. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344*, 2019.

[35] Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2287–2295, 2019.

[36] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

[37] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR, 2021.

[38] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[39] Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 100(101):102, 2017.

[40] Michael W Coughlin, Joshua S Bloom, Guy Nir, Sarah Antier, Theophile Jegou Du Laz, Stefan van der Walt, Arien Crellin-Quick, Thomas Culino, Dmitry A Duev, Daniel A Goldstein, et al. A data science platform to enable time-domain astronomy. *The Astrophysical Journal Supplement Series*, 267(2):31, 2023.

[41] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital watermarking and steganography*. Morgan kaufmann, 2007.

[42] Dustin T Crystal, Nicholas G Cuccolo, Ahmed Ibrahim, Heather Furnas, and Samuel J Lin. Photographic and video deepfakes have arrived: how machine learning may influence plastic surgery. *Plastic and reconstructive surgery*, 145(4):1079–1086, 2020.

[43] Yingqian Cui, Jie Ren, Han Xu, Pengfei He, Hui Liu, Lichao Sun, and Jiliang Tang. Diffusionshield: A watermark for copyright protection against generative diffusion models. *arXiv preprint arXiv:2306.04642*, 2023.

[44] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[45] Boris Dayma, Suraj Patil, Pedro Cuenca, Khalid Saifullah, Tanishq Abraham, Phúc Le Khac, Luke Melas, and Ritobrata Ghosh. Dall e mini, 7 2021. URL https://github.com/borisdayma/dalle-mini.

[46] Adrienne De Ruiter. The distinct wrong of deepfakes. *Philosophy & Technology*, 34(4): 1311–1332, 2021.

[47] 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.

[48] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

[49] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[50] Jennifer Ding, Christopher Akiki, Yacine Jernite, Anne Lee Steele, and Temi Popo. Towards openness beyond open access: User journeys through 3 open ai collaboratives. *arXiv preprint arXiv:2301.08488*, 2023.

[51] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) dataset. *arXiv preprint arXiv:2006.07397*, 2020.

[52] Chengdong Dong, Ajay Kumar, and Eryun Liu. Think twice before detecting gan-generated fake images from their spectral domain imprints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7865–7874, 2022.

[53] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Ting Zhang, Weiming Zhang, Nenghai Yu, Dong Chen, Fang Wen, and Baining Guo. Protecting celebrities from deepfake with identity consistency transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9468–9478, 2022.

[54] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[55] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology - EUROCRYPT 2006*, pages 486–503. Springer Berlin Heidelberg, 2006.

[56] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[57] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[58] Sandy Engelhardt, Lalith Sharan, Matthias Karck, Raffaele De Simone, and Ivo Wolf. Cross-domain conditional generative adversarial networks for stereoscopic hyperrealism in surgical training. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part V*, pages 155–163. Springer, 2019.

[59] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[60] European Commission. Eu artificial intelligence act. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206, 2023. Accessed: 2023-12-14.

[61] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 april 2016 on the protection of natural persons with regard to the processing of personal

data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal*, L 110:1–88, 2016.

[62] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[63] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32, 2019.

[64] Federal Bureau of Investigation. Malicious actors manipulating photos and videos to create explicit content and sextortion schemes. Internet Crime Complaint Center (IC3), June 2023. URL https://www.ic3.gov/Media/Y2023/PSA230605.

[65] Federal Register. Safe, secure, and trustworthy development and use of artificial intelligence. https://www.federalregister.gov/documents/2023/11/01/2023-24283/safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence, 2023. Accessed 4 Nov. 2023.

[66] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

[67] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[68] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. *arXiv preprint arXiv:2303.15435*, 2023.

[69] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. Leveraging frequency analysis for deep fake image recognition. In *International conference on machine learning*, pages 3247–3258. PMLR, 2020.

[70] Galen Andrew, Steve Chien, and Nicolas Papernot. Tensorflow privacy. 2019 (accessed July 5, 2020). URL https://github.com/tensorflow/privacy.

[71] A Shaji George and AS Hovan George. A review of chatgpt ai's impact on several business sectors. *Partners Universal International Innovation Journal*, 1(1):9–23, 2023.

[72] Arindam Ghosh and Aritri Bir. Evaluating chatgpt's ability to solve higher-order questions on the competency-based medical education curriculum in medical biochemistry. *Cureus*, 15 (4), 2023.

[73] Sharath Girish, Saksham Suri, Sai Saketh Rambhatla, and Abhinav Shrivastava. Towards discovery and attribution of open-world gan generated images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14094–14103, 2021.

[74] Philippe Golle. Revisiting the uniqueness of simple demographics in the us population. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, pages 77–80, 2006.

[75] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[76] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[77] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[78] Sven Gowal and Pushmeet Kohli. Identifying ai-generated images with synthid, 2023. URL https://www.deepmind.com/blog/identifying-ai-generated-images-with-synthid. Accessed: 2023-09-23.

[79] Alexei Grinbaum and Laurynas Adomaitis. The ethical need for watermarks in machine-generated language. *arXiv preprint arXiv:2209.03118*, 2022.

[80] Mark Gurman. Samsung bans staff's ai use after spotting chatgpt data leak. *Bloomberg*, May 2023. URL https://www.bloomberg.com/news/articles/2023-05-02/samsung-bans-chatgpt-and-other-generative-ai-use-by-staff-after-leak. Accessed June 14th, 2023.

[81] Drew Harwell. Scarlett johansson on fake ai-generated sex videos:'nothing can stop someone from cutting and pasting my image'. *Washington Post*, 31:12, 2018.

[82] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[83] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[84] 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. *Advances in neural information processing systems*, 30, 2017.

[85] Emmie Hine and Luciano Floridi. New deepfake regulations in china are a tool for social stability, but at what cost? *Nature Machine Intelligence*, 4(7):608–610, 2022.

[86] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[87] Dorjan Hitaj and Luigi V Mancini. Have you stolen my model? evasion attacks against deep neural network watermarking techniques. *arXiv preprint arXiv:1809.00615*, 2018.

[88] Dorjan Hitaj, Briland Hitaj, and Luigi V Mancini. Evasion attacks against watermarking techniques found in mlaas systems. In *2019 Sixth International Conference on Software Defined Systems (SDS)*, pages 55–63. IEEE, 2019.

[89] Andrew Hoang, Antoine Bosselut, Asli Celikyilmaz, and Yejin Choi. Efficient adaptation of pretrained transformers for abstractive summarization. *arXiv preprint arXiv:1906.00138*, 2019.

[90] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

[91] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength natural language processing in Python. 2020. doi: 10.5281/zenodo.1212303.

[92] 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. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[93] Shu Hu, Yuezun Li, and Siwei Lyu. Exposing gan-generated faces using inconsistent corneal specular highlights. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2500–2504. IEEE, 2021.

[94] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[95] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. Are large pre-trained language models leaking your personal information? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.148. URL https://aclanthology.org/2022.findings-emnlp.148.

[96] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[97] Thomas Humphries, Simon Oya, Lindsey Tulloch, Matthew Rafuse, Ian Goldberg, Urs Hengartner, and Florian Kerschbaum. Investigating membership inference attacks under data dependencies. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 473–488. IEEE, 2023.

[98] Thomas Humphries, Simon Oya, Lindsey Tulloch, Matthew Rafuse, Ian Goldberg, Urs Hengartner, and Florian Kerschbaum. Investigating membership inference attacks under data dependencies. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 473–488. IEEE, 2023.

[99] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

[100] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[101] Huseyin A Inan, Osman Ramadan, Lukas Wutschitz, Daniel Jones, Victor Rühle, James Withers, and Robert Sim. Privacy analysis in language models via training data leakage report. *ArXiv, abs/2101.05405*, 2021.

[102] Daphne Ippolito, Florian Tramer, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher Choquette Choo, and Nicholas Carlini. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In C. Maria Keet, Hung-Yi Lee, and Sina Zarrieß, editors, *Proceedings of the 16th International Natural Language Generation Conference*, pages 28–53, Prague, Czechia, September 2023.

Association for Computational Linguistics. doi: 10.18653/v1/2023.inlg-main.3. URL https://aclanthology.org/2023.inlg-main.3.

[103] Abhyuday Jagannatha, Bhanu Pratap Singh Rawat, and Hong Yu. Membership inference attack susceptibility of clinical language models. *arXiv preprint arXiv:2104.08305*, 2021.

[104] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *29th USENIX security symposium (USENIX Security 20)*, pages 1345–1362, 2020.

[105] Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=7bJizxLKrR.

[106] Mohd Javaid, Abid Haleem, and Ravi Pratap Singh. Chatgpt for healthcare services: An emerging stage for an innovative perspective. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 3(1):100105, 2023.

[107] Bargav Jayaraman, Esha Ghosh, Melissa Chase, Sambuddha Roy, Wei Dai, and David Evans. Combing for credentials: Active pattern extraction from smart reply. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 41–41. IEEE Computer Society, 2023.

[108] Yonghyun Jeong, Doyeon Kim, Seungjai Min, Seongho Joe, Youngjune Gwon, and Jongwon Choi. Bihpf: bilateral high-pass filters for robust deepfake detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 48–57, 2022.

[109] Hengrui Jia, Christopher A Choquette-Choo, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. *30th {USENIX} Security Symposium ({USENIX} Security 21) (to appear)*, 2021.

[110] Zhengyuan Jiang, Jinghuai Zhang, and Neil Zhenqiang Gong. Evading watermark based detection of ai-generated content. *arXiv preprint arXiv:2305.03807*, 2023.

[111] Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning*, pages 10697–10707. PMLR, 2022.

[112] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[113] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[114] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[115] Joe Katz. A new balance between health care privacy and artificial intelligence. The Regulatory Review, 2023. URL https://www.theregreview.org/2023/06/01/katz-a-new-balance-between-health-care-privacy-and-artificial-intelligence/. Accessed: December 13, 2023.

[116] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[117] Bryan Klimt and Yiming Yang. Introducing the enron corpus. In *CEAS*, 2004.

[118] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 491–507. Springer, 2020.

[119] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[120] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. *8th International Conference on Learning Representations, ICLR*, 2020.

[121] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[122] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[123] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[124] 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. *arXiv preprint arXiv:1811.00982*, 2018.

[125] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[126] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL https://aclanthology.org/N16-1030.

[127] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.

[128] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32:9233–9244, 2020.

[129] Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, pages 3637–3647, 2023.

[130] Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron Wallace. Does BERT pretrained on clinical notes reveal sensitive data? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 946–959, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.73. URL https://aclanthology.org/2021.naacl-main.73.

[131] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and Baining Guo. Face x-ray for more general face forgery detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5001–5010, 2020.

[132] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=bVuP3ltATMz.

[133] Yue Li, Hongxia Wang, and Mauro Barni. A survey of deep neural network watermarking techniques. *Neurocomputing*, 461:171–193, 2021.

[134] Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryetZ20ctX.

[135] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[136] Wei-An Lin, Yogesh Balaji, Pouya Samangouei, and Rama Chellappa. Invert and defend: Model-based approximate inversion of generative adversarial networks for secure inference. *arXiv preprint arXiv:1911.10291*, 2019.

[137] Gaoyang Liu, Chen Wang, Kai Peng, Haojun Huang, Yutong Li, and Wenqing Cheng. Socinf: Membership inference attacks on social media health data with machine learning. *IEEE Transactions on Computational Social Systems*, 6(5):907–921, 2019.

[138] Jian Liu, Rui Zhang, Sebastian Szyller, Kui Ren, and N Asokan. False claims against model ownership resolution. *arXiv preprint arXiv:2304.06607*, 2023.

[139] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018.

[140] Xuankai Liu, Fengting Li, Bihan Wen, and Qi Li. Removing backdoor-based watermarks in neural networks with limited data. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 10149–10156. IEEE, 2021.

[141] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Sys6GJqxl.

[142] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[143] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[144] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[145] Nils Lukas and Florian Kerschbaum. Ptw: Pivotal tuning watermarking for pre-trained image generators. *32nd USENIX Security Symposium*, 2023.

[146] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. In *The Ninth International Conference on Learning Representations (ICLR)*, 2021.

[147] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. *International Conference on Learning Representations*, 2021.

[148] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. Sok: How robust is image classification deep neural network watermarking? In *43rd IEEE Symposium on Security and Privacy (SP)*, pages 787–804. IEEE, 2022.

[149] Nils Lukas, Abdulrahman Diaa, Lucas Fenaux, and Florian Kerschbaum. Leveraging optimization for adaptive attacks on image watermarks. *Under Submission*, 2023.

[150] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Analyzing leakage of personally identifiable information in language models. *44th IEEE Symposium on Security and Privacy (SP)*, 2023.

[151] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[152] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints? In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 506–511. IEEE, 2019.

[153] Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pages 83–92. IEEE, 2019.

[154] Shiona McCallum. Chatgpt accessible again in italy, 2023. URL https://www.bbc.com/news/technology-65431914.

[155] R. Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. How much do language models copy from their training data? evaluating linguistic novelty in text generation using RAVEN. *Transactions of the Association for Computational Linguistics*, 11:652–670, 2023. doi: 10.1162/tacl_a_00567. URL https://aclanthology.org/2023.tacl-1.38.

[156] Brendan McMahan and Abhradeep Thakurta. Federated learning with formal differential privacy guarantees. Technical report, Google, 2022. URL https://ai.googleblog.com/2022/02/federated-learning-with-formal.html.

[157] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842, 2019.

[158] Jaron Mink, Licheng Luo, Natã M Barbosa, Olivia Figueira, Yang Wang, and Gang Wang. Deepphish: Understanding user trust towards artificially generated profiles in online social networks. In *Proc. of USENIX Security*, 2022.

[159] Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. Quantifying privacy risks of masked language models using membership inference attacks. *arXiv preprint arXiv:2203.03929*, 2022.

[160] Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. Memorization in nlp fine-tuning methods. *arXiv preprint arXiv:2205.12506*, 2022.

[161] Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)*, 54(1):1–41, 2021.

[162] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[163] Moin Nadeem, Anna Bethke, and Siva Reddy. StereoSet: Measuring stereotypical bias in pretrained language models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.416. URL https://aclanthology.org/2021.acl-long.416.

[164] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[165] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

[166] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[167] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

[168] Ding Sheng Ong, Chee Seng Chan, Kam Woh Ng, Lixin Fan, and Qiang Yang. Protecting intellectual property of generative adversarial networks from ambiguity attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3630–3639, 2021.

[169] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[170] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[171] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[172] Rahil Parikh, Christophe Dupuy, and Rahul Gupta. Canary extraction in natural language understanding models. *arXiv preprint arXiv:2203.13920*, 2022.

[173] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[174] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021.

[175] Sen Peng, Yufei Chen, Cong Wang, and Xiaohua Jia. Protecting the intellectual property of diffusion models by the watermark diffusion process. *arXiv preprint arXiv:2306.03436*, 2023.

[176] Renana Peres, Martin Schreier, David Schweidel, and Alina Sorescu. On chatgpt and beyond: How generative artificial intelligence may affect research, teaching, and practice. *International Journal of Research in Marketing*, 2023.

[177] Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. The text anonymization benchmark (tab): A dedicated corpus and evaluation framework for text anonymization. *Computational Linguistics*, 48(4):1053–1101, 2022.

[178] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[179] Gil Press. Cleaning big data: Most time-consuming, least enjoyable data science task, survey says. 2016 (accessed July 5, 2020). URL https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=7cfe7eae6f63.

[180] Fabi Prezja, Juha Paloneva, Ilkka Pölönen, Esko Niinimäki, and Sami Äyrämö. Deepfake knee osteoarthritis x-rays from generative adversarial neural networks deceive medical experts and offer augmentation potential to automatic classification. *Scientific Reports*, 12 (1):18573, 2022.

[181] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*, 2023.

[182] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII*, pages 86–103. Springer, 2020.

[183] 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. *Image*, 2:T2.

[184] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[185] 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*, pages 8748–8763. PMLR, 2021.

[186] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[187] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[188] Luc Rocher, Julien M Hendrickx, and Yves-Alexandre De Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, 10(1):1–9, 2019.

[189] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)*, 42(1):1–13, 2022.

[190] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[191] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1–11, 2019.

[192] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750*, 2018.

[193] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[194] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Artificial Intelligence and Statistics*, pages 412–419. PMLR, 2007.

[195] Ahmed Salem, Giovanni Cherubin, David Evans, Boris Köpf, Andrew Paverd, Anshuman Suri, Shruti Tople, and Santiago Zanella-Béguelin. Sok: Let the privacy games begin! a unified treatment of data inference privacy in machine learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 327–345. IEEE, 2023.

[196] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Computer Science Laboratory, SRI International, 1998. URL http://www.csl.sri.com/papers/sritr-98-04/.

[197] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[198] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

[199] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[200] R Sennrich, B Haddow, and A Birch. Neural machine translation of rare words with subword units. in: Proceedings of the 54th annual meeting of the association for computational linguistics. Association for Computational Linguistics Berlin, Germany, 2016.

[201] Zeyang Sha, Zheng Li, Ning Yu, and Yang Zhang. De-fake: Detection and attribution of fake images generated by text-to-image generation models. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 3418–3432, 2023.

[202] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of backdoor-based watermarking in deep neural networks. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, pages 177–188, 2021.

[203] William Shaw. Jpmorgan clamps down on staff's use of ai-powered chatgpt bot. Bloomberg, 2023. URL https://www.bloomberg.com/news/articles/2023-02-22/jpmorgan-clamps-down-on-staff-s-use-of-ai-powered-chatgpt-bot?srnd=premium. Accessed: December 13, 2023.

[204] Weiyan Shi, Aiqi Cui, Evan Li, Ruoxi Jia, and Zhou Yu. Selective differential privacy for language modeling. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2848–2859, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.205. URL https://aclanthology.org/2022.naacl-main.205.

[205] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[206] Rashish Shrivastava. This congressional candidate is using ai to have conversations with thousands of voters. *Forbes*, Dec 2023. URL https://www.forbes.com/sites/

rashishrivastava/2023/12/12/this-congressional-candidate-is-using-ai-to-have-conversations-with-thousands-of-voters/?sh=613f3cd810f7.

[207] Jessica Silbey and Woodrow Hartzog. The upside of deep fakes. *Md. L. Rev.*, 78:960, 2018.

[208] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[209] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[210] Vera Sorin, Yiftach Barash, Eli Konen, and Eyal Klang. Creating artificial images for radiology applications using generative adversarial networks (gans)–a systematic review. *Academic radiology*, 27(8):1175–1185, 2020.

[211] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[212] Pierre Stock, Igor Shilov, Ilya Mironov, and Alexandre Sablayrolles. Defending against reconstruction attacks with renyi differential privacy. *arXiv preprint arXiv:2202.07623*, 2022.

[213] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2014.

[214] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[215] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[216] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4417–4425, 2021.

[217] Sebastian Szyller, Vasisht Duddu, Tommi Gröndahl, and N Asokan. Good artists copy, great artists steal: Model extraction attacks against image translation generative adversarial networks. *arXiv preprint arXiv:2104.12623*, 2021.

[218] Alex Tamkin, Miles Brundage, Jack Clark, and Deep Ganguli. Understanding the capabilities, limitations, and societal impact of large language models. *arXiv preprint arXiv:2102.02503*, 2021.

[219] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2020.

[220] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314. ACM, 2018.

[221] Aaron Tilley. Apple restricts employee use of chatgpt, joining other companies wary of leaks. *The Wall Street Journal*, 2023. URL https://www.wsj.com/articles/apple-restricts-use-of-chatgpt-joining-other-companies-wary-of-leaks-d44d7d34.

[222] Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=u3vEuRr08MT.

[223] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[224] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[225] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[226] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkZvSe-RZ.

[227] 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.

[228] Thomas Vakili and Hercules Dalianis. Are clinical bert models privacy preserving? the difficulty of extracting patient-condition associations. In *HUMAN@ AAAI Fall Symposium*, 2021.

[229] Thomas Vakili, Anastasios Lamproudis, Aron Henriksson, and Hercules Dalianis. Downstream task performance of bert models pre-trained using automatically de-identified clinical data. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 4245–4252, 2022.

[230] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[231] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[232] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[233] Tianhao Wang and Florian Kerschbaum. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2622–2626. IEEE, 2019.

[234] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

[235] Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 2011.

[236] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[237] Mika Westerlund. The emergence of deepfake technology: A review. *Technology innovation management review*, 9(11), 2019.

[238] Junde Wu and Rao Fu. Universal, transferable and targeted adversarial attacks. *arXiv preprint arXiv:1908.11332*, 2019.

[239] Xi Wu, Zhen Xie, YuTao Gao, and Yu Xiao. Sstnet: Detecting manipulated faces through spatial, steganalysis and temporal features. In *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2952–2956. IEEE, 2020.

[240] Lukas Wutschitz, Huseyin A. Inan, and Andre Manoel. dp-transformers: Training transformer models with differential privacy. https://www.microsoft.com/en-us/research/project/dp-transformers, August 2022.

[241] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings of the 2018 Network and Distributed System Security Symposium*. NDSS, 2018.

[242] Tianyun Yang, Juan Cao, Qiang Sheng, Lei Li, Jiaqi Ji, Xirong Li, and Sheng Tang. Learning to disentangle gan fingerprint for fake image attribution. *arXiv preprint arXiv:2106.08749*, 2021.

[243] Tianyun Yang, Ziyao Huang, Juan Cao, Lei Li, and Xirong Li. Deepfake network architecture attribution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4662–4670, 2022.

[244] Samuel Yeom, Irene Giacomelli, Alan Menaged, Matt Fredrikson, and Somesh Jha. Overfitting, robustness, and malicious algorithms: A study of potential causes of privacy risk in machine learning. *Journal of Computer Security*, 28(1):35–70, 2020.

[245] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational intelligence magazine*, 13(3):55–75, 2018.

[246] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. Opacus: User-friendly differential privacy library in pytorch. *arXiv preprint arXiv:2109.12298*, 2021.

[247] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Q42f0dfjECO.

[248] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7556–7566, 2019.

[249] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14448–14457, 2021.

[250] Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry S. Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=sOK-zS6WHB.

[251] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[252] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 363–375, 2020.

[253] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 39–49, 2017.

[254] Bowen Zhang, Shuyang Gu, Bo Zhang, Jianmin Bao, Dong Chen, Fang Wen, Yong Wang, and Baining Guo. Styleswin: Transformer-based gan for high-resolution image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11304–11314, 2022.

[255] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018.

[256] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019.

[257] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[258] Ruisi Zhang, Seira Hidano, and Farinaz Koushanfar. Text revealer: Private text reconstruction via model inversion attacks against transformers. *arXiv preprint arXiv:2209.10505*, 2022.

[259] Ziqi Zhang, Chao Yan, and Bradley A Malin. Membership inference attacks against synthetic health data. *Journal of biomedical informatics*, 125:103977, 2022.

[260] Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Provably confidential language modelling. In *North American Chapter of the Association for Computational Linguistics*, 2022. URL https://api.semanticscholar.org/CorpusID:248512871.

[261] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*, 2023.

[262] Xin Zhong, Pei-Chi Huang, Spyridon Mastorakis, and Frank Y Shih. An automated and robust image watermarking scheme based on deep neural networks. *IEEE Transactions on Multimedia*, 23:1951–1961, 2020.

[263] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020.

[264] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *6th International Conference on Learning Representations, ICLR*, 2018.

APPENDICES

Appendix A

Deep Neural Network Fingerprinting by Conferrable Adversarial Examples

A.1 Supplementary Material

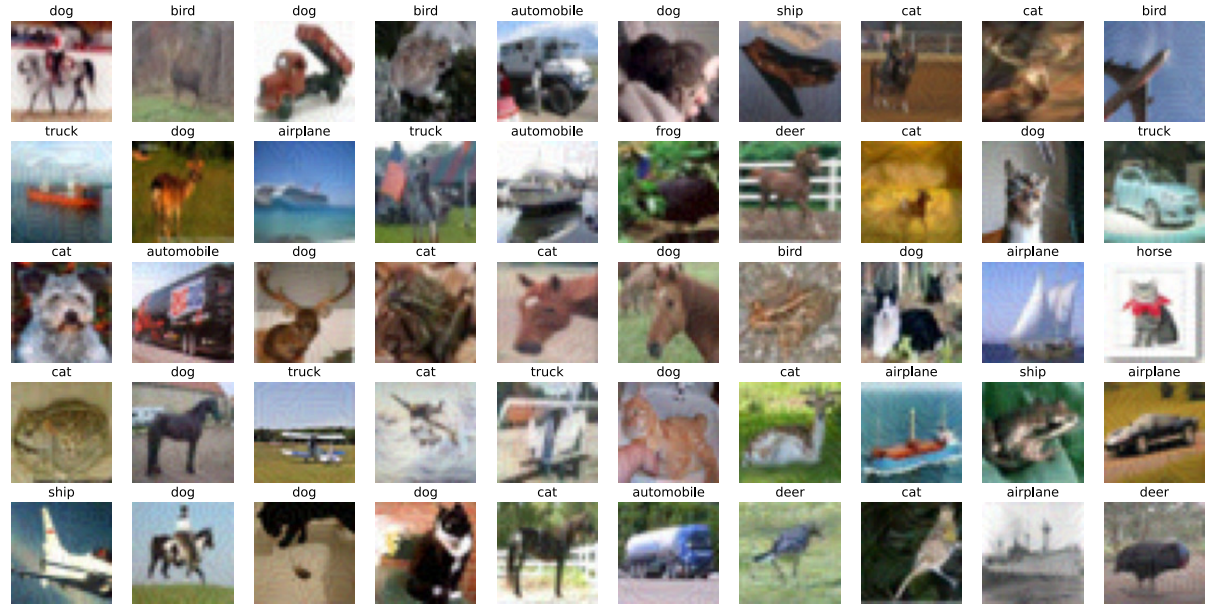
Setup. We train all models on a server running Ubuntu 18.04 in 64-bit mode using four Tesla P100 GPUs and 128 cores of an IBM POWER8 CPU (2.4GHz) with up to 1TB of accessible RAM. The machine learning is implemented in Keras using the Tensorflow v2.2 backend, and datasets are loaded using Tensorflow Dataset[1]. We re-use implementations of existing adversarial attacks from the Adversarial Robustness Toolbox v1.3.1 [167]. DP-SGD [2] is implemented through Tensorflow Privacy [70]. All remaining attacks and IPGuard [23] are re-implemented from scratch.

For the generated adversarial examples using attacks like FGM, PGD, and CW-L_∞, we use the following parametrization. We limit the maximum number of iterations in PGD to 10 and use a step size of 0.01. For FGM, we use a step size of ϵ, and for CW-L_∞ we limit the number of iterations to 50, use a confidence parameter $k = 0.5$, a learning rate of 0.01 and a step-size of 0.01. Fingerprints generated by CEM for CIFAR-10 can be seen in Fig. A.1a.

Analyzing Conferrable Examples. We further examine our fingerprint to find an explanation for the large mean CAE accuracy in all reference models of more than 50%, even though the baseline for random guessing is just 10%. For this, we plot the confusion matrix for our fingerprint at $\epsilon = 0.025$ in Fig. A.2.

The confusion matrix shows that all classes are represented in the fingerprint, and the target class distribution is balanced. We notice a symmetry in the confusion matrix along the diagonal line. A source-target class pair appears more often in our fingerprint if the classes are more similar

[1] https://www.tensorflow.org/datasets



(a)

(b)

Figure A.1: Figure A.1a shows conferrable examples generated with CEM on CIFAR-10 with $\epsilon \approx 6/255$. Figure A.1b shows examples generated with CEM on ImageNet32 with $\epsilon \approx 26/255$.

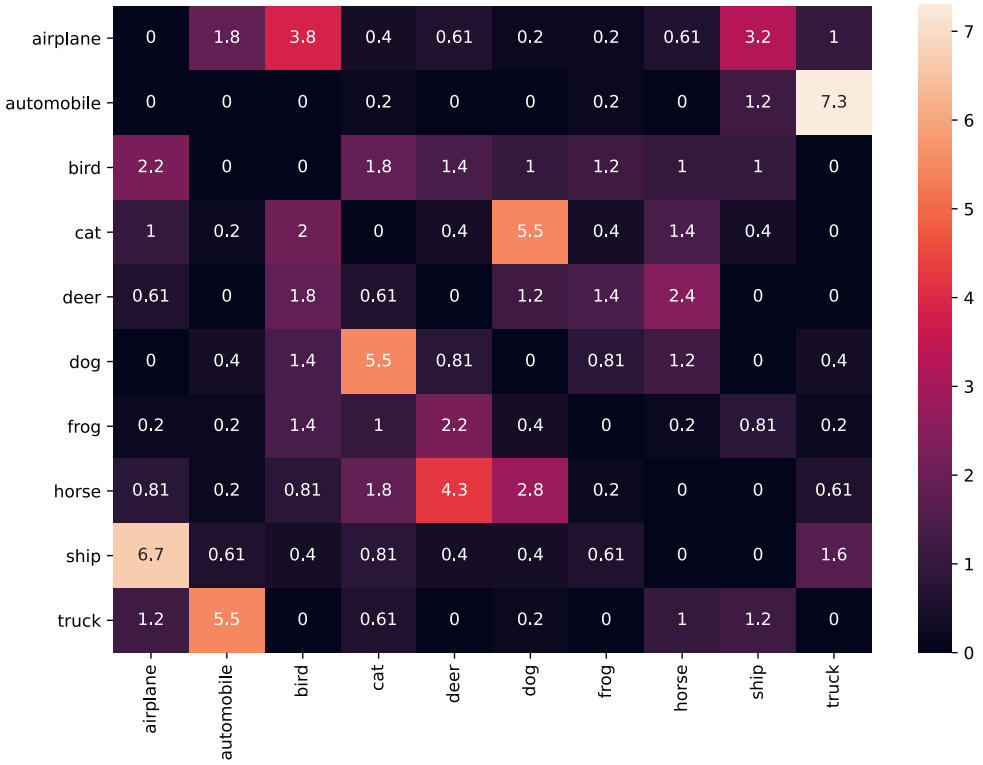


Figure A.2: Confusion matrix for a fingerprint on CIFAR-10 models at $\epsilon \approx 6/255$ with the initial label on the vertical axis and the target label on the horizontal axis. Depicted values are normalized and represent percentages.

to each other, e.g. classes 'dog' and 'cat' or 'automobile' and 'truck'. We hypothesize that this partially explains why reference models have large CAE accuracies of over 50%, even though the baseline for random guessing would be at 10%. A reference model is more likely to misclassify the class 'cat' as the class 'dog' than any other class, which moves the baseline closer to 50% for many cases. The similarity of the reference model to the source model is another important aspect of explaining the high CAE accuracy in the reference model. Our experiments show that reference models trained on the same dataset are also more likely to share the same adversarial vulnerabilities.

A.2 Experiments on ImageNet32

We select the following 100 classes from ImageNet32 [38].

`kit fox`, `Persian cat`, `gazelle`, `porcupine`, `sea lion`, `killer whale`, `African elephant`, `jaguar`, `otterhound`, `hyena`, `sorrel`, `dalmatian`, `fox squirrel`, `tiger`, `zebra`, `ram`, `orangutan`, `squirrel monkey`, `komondor`, `guinea pig`, `golden retriever`, `macaque`, `pug`, `water buffalo`, `American black bear`, `giant panda`, `armadillo`, `gibbon`, `German shepherd`, `koala`, `umbrella`, `soccer ball`, `starfish`, `grand piano`, `laptop`, `strawberry`, `airliner`, `balloon`, `space shuttle`, `aircraft carrier`, `tank`, `missile`, `mountain bike`, `steam locomotive`, `cab`, `snowplow`, `bookcase`, `toilet seat`, `pool table`, `orange`, `lemon`, `violin`, `sax`, `volcano`, `coral reef`, `lakeside`, `hammer`, `vulture`, `hummingbird`, `flamingo`, `great white shark`, `hammerhead`, `stingray`, `barracouta`, `goldfish`, `American chameleon`, `green snake`, `European fire salamander`, `loudspeaker`, `microphone`, `digital clock`, `sunglass`, `combination lock`, `nail`, `altar`, `mountain tent`, `scoreboard`, `mashed potato`, `head cabbage`, `cucumber`, `plate`, `necklace`, `sandal`, `ski mask`, `teddy`, `golf ball`, `red wine`, `sunscreen`, `beer glass`, `cup`, `traffic light`, `lipstick`, `hotdog`, `toilet tissue`, `cassette`, `lotion`, `barrel`, `basketball`, `barbell`, `pole`

We train a ResNet20 source model and locally retrain $c_1 = 14$ surrogate and $c_2 = 15$ reference models. In contrast to the experiments on CIFAR-10, we allow the defender access to various model architectures, such as ResNet56, Densenet, VGG19, and MobilenetV2. We want to verify if generating conferrable adversarial examples is feasible when the defender trains a set of models with a larger variety in model architecture. In practice, it may be the case that the defender locally searches for the best model architecture and, in the process, obtains multiple surrogate and reference models with various model architectures.

Table A.1 and A.2 show the test accuracies and CAEAcc values of ImageNet32 surrogate and reference models. Values in the brackets denote the lowest and highest values measured. Our results for ImageNet32 are comparable to those obtained with models trained on CIFAR-10. We

even measure a lower CAE accuracy in reference models than we did in reference models trained on CIFAR-10. Note that the experiments conducted on ImageNet32 are results reported for $\epsilon = 0.15$, and the results for CIFAR-10 were generated with a perturbation threshold of $\epsilon = 0.025$.

Table A.1: Surrogate model accuracies for ImageNet32.

	Fine-Tuning				
Metric	**Source**	**FTLL**	**FTAL**	**RTLL**	**RTAL**
Test Acc	55.70	57.30	61.30	59.00	45.00
CAEAcc	100.00	100.00	100.00	100.00	82.00
	Retrain				
Metric	**ResNet20**	**ResNet56**	**Densenet**	**VGG19**	**MobileNetV2**
Test Acc	53.10	54.50	50.95	52.50	52.87
CAEAcc	97.00	99.00	90.00	76.00	97.00
	Extraction				
Metric	**Jagielski**	**Papernot**	**Knockoff**		
Test Acc	53.20	50.90	47.40		
CAEAcc	98.00	90.00	98.00		

Table A.2: Reference model accuracies for ImageNet32.

	ResNet20	**ResNet56**	**Densenet**	**VGG19**	**MobileNetV2**
Test Acc	55.00	59.10	55.95	54.10	62.90
CAEAcc	47.00	50.00	47.00	34.00	61.00

A.3 Fingerprinting Definitions

In this section, we formally define the term 'surrogate' and provide security games for *irremovability* and *non-evasiveness* of DNN fingerprinting. For ease of notation, we define an auxiliary function to generate a fingerprint for a source model. enumerate

Definition of Surrogates. We define the randomized process 'distill' that on the input of a source model M and a dataset $D \in \mathcal{D}$, returns a distilled surrogate S of the source model. Note

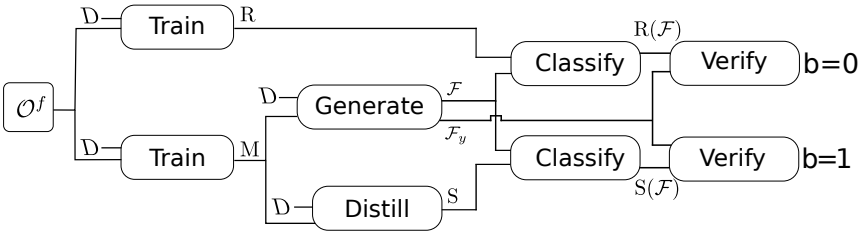


Figure A.3: A schematic illustration of the source model and the two types of models, the surrogate S and the reference model R, that a fingerprint verification should distinguish. DISTILL is any distillation attack that results in a surrogate model with similar performance as the source model, and CLASSIFY returns the predicted output of a model an image.

Algorithm 19 FModel Generation

```
procedure FMODEL
    M ← T(D)
    (F, F_y) ← GENERATE(M, D)
    return M, F, F_y
```

that other common derivation methods, such as retraining, fine-tuning, and pruning on labels provided by the source model, can be expressed in terms of knowledge distillation [86].

For a given source model M we recursively define the set $\mathcal{S}_M$ of its *surrogate* models as follows.

$$S_M \leftarrow \{\text{DISTILL}(S, D) | S \in \mathcal{S}_M \cup \{M\}\} \tag{A.1}$$

If a model $R \in \mathcal{M}$ is trained from a labeled dataset without access to $\mathcal{S}$ and hence is not contained in the set of surrogate models for source model M, we refer to that model as *reference model* relative to M. The goal of this work is to find a method that verifies whether a surrogate model is in $\mathcal{S}_M$.

Irremovability. The attacker accesses the source model M to derive a surrogate $\hat{M} \leftarrow \mathcal{A}(M)$. We say fingerprinting is not removable if the adversary has a low probability of winning the following security game.

Note that our definition of a robust fingerprint differs from related work [23] because we include in the set of removal attacks $\mathcal{A}$ also model extraction attacks. Fig. A.3 schematically shows the types of models that a fingerprint verification should distinguish.

Non-Evasiveness. Hitaj and Mancini [87] show that an attacker can evade the black-box verification by rejecting queries or returning random labels when the verification process is detectable. We specify a *non-evasiveness* property, which makes it hard to separate members of the fingerprint to benign data samples. A fingerprinting method is non-evasive, if for some

Algorithm 20 Defense Strategy

procedure DEFENSE($\mathcal{O}, \mathcal{D}, \mathcal{A}$)
  $(M, \mathcal{F}, \mathcal{F}_y) \leftarrow$ FMODEL ▷ *compute model and fingerprint*
  $\hat{M}_0 \leftarrow$ TRAIN($\mathcal{O}, \mathcal{D}$) ▷ *train oracle model*
  $\hat{M}_1 \leftarrow$ ADVERSARYTRAIN(M) ▷ *adversary trains model*
  $b \xleftarrow{\$} \{0, 1\}$ ▷ *randomly select a model*
  Send $\hat{M}_b$ to the Defender
  if $\Pr[\text{VERIFY}(\hat{M}_b(\mathcal{F}), \mathcal{F}_y) = b] \approx 0.5$ **then**
    Adversary wins

$\varepsilon \in \mathbb{R}^+$, the adversary has a low probability of winning the following security game. Let $\text{Adv}(\mathcal{D}, \epsilon)$ be an adversarial attack with maximum perturbation $\epsilon \in \mathbb{R}^+$ performed on the dataset D. An adversarial example with a perturbation smaller than ϵ can be hard to detect [24]. The security game is defined as follows.

Algorithm 21 Adversary Detection

procedure ADVERSARYDETECTION
  $(M, \mathcal{F}, \mathcal{F}_y) \leftarrow$ FMODEL
  $\hat{M} \leftarrow$ TRAINDETECTOR(M)
  $b \xleftarrow{\$} \{0, 1\}$
  $X_0 \xleftarrow{\$}$ ADV(D, ϵ) ▷ *generate adversarial examples*
  $X_1 \xleftarrow{\$} \mathcal{F}$
  Send X_b to the adversary
  if $\Pr[\hat{M}(X_b) = b] > 0.5 + \epsilon$ **then**
    Adversary wins

Note that ε bounds the trade-off between evasiveness and the suspect model's performance. When ε becomes smaller, more queries from benign users are falsely rejected, or a random label is returned at the same level of evasiveness. The goal of the attacker is to find a model that maximizes evasiveness while also maximizing model performance.

A.4 Optimizing Conferrability

In this section, we formally present the optimization problem that has to be solved to find highly conferrable adversarial examples. We denote the set of surrogate models for a source model M by $\mathcal{S}_M$ and the set of reference models as $\mathcal{R}$.

Conferrability scores should be minimal when all reference and surrogate models assign the same target label (perfect transferability), or none of them assign the target label (non-transferability). An adversarial example that is perfectly transferable to surrogate models and non-transferable to reference models should have the maximum conferrability score. The following formula satisfies these constraints for a target class $t \in \mathcal{Y}$.

$$\text{Confer}(\mathcal{S}, \mathcal{R}, x; t) = \text{Transfer}(\mathcal{S}, x; t)(1 - \text{Transfer}(\mathcal{R}, x; t)) \tag{A.2}$$

Note that Equation A.2 does not have weight factors, as it is equally important to transfer to surrogate models as it is to not transfer to reference models.

The optimization constraints to find adversarial examples with high conferrability scores can be formalized as follows for a benign input $x \in \mathcal{X}$ and a perturbation δ in the infinity norm.

Minimize δ s.t.

1. $M(x + \delta) = t$
2. $\underset{S \in \mathcal{S}}{Pr}[S(x + \delta) = t] \approx 1$
3. $\underset{R \in \mathcal{R}}{Pr}[R(x + \delta) \neq t] \approx 1$

subject to $||\delta||_\infty \leq \epsilon$

The challenge of generating conferrable examples is to find a good optimization strategy that (i) finds local minima close to the global minimum and (ii) uses the least amount of surrogate and reference models to find those conferrable examples. Obtaining many surrogate and reference models that allow optimizing for conferrability is a one-time effort, but may be a practical limitation for datasets where training even a single model is prohibitively expensive.

A.5 Removal Attacks

In this section, we provide more details on the removal attacks and their parametrization.

A.5.1 Model Extraction

Retraining uses the source model M to provide labels to the training data D and then uses these labels to train the surrogate model S.

$$S \leftarrow \mathcal{T}(D) \tag{A.3}$$

Jagielski et al. [104] post-process the labels received from the source model by a distillation parameter T' to obtain soft labels. For an input $x \in \mathcal{D}$ and a source model M, the soft labels $M'(x)$ can be computed as follows.

$$M'(x)_i = \frac{\exp(M(x)_i^{1/T'})}{\sum_j \exp(M(x)_j^{1/T'})} \tag{A.4}$$

The surrogate model is trained on the soft labels.

$$S \leftarrow \textsc{Train}(M', D) \tag{A.5}$$

Papernot et al. [171] propose training the surrogate model iteratively starting with an initial dataset D_0 that is concatenated after each round with a set of the surrogate model's adversarial examples.

Algorithm 22 Sequential Model Training

```
S_0 ← T(D_0)                                          ▷ Initial training with dataset D_0
for i = 0, 1, 2, ... do                               ▷ Iterative training process
    D_{i+1} ← {x + λ · sign(JACOBIAN(S_i, M, x)) | x ∈ D_i} ∪ D_i   ▷ Dataset update
    S_{i+1} ← FINETUNE(M, D_{i+1})                    ▷ Train model with updated dataset
```

J_S denotes the Jacobian matrix computed on the surrogate model S for the labels assigned by the source model M. We use the FGM adversarial attack to generate adversarial examples, as described by the authors.

Knockoff [169] uses cross-domain transferability to derive a surrogate model. They construct a *transfer set*, which is cross-domain data used to derive the surrogate model. We implement the random selection approach presented by the authors

A.6 Empirical Sensitivity Analysis

We perform a grid search to understand the sensitivity of the choice of hyperparameters in Equation (5.6) on the mean conferrability score over $n = 50$ inputs. Mean conferrability is evaluated over five surrogate and five reference models trained on CIFAR-10, which have not been used in the generation process of the conferrable examples. We evaluate all parameters in the range $[0.1, 0.5, 1.0]$ and leave out all combinations where $\alpha = \beta = \gamma$, except for the baseline used in Chapter 5 ($\alpha = \beta = \gamma = 1$). The examples are generated for $\epsilon \approx 6/255$ with an Adam optimizer [116] with a learning rate 5e−4, and we optimize for 300 iterations.

Algorithm 23 Model Fingerprinting

procedure GENERATE($M, D, \mathcal{O}$)
  $\mathcal{S} \leftarrow \{\text{TRAIN}(M, D) | i \in \{1..c_1\}\}$ ▷ *Train surrogate models*
  $\mathcal{R} \leftarrow \{\text{TRAIN}(\mathcal{O}, D) | i \in \{1..c_2\}\}$ ▷ *Train reference models*
  $\mathcal{D}_C \leftarrow \{x \in D | \arg\max_i M(x)_i = \arg\max_j \mathcal{O}(x)_j\}$ ▷ *Filter benign inputs*
  $M_E \leftarrow \text{COMPOSE}(M, \mathcal{S}, \mathcal{R})$ ▷ *Generate conferrable examples*
  $\mathcal{F}' \leftarrow \emptyset$
  for $x_0 \in \mathcal{D}_C$ **do**
    $\mathcal{F}' \leftarrow \mathcal{F}' \cup \{x_0 + \arg\min_\delta(\mathcal{L}(x_0, M_E(x_0 + \delta)))\}$
  $\mathcal{F} \leftarrow \{x' \in \mathcal{F}' | \text{CONFER}(\mathcal{S}, \mathcal{R}, x') \geq \tau\}$ ▷ *Apply threshold*
  return $\mathcal{F}, M(\mathcal{F})$ ▷ *Output fingerprint and key*

Algorithm 24 Model Verification

procedure VERIFY($\hat{M}(\mathcal{F}), \mathcal{F}_y$)
  $\mathcal{F}'_y \leftarrow \text{PREDICT}(\hat{M}, \mathcal{F})$ ▷ *Get suspect model's output*
  $p_{ret} \leftarrow \frac{1}{|\mathcal{F}_y|} \sum_{i=0}^{|\mathcal{F}_y|} \mathbb{1}_{\arg\max_j \mathcal{F}_{y_{ij}} = \arg\max_j \mathcal{F}'_{y_{ij}}}$ ▷ *Calculate agreement*
  return $p_{ret} \geq \theta$ ▷ *Return verification result*

The results are illustrated in Table A.3. We find that when α is small, the optimization does not converge, and the source model does not predict the target label. When α is large relative to the other parameters, we measure the highest mean conferrability. For $\alpha = 1.0$ and $\beta = \gamma = 0.1$, we measure a mean conferrability of 0.49.

α	β	γ	**Conferrability**	α	β	γ	**Conferrability**
0.1	0.1	0.5	0.20	0.5	1.0	0.1	0.44
0.1	0.1	1.0	0.20	0.5	1.0	0.5	0.37
0.1	0.5	0.1	0.20	0.5	1.0	1.0	0.29
0.1	0.5	0.5	0.20	1.0	0.1	0.1	**0.49**
0.1	0.5	1.0	0.20	1.0	0.1	0.5	0.42
0.1	1.0	0.1	0.20	1.0	0.1	1.0	0.40
0.1	1.0	0.5	0.20	1.0	0.5	0.1	0.45
0.1	1.0	1.0	0.20	1.0	0.5	0.5	0.37
0.5	0.1	0.1	0.20	1.0	0.5	1.0	0.37
0.5	0.1	0.5	0.20	1.0	1.0	0.1	0.41
0.5	0.1	1.0	0.20	1.0	1.0	0.5	0.42
0.5	0.5	0.1	0.20	1.0	1.0	1.0	0.40
0.5	0.5	1.0	0.20				

Table A.3: An empirical sensitivity analysis of the hyperparameters in Equation (5.6). Conferrability measures the mean conferrability score over five surrogate and five reference models trained on CIFAR-10.

Appendix B

SoK: How Robust is Deep Neural Network Image Classification Watermarking?

Appendix B.3 describes the datasets used in our experiments. Section 4.2 describes the hyper-parameters used for all surveyed watermarking schemes in our ablation study. Appendix B.2 describes all surveyed removal attacks, including novel attacks such as weight shifting, and contains a description of the parameters we used in the ablation study. A detailed description of each approach can be found in the author's papers. Section 4.3 provides further details on the computation of the decision thresholds (see Section 4.5.1).

B.1 Watermarking Methods

In this section, we present the surveyed watermarking schemes and the parameters used for our ablation study. For simplicity, we refer to a watermarking scheme by the first author's name unless it is known under a different name.

B.1.1 Model Independent

Adi [3]. We embed the same 100 watermarking keys used by the authors[1]. Images are resized along their shortest side to the dimensions of the training data, followed by center cropping. For

[1] https://github.com/adiyoss/WatermarkNN

ImageNet, we embed using early stopping [30] on the watermarking loss with a patience of five, evaluated at the end of every 200th batch. The watermarking loss is the cross-entropy loss of the model computed on the watermarking key. We ablate over the learning rate lr $\in \{10^{-3}, 10^{-4}\}$. To speed up the embedding, we repeat the watermarking keys 1000 times for ImageNet and 100 times for CIFAR-10.

Zhang [255]. The authors propose three different schemes, referred to as *Content*, *Noise* and *Unrelated*.

- **Content**: We use a white square embedded at the top left corner of the image. The square's size is $s \in \{32, 128\}$ for ImageNet and $s \in \{8, 16\}$ for CIFAR-10.
- **Noise**: We add the noise across the entire image and clip the resulting values into the range $[0, 1]$. We ablate over the standard deviation $\sigma \in \{0.4, 1.0\}$ for both ImageNet and CIFAR-10. For CIFAR-10, we ablate over the learning rate during the embedding lr $\in \{10^{-3}, 10^{-4}\}$.
- **Unrelated**: We sample watermarking images from the Omniglot dataset [125] for both CIFAR-10 and ImageNet. We ablate over the learning rate lr $\in \{10^{-3}, 10^{-4}\}$.

For CIFAR-10, we randomly sample the source-target class pair 'cat' and 'dog' and for ImageNet, we sample 'tiger shark' and 'stingray'. We use early stopping on the watermarking loss during the embedding and repeat the watermarking keys 1000 times for ImageNet and 100 times for CIFAR-10.

B.1.2 Model Dependent

Frontier-Stitching [127]. We use FGM [77] to generate adversarial examples and ablate over the perturbation threshold perturbation threshold $0.1 \leq \epsilon \leq 0.25$.

Blackmarks [34]. We ablate over the loss term that minimizes the bit error rate between the predicted cluster and the assigned cluster $0.01 \leq \lambda \leq 100$.

Jia [109]. We sample the watermarking key from the training data and use a square as the secret trigger pattern (same as the authors). For CIFAR-10, we compute the source class 4 ('deer') and target class 6 ('frog'). We use SNNL weights $w \in \{0.25, 1, 4\}$ and a rate of $r = 2$, i.e., every second batch consists of watermark data. The trigger has a size of 3×3 pixels and resets values to zero in the image across all three channels. For ImageNet, we compute source class 3 ('tiger shark, Galeocerdo cuvieri') and target class 4 ('hammerhead, hammerhead shark'). We use an SNNL weight $w = 64$ and a ratio of ten during the embedding using a square trigger with 5×5 pixels. We compute the SNNL on a single layer, as mentioned by the authors, due to GPU memory restrictions when computing the SNNL on all layers. When embedding 100 elements with a batch size of 64, we observe the convergence of the SNNL and cross-entropy losses after about 100k images are shown to the source model.

B.1.3 Parameter Encoding

Uchida [227]. We embed the Uchida watermark with early stopping on the loss during training and a patience of five, whereby we evaluate the condition at the end of every epoch for CIFAR-10 and after every 200 batches for ImageNet. The target layer has 9 408 weights for the ImageNet models and 432 for CIFAR-10 models. For CIFAR-10, we ablate over the constant weight factor of the embedding loss $\lambda \in \{0.1, 1, 10\}$ and for ImageNet, we ablate over $\lambda \in \{1, 10\}$.

DeepMarks[2] [33]. We ablate over the embedding strength $\gamma \in \{0.1, 10\}$ and use the same target layer as in Uchida.

DeepSigns [192]. In the author's paper, clusters are modelled using a Gaussian Mixture Model, whereby each feature cluster c_i is described by a mean μ_i and a standard deviation σ_i. In our experiments, we had difficulties embedding the watermark in ImageNet models using more than $m = 1$ Gaussian distributions because of instabilities during training.

Even after extensive parameter search, we observe that for $m > 1$ (i) the test accuracy drops significantly over time, and (ii) the regularization loss does not converge. The authors do not provide source code, nor did they validate their scheme for ImageNet. We solve the issue for ImageNet by modifying two elements of the embedding procedure.

- **Single Gaussian**: We use $m = 1$ Gaussian and $n = 100$ bits to embed the message on ImageNet. We observe that the regularization loss converges.
- **Alternating Training**: We train on the whole dataset without the regularization loss for two batches. Then, we fine-tune with the embedding loss on samples from the source class for one batch. We observe that this stabilizes training and maintains a high test accuracy.

We can replicate the author's result on CIFAR-10 by using $m = 10$ Gaussian distributions (one for each class) and embedding $n = 10$ bits per Gaussian. On ImageNet, we embed the watermark into a layer with 25 088 features and 24 576 features for CIFAR-10.

B.1.4 Active Schemes

DAWN [216]. We ablate over the expected rate $r \in \{0.01, 0.02\}$ at which a false label is returned.

[2]DeepMarks is labeled as a fingerprint by the authors, but since it modifies the model by embedding a message, it is a watermark as per our definition.

B.2 Watermark Removal Attacks

In this section, we describe the parameters used in our ablation study for all removal attacks surveyed in Chapter 4, sorted by their attack category. We make configuration files that show the parameter ablations for all removal attacks publicly available as part of our Watermark-Robustness-Toolbox (WRT).

B.2.1 Input Preprocessing

Input Reconstruction [136]. uses an autoencoder[3] [166] to compress and reconstruct images before passing them to the surrogate model. We ablate over the size of its bottleneck layer $64 \leq h \leq 512$. We do not perform Input Reconstruction on ImageNet because, to the best of our knowledge, no high-fidelity autoencoder for ImageNet is available.

Input Noising [253]. We ablate over the standard deviation $0.01 \leq \sigma \leq 0.2$ for Gaussian noise with zero mean.

Input Quantization [134]. For a given number of bits b we discretize the input space into 2^b evenly spaced intervals, referred to as *quanta*. We project every value of the input image to the mean of its quantum and ablate over the number of bits $b \in \{3, 4, 5\}$.

Input Smoothing [241]. We use a mean, median, and Gaussian kernel. For the mean and median kernels, we use a filter size of three, and for the Gaussian kernel, we ablate over the standard deviation $0.1 \leq \sigma \leq 0.3$.

Input Flipping. We flip an image along its horizontal axis.

JPEG Compression [57]. We ablate over a parameter $5 \leq q \leq 95$ that controls the quality of the compression.

Feature Squeezing [241]. The quanta values are chosen to be multiples of 0.5^k for some $1 \leq k \leq 6$.

B.2.2 Model Modification

Adversarial Training [151]. We inject about 10% of the training dataset's size with adversarial examples generated using Projected Gradient Descent [151] for $\epsilon \in \{0.01, 0.1, 0.25\}$, a step size of 0.01 and a maximum number of 40 iterations. Each adversarial example is repeated twice, and we fine-tune the surrogate model for five epochs.

[3] https://github.com/foamliu/Autoencoder

Feature Permutation. DNNs are invariant to feature permutations, meaning that neurons in a hidden layer can be permuted without affecting the model's functionality. We use (random) feature permutation as an adaptive attack designed specifically against Deepsigns [192], which encodes the message into the activations of hidden layers.

Fine-Pruning [139]. We ablate over the sparsity $0.8 \leq \rho \leq 0.95$ and fine-tune for ten epochs on CIFAR-10 and five epochs on ImageNet.

Fine-Tuning [227]. Fine-Tuning as a model stealing attack refers to a set of attacks that first apply a transformation to the model, followed by fine-tuning.

- Fine-Tune All Layers (**FTAL**). All weights are fine-tuned.
- Fine-Tune Last Layer (**FTLL**). All but the last layer's weights are frozen while the model is fine-tuned.
- Retrain All Layers (**RTAL**). The last layer's weights are re-initialized, and all weights are fine-tuned.
- Retrain Last Layer (**RTLL**). The last layer's weights are re-initialized, and only that layer's weights are fine-tuned.

RTAL and RTLL use predicted labels, whereas FTAL and FTLL use ground-truth labels (otherwise, gradients are zero).

Label Smoothing [215]. We use a weight of $\epsilon = 0.3$ for the weighted sum between the prediction and a uniform vector.

Regularization [202]. We L2-regularize for five epochs on CIFAR-10 and one epoch on ImageNet using a weight decay of 0.1 (two orders of magnitudes higher than during training).

Neural Cleanse [232]. We implement both *unlearning* and *pruning* methods proposed by the authors and ablate over the learning rate $10^{-3} \leq \alpha \leq 10^{-2}$ for unlearning and the sparsity $0.8 \leq \rho \leq 0.99$ for pruning.

Neural Laundering [5]. We ablate over the activation threshold to prune convolutional layer neurons $0.03 \leq c \leq 3$ and the learning rate for fine-tuning $10^{-4} \leq \alpha \leq 10^{-2}$.

Weight Pruning [264]. We ablate over the sparsity $0.1 \leq \rho \leq 0.95$ for the trainable weights of each layer.

Weight Shifting. Weight shifting is a novel, adapted attack against parameter encoding watermarking schemes. The idea is to apply a small perturbation to all filters of each convolutional layer in the network, followed by fine-tuning the model to regain the loss in test accuracy. We design weight shifting as an efficient and effective model stealing attack specifically against Uchida [227] and Deepmarks [33].

We explain the attack's idea at the example of Uchida, but a similar intuition holds for Deepmarks where the extraction is highly similar. Let $W \in \mathbb{R}^{n \times c \times w \times}$ be the convolutional filters of a target layer, where n is the number of filters, c are the number of channels, and w, h are the width and height of each filter. A weakness of Uchida exploited by weight shifting is that the attacker knows that if all convolutional filters were inverted, i.e. $W_i' = -W_i$, then the watermark accuracy would be zero. We cannot directly invert all filters, as the model experiences a significant drop in test accuracy. Hence, we construct a 'softer' version of the attack that only moves each filter in the direction of the inverse mean multiplied by some constant weight parameter $\lambda_1 \in \mathbb{R}$. We additionally add small random Gaussian noise to each filter to encourage the network to find slightly different filters in the fine-tuning phase.

Our attack can be formalized by the function $S(W; \lambda_1, \lambda_2)$, which takes as input a set of filters W and outputs a shifted set of filters W'. The parameter λ_1, λ_2 trade off the attack's efficiency with its effectiveness. Let A be a random normal matrix of the same shape as each filter W_i with a variance equivalent to the variance over all filters for a convolutional layer and a mean of zero. Shifted weights for each convolutional layer can be computed by applying the following function.

$$\mathrm{S}(W; \lambda_1, \lambda_2)_i = W_i - \frac{\lambda_1}{n} \sum_{j=1..n} W_j - \lambda_2 A \tag{B.1}$$

In our experiments, we use $\lambda_1 = 1.5, \lambda_2 = 1.0$ for CIFAR-10 and $\lambda_1 = 1.3, \lambda_2 = 0$ for Imagenet. We fine-tune the model for ten epochs on CIFAR-10 and for five epochs on ImageNet.

Weight Quantization [96]. We ablate over the bit-size $b \in \{4, 5\}$ (i.e., there are 2^b discrete states) for CIFAR-10 and ImageNet and fine-tune the model for one epoch.

B.2.3 Model Extraction

Retraining [224]. We use the same parameters for the surrogate model that were used to train the source model.

Smooth Retraining. Smooth Retraining trains a surrogate model on smoothed labels obtained from querying the source model for multiple variations of the same image. For each query, a random affine transformation (e.g., random cropping) is applied to the image, and the mean of all received labels is computed as the final label. We design smooth retraining as an adaptive attack against the active watermarking scheme DAWN. The intuition is that if DAWN responds with a false label for one image, variations of the same image have a high probability of receiving the label predicted by the source model. In our experiments, we use $n = 3$ queries.

Knockoff Nets [169]. We implement the random selection approach on the Open Images [124] dataset.

Transfer Learning [223]. Transfer Learning is an established method from related work, where a pre-trained model from a different domain is fine-tuned for a new domain. We propose using transfer learning as a novel method to remove DNN watermarks. We use a pre-trained ResNet-101 model[4] for Open Images (v2) [124] that was published by Google in 2017. The model defines an output layer with 5k output classes, which we replace by a layer with ten output classes for CIFAR-10 and 1k output classes for ImageNet. We transfer-learn the model using stochastic gradient descent (SGD) and freeze all but the last layer for the first 300 batches. We proceed by training the entire model for five epochs and reduce the learning rate by a factor of ten in epochs three and four.

Adversarial Training (from scratch) [151]. This method is equivalent to adversarial training described earlier, except that the attacker trains the surrogate model from scratch.

B.3 Datasets

We now describe the datasets used in our experiments.

- **CIFAR-10** [121] contains 50k training images and 10k testing images from 10 classes. All images have a resolution of 32×32 pixels.
- **ImageNet** [47] contains 1.28 million training images and 150k testing images from 1k classes. We resize and center crop all images to 224×224 pixels.
- **Open Images** [124] defines 19.794 classes and contains in total 8.85 million training images, out of which we use a subset of 1.7 million images due to storage constraints on our machines. Multiple classes can label images. We resize and center-crop all images to 224×224 pixels.

All source models are trained on either CIFAR-10 or ImageNet. The Open Images dataset is only used in the transfer learning attack. We use standard training procedures and data augmentation, such as horizontal flipping, to train models for CIFAR-10 and ImageNet from scratch. On CIFAR-10 and ImageNet, the source models achieve a test accuracy of 94.20% and 75.48%, respectively.

B.3.1 Dataset Splitting

We split the whole training dataset into thirds and assigned two-thirds to the defender for embedding the watermark. For the attacker's training data, we distinguish between the availability of the following three datasets to the attacker.

[4] https://storage.googleapis.com/openimages/2017_07/oidv2-resnet_v1_101.ckpt.tar.gz

1. **Labeled**: Data from the same distribution where a subset of at most a third of the data is labeled.
2. **Domain**: Unlabeled data from the same distribution.
3. **Transfer**: Labeled data from a different distribution.

In the first two cases, we assign the remaining third of the training dataset to the attacker. We make an exception for model extraction attacks, where the attacker can access the whole training dataset without labels. Such an exception is necessary because model extraction attacks require a substantial amount of data to output well-trained surrogate models. We underpin this argument with an ablation study in Section 4.6.5. Otherwise, if the attacker is given domain data, we replace all labels with the predictions of the source model.

B.4 Estimating the Decision Threshold

For model-independent, model-dependent, and active watermarking schemes, we use 20 publicly available, pre-trained models from the torchvision[5] package that do not necessarily share the source model's architecture (ResNet-50). We use the following model architectures.

ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152 [82], Wide ResNet-50, Wide ResNet-101 [251], VGG11, VGG13, VGG16, VGG19 [208], SqueezeNet [100], DenseNet-121, DenseNet-161 [94], GoogleNet [214], Alexnet, Alexnet-50 [122], InceptionNet [215], MobileNetV2 [197]

B.5 Datasets

We survey three datasets that we split into *training*, *validation*, and *testing* sets. The training and validation sets are equally large and are used to train the target and shadow models, respectively. We evaluate the perplexity of all models on the testing set, which neither the target nor the shadow models have seen during training.

ECHR. ECHR contains cases from the European Court of Human Rights, which consists of 118 161 records. Each record consists of 88.12 tokens on average. A case contains a numbered list of *facts*, describing the case, such as persons involved or an order of events. We split the dataset so that each record contains a single fact.

Flair NER tags 16 133 unique PII of the entity class 'person' and we identify that 23.75% of the records contain at least one PII sequence. PII is duplicated according to a power law

[5] https://pytorch.org/vision/stable/models.html

distribution with a mean duplication rate of 4.66. Most PII ($\geq 90\%$) sequences are duplicated less than 3 times. For ECHR, many PII from the 'person' class are full names (containing a first and a second name), but there are also name abbreviations (e.g., 'J.D.' instead of 'John Doe').

Enron. Enron contains about 600 000 real e-mails from 158 employees of the Enron Corporation. The e-mails were made public by the Federal Energy Regulatory Commission after an investigation. We observe that e-mails use informal language and may contain typographical errors. E-mails do not contain a header (e.g., a 'from', 'to' or 'title' field) unless an e-mail has been forwarded, in which case the forwarded e-mails and their headers are contained in the e-mail's body. The e-mails are typically structured with a greeting, followed by the e-mail's content, followed by a footer containing the name and personal information about the sender, such as their e-mail or work address, phone number, and homepage. We randomly sample a subset of 123 317 records from the Enron dataset, where each record consists of 346.10 tokens on average. Flair tags 105 880 unique PII sequences from the 'person' entity class, and most records contain at least one PII sequence (81.45%). PII sequences have a duplication rate of 11.68 and contain an average of 3.00 tokens. From qualitative analysis, we observe that PII in the Enron dataset consists of either just the first name or a first and a second name.

Yelp-Health. The Yelp-Health dataset consists of comments made on the `yelp.com` website about facilities from the 'Health & Medical' category. Facilities from this category include Psychoanalysts, Dental Hygienists, and Cardiologists, among many others. Compared to ECHR and Enron, Yelp-Health reviews contain relatively short and few PII from the 'person' entity class.

We randomly sample a subset of 78 794 reviews with an average length of 143.92 tokens per record. Flair tags 17 035 pieces of PII with an average duplication rate of 5.53 and 2.17 tokens per piece. In total, 54.55% of records contain at least one PII sequence. From a qualitative analysis, we find that pieces of PII from the 'person' entity class typically only contain the first name unless they refer to the name of the doctor, who is usually addressed by their last name.

Even though reviews on Yelp-Health were knowingly posted to a public forum[6], these posts may contain sensitive information. For example, we find detailed descriptions of a relative's disease with timestamps and locations, which could be used to re-identify that individual. We believe that studying leakage using datasets containing unprocessed data from users who may be unaware of the consequences resulting from re-identification is of particular interest to the community.

B.5.1 PII Definition

This subsection describes what we mean by "direct" and "quasi-identifying" PII. Our definition is equivalent to Pilán et al. [177], who study leakage of PII in large text datasets.

[6] https://www.yelp.com/developers/documentation/v3/all_category_list

- *Direct identifiers*: A direct identifier is a type of PII that can be used to uniquely identify an individual within a dataset. Examples of direct identifiers include an individual's full name, social security number, address of residence, cellphone number, or email address. As a result of their sensitive nature, direct identifiers are often subject to legal and regulatory frameworks aimed at safeguarding the privacy of individuals [61].
- *Quasi-identifiers*: A of PII is a quasi-identifier when it can indirectly identify an individual through its combination with other quasi-identifying information. Examples of quasi-identifiers include an individual's age, gender, ethnicity, religion, and occupation. Because of their potential to indirectly identify individuals, both direct and quasi-identifiers are considered personal information. They are subject to the same legal and regulatory frameworks in order to protect the privacy of individuals [61].

B.5.2 PII Entity Classes

We group PII by the following *entity classes*[7].

1. cardinal: A cardinal value (e.g., "12").
2. ordinal: An ordinal value (e.g., "11th").
3. date: A date (e.g., "May 23rd 2015")
4. event: An event name (e.g., "Blackhat").
5. fac: A building name (e.g., "Newark Airport").
6. gpe: Geo-political entity (e.g., "UAE").
7. language: A language name (e.g., "Catalan")
8. law: Law names (e.g., "Bill C-99").
9. money: Currency (e.g., "1.25m USD")
10. norp: Affiliation (e.g., "Alaskan")
11. person: Names (e.g., "John Henry Doe")
12. loc: Location (e.g., "11th District")
13. org: Organization (e.g., "Microsoft")
14. percent: (e.g., "(+0.78%)")
15. product: (e.g., "GX-532")
16. quantity: (e.g. "2000 lbs")

[7] https://huggingface.co/flair/ner-english-ontonotes-large

17. time: (e.g., “5:30 pm EDT”)
18. work of art: (e.g., “Star Wars”)
19. phone number: (e.g., “(+1)123-456-7890”)
20. email address: (e.g., “john.doe@anon.com”)
21. url: (e.g., “www.johndoe.com”)

Replacing PII

The attack described in Algorithm 5 involves substituting existing PII with PII of the same class. For example, in the sentence “Teo Peric is an engineer.” we could replace “Teo Peric” with a different PII of the class ’person’, resulting in “Ana Jaksic is an engineer.”.

B.5.3 PII Distribution

Table B.1 show the PII counts in each of the three surveyed datasets.

Table B.1: A summary of the evaluated datasets and statistics for PII from the entity class ‘person’. This table summarizes the part of the dataset that was used to train the model after splitting the entire dataset into equally large *training* and *validation* sets and a small *testing* set.

	Records	Tokens / Record	PII	Rec. w. PII	Dup. / PII	Tokens / PII
ECHR	118 161	88.12	16 133	23.75%	4.66	4.00
Enron	138 919	346.10	105 880	81.45%	11.68	3.00
Yelp-Health	78 794	143.92	17 035	54.55%	5.35	2.17

B.5.4 PII Leakage Metrics

We define formally the metrics introduced in Section 3.5.4.

1. **PII Extractability:** We assess PII extractability using precision and recall. Precision (also called positive predictive value) quantifies the ratio of true positive predictions to all positive predictions made by a classifier, while recall is the true positive rate, i.e. the ratio of true positive predictions to all positive instances in the dataset. High recall implies that a significant proportion of PII can be extracted from the LM’s generated output, and high

Algorithm 25 Fill residual masks

```
procedure FILL-MASKS(S)
    i ← 0
    while [MASK] ∈ S do
        S_i, S ← SPLIT(S, [MASK])                ▷ At first [MASK]
        i ← i + 1
    S_i ← S
    S' ← S_0
    for j ← 1 to i do
        w_j ← MLM(S', S_j)
        S' ← S' w_j S_j
    return S'
```

precision suggests that a generated PII is highly probable to appear in the training dataset. Algorithm 5 represents the PII extraction game, and Equation (3.1) defines the adversary's success as its recall. Given the number of unique PII sequences $|\mathcal{C}|$ in D, the adversary produces a set of PII sequences $\tilde{\mathcal{C}}$ with a maximum size of $|\mathcal{C}|$. PRECISION denotes the expected fraction of correctly identified PII sequences in $\tilde{\mathcal{C}}$ relative to the total number of PII sequences produced by the adversary ($|\tilde{\mathcal{C}}|$). Recall and precision are formally calculated as follows:

$$\text{RECALL} = \mathbb{E}\left[\frac{|\mathcal{C} \cap \tilde{\mathcal{C}}|}{|\mathcal{C}|}\right] \quad \text{PRECISION} = \mathbb{E}\left[\frac{|\mathcal{C} \cap \tilde{\mathcal{C}}|}{|\tilde{\mathcal{C}}|}\right] \tag{B.2}$$

2. **PII Reconstruction and Inference:** We measure the top-1 accuracy for PII reconstruction and inference as formalized in Algorithm 6. Given a randomly sampled sequence S from the training dataset that contains at least one piece of PII C, we compute the accuracy of an attacker to produce a guess $\tilde{C} = C$ for a randomly selected [MASK] within the sequence.

$$\text{Succ}_{\text{RECON}} = \Pr\left[\tilde{C} = C\right] \tag{B.3}$$

The attack's success depends mildly on the presence of other PII in the sampled sequence S. This is because other PII sequences would be scrubbed, and the adversary would have less contextual information to infer. Figure 3.4 illustrates this point for a sequence containing multiple pieces of PII.

B.6 Filling Residual Masks

Algorithm 25 describes our procedure for filling residual masks in a masked query. We use a publicly available masked language model (MLM) to fill in one masked token at a time. Given a

sentence with multiple masked PII, for example "`[MASK]` plays soccer in `[MASK]`, England." the goal of FILL-MASKS is to fill in all masked tokens in a way that preserves the meaning of the sentence.

Publicly available MLMs do not jointly fill multiple tokens. Hence we resort to filling in masked tokens from left to right using the top token suggested by the MLM. In the example above, we would query the MLM twice. The first query is "`[MASK]` plays soccer in" and the second is "John plays soccer in `[MASK]`, England.", assuming the MLM fills the first mask with "John".

Given a sentence $S = S_0\texttt{[MASK]}S_1 \ldots \texttt{[MASK]}S_i$, algorithm 25 first identifies the subsequences $S_0, \ldots, S_i$ (lines 2–6) and then queries the MLM to fill in each masked token separating them from left to right (lines 7–10). The function $\text{MLM}(S_0, S_1)$ queries the MLM on $S_0\texttt{[MASK]}S_1$ to predict the most likely token in the position of `[MASK]`.

Appendix C

PTW: Pivotal Tuning Watermarking for Pre-Trained Iamge Generators

This section describes the attacks and shows which parameters we explored in our grid search. We refer to the image before attacking as the *base* image x and as the *attacked* image after the attacker performs their attack $\tilde{x} = \mathcal{A}(x, I)$ with auxiliary information I. We find the range of evaluated parameters through experimentation by limiting the degradation of the attack on the visual quality of the images.

C.1 Black-box Attacks

Black-box attacks assume only black-box API access to the *target generator*, meaning that they can query the generator on arbitrary latent codes $z \in \mathcal{Z}$, but they have no knowledge of or control over the generator's parameters or its intermediate activations.

The black-box attacks can be described as follows.

- **Crop**: First, the base image is center-cropped with a given cropping ratio $\rho \in (0, 1]$ and then resizes the cropping back to the base image's original size. We experiment with cropping ratios $\rho \in [0.9, 1]$.
- **JPEG Compression**: This attack performs JPEG compression [231] on the base image with a quality q. A higher quality better preserves the visual quality of the image and we experiment with $q \in [80, 200]$.
- **Noise**: This attack adds Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the image. We experiment with $\sigma \in (0, 0.05]$.

Algorithm 26 Super-Resolution Attack

procedure SUPER-RESOLUTION(x, ρ)
    $\tilde{x} \leftarrow$ RESIZE(x, $\lfloor$RESOLUTION(x) $\cdot\ \rho\rfloor$)
    while RESOLUTION($\tilde{x}$) $<$ RESOLUTION(x) **do**
        $\tilde{x} \leftarrow \mathrm{SR}(\tilde{x})$ ▷ *apply SR model*
    return RESIZE($\tilde{x}$, RESOLUTION(x))
procedure RESOLUTION(x)
    return resolution of x in pixels
procedure RESIZE(x, d)
    return downsized image x with resolution d

- **Quantize**. This attack quantizes the number of states that a pixel can have. We compute quantization by the following formula for a quantization $q \in [0, 1]$.

$$\text{QUANTIZE}(x) = q \cdot \lfloor x/q \rfloor \tag{C.1}$$

  We experiment with quantization strengths $q \in [0.5, 1]$.

- **Super-Resolution**. Our Super-Resolution attack uses Latent Diffusion models [190] provided through HuggingFace[1]. The used model increases an image's resolution by a factor of 4 through an optimization process. We use Super-Resolution as a removal attack summarized by Algorithm 26. We experiment with scaling factors $\rho \in [0.125, 0.5]$.

C.2 White-box Attacks

White-box attacks assume full access to the target generator's parameters, meaning that the adversary can issue virtually unlimited queries to the target generator and can control the generator's parameters and intermediate activations.

The white-box attacks can be described as follows.

- **Overwriting**. The overwriting attack trains a decoder according to Algorithm 14 and overwrites the existing watermark using PTW described in Algorithm 13. We experiment with different weights of the watermark embedding λ_R for PTW (see Algorithm 13). Increasing the weight of the decoder's loss λ_M results in a stronger perturbation of all images synthesized by the target generator. We experiment with a weight $\lambda_M \in [0.5, 1.5]$.

[1] https://huggingface.co/CompVis/ldm-super-resolution-4x-openimages

- **Reverse Pivotal Tuning** (RPT). Our RPT attack is parameterized by the number of real, non-watermarked images R available to the adversary. We invert images in the generator's latent space by backpropagating the LPIPS loss between the currently generated image and the base image and updating the current latent code. While other methods [263, 189] to invert real images can yield better results, backpropagation is simple and works well in practice. During training, we iteratively synthesize images from a randomly sampled batch of inverted latent codes and optimize the LPIPS similarity between the generated and corresponding base images. Algorithm 15 encodes our RPT attack.

C.3 Generator Checkpoints

We experiment with the following checkpoints. All checkpoints were made publicly available by the authors [113, 114, 198]. On **FFHQ-256**, we use the following models: StyleGAN2[2], StyleGAN-XL[3], StyleGAN3[4]. On **AFHQv2** we use the following models: StyleGAN2[5], StyleGAN3[6]. On **FFHQ-1024** we use the following models: StyleGAN2[7], StyleGAN-XL[8], StyleGAN3[9]

C.4 Watermarking Parameters

All watermarks in Chapter 4 are embedded with the following parameters. We use an Adam optimizer [116] and we use a learning rate of 10^{-4} for the generator during PTW (see Algorithm 13). We use the same generator learning rate for training a watermark decoder (see Algorithm 14) and a learning rate of 10^{-3} for the watermark decoder. The watermark decoder training from Algorithm 14 contains a similarity loss and a binary cross-entropy loss for the watermark decoder. We scale the similarity loss with a weight $\lambda_{LPIPS} = 1$ and the loss for the decoder with $\lambda_M = 0.1$. We train with a batch size of 128 on FFHQ-256, a batch size of 64 on AFHQv2, and a batch size of 32 for FFHQ-1024.

[2] https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/paper-fig7c-training-set-sweeps/ffhq70k-paper256-ada.pkl

[3] https://s3.eu-central-1.amazonaws.com/avg-projects/stylegan_xl/models/ffhq256.pkl

[4] https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-t-ffhqu-256x256.pkl

[5] https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan2/versions/1/files/stylegan2-afhqv2-512x512.pkl

[6] https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-t-afhqv2-512x512.pkl

[7] https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada-pytorch/pretrained/ffhq.pkl

[8] https://s3.eu-central-1.amazonaws.com/avg-projects/stylegan_xl/models/ffhq1024.pkl

[9] https://api.ngc.nvidia.com/v2/models/nvidia/research/stylegan3/versions/1/files/stylegan3-t-ffhq-1024x1024.pkl

Appendix D

Leveraging Optimization for Adaptive Attacks against Image Watermarks

D.1 Parameters for Watermarking Methods

Tree Ring Watermark (TRW) [236]: We evaluate the Tree-Ring$_{\text{Rings}}$ method, which the authors state "delivers the best average performance while offering the model owner the flexibility of multiple different random keys". Using the author's implementation[1], we generate and verify watermarks using 20 inference steps, where we use no knowledge of the prompt during verification and keep the remaining default parameters chosen by the authors.

Watermark Diffusion Model (WDM) [261]: Instead of stamping the model's training data to embed a watermark, we apply the pre-trained encoder to a generated image as a post-processing step. We choose messages with $n = 40$ bits and use the encoder architecture proposed by [249], followed by a ResNet-50 decoder. Each call to $\textsc{KeyGen}(\hat{\theta}_G)$, where $\hat{\theta}_G$ is the surrogate generator, trains a new autoencoder from scratch.

DWT, DWT-SVD [41] and **RivaGAN** [256]. We use 32-bit messages and keep the default parameters set in the implementation used by the Stable Diffusion models[2].

D.2 Statistical Tests

Matching Bits. WDM [261], DWT, DWT-SVD [41] and RivaGAN [256] encode messages $m \in \mathcal{M}$ by bits and our goal is to verify whether message $m \in \mathcal{M}$ is present in $x \in \mathcal{X}$ using key τ.

[1] https://github.com/YuxinWenRick/tree-ring-watermark
[2] https://github.com/ShieldMnt/invisible-watermark

We extract m' from x and want to reject the following null hypothesis.

$$H_0 : m \text{ and } m' \text{ match by random chance.}$$

For a given pair of bit-strings of length n, if we denote the number of matching bits as k, the expected number of matches by random chance follows a binomial distribution with parameters n and expected value 0.5. The p-value for observing at least k matches is given by:

$$p = 1 - \text{CDF}(k-1; n, 0.5) \tag{D.1}$$

Where CDF represents the cumulative distribution function of the binomial distribution.

Matching Latents. TRW [236] leverages the forward diffusion process of the diffusion model to reverse an image x to its initial noise representation x_T. This transformation is represented by $m' = \mathcal{F}(x_T)$, where $\mathcal{F}$ denotes a Fourier transform. The authors find that reversed real images and their representations in the Fourier domain are expected to follow a Gaussian distribution. The watermark verification process aims to reject the following null hypothesis:

$$H_0 : y \text{ originates from a Gaussian distribution } N(0, \sigma_{IC}^2)$$

Here, y is a subset of m' based on a watermarking mask chosen by the provider, which determines the relevant coefficients. The test statistic, η, denotes the normalized sum-of-squares difference between the original embedded message m and the extracted message m', which can be complex-valued due to the Fourier transform. Specifically,

$$\eta = \frac{1}{\sigma^2} \sum_i |m_i - m'_i|^2 \tag{D.2}$$

And,

$$p = \Pr\left(\chi^2_{|M|,\lambda} \leq \eta \mid H_0\right) = \Phi_{\chi^2}(\eta) \tag{D.3}$$

Where Φ_{χ^2} represents the cumulative distribution function of the noncentral χ^2 distribution. We refer to [236] for more detailed descriptions of these statistical tests.

D.3 Details on GKeyGen

This section provides more details on Algorithm 16. VERIFY consists of a sub-procedure EXTRACT, which maps an image to a message using the secret key. The space of messages is specific to the watermarking method. We consider two message spaces $\mathcal{M}$: multi-bit messages and messages in the Fourier space. All surveyed methods except TRW are multi-bit watermarks, for which we use

the categorical cross-entropy to measure similarity, and for TRW, we use the mean absolute error as a similarity measure between messages (line 7 of Algorithm 16).

We instantiate GKeyGen only for the DWT and DWT-SVD watermarking methods. We train a ResNet-50 decoder θ_D in Algorithm 16 to predict a bit vector of the same length as the message and calculate gradients during training using the cross-entropy loss. Attacking TRW, WDM, and RivaGAN only requires invoking KeyGen, as the keys are the parameters of (differentiable) decoders. We train these keys from scratch for WDM and RivaGAN. For TRW, the key generation does not require any training, as the key only specifies elements in the Fourier space that encode the message. The forward diffusion process used in Verify is already differentiable.

D.4 Qualitative Analysis of Watermarking Techniques

We refer to Figure D.1 for examples of non-watermarked, watermarked, and attacked images using the attacks summarized in Table 7.1. We show three images for each of the five surveyed watermarking methods: an image without a watermark, one with a watermark, and the watermarked image after an evasion attack. We show the prompt that was used to generate these images and label each image with the p-value with which the expected message was detected in the image using the secret watermarking key and the Verify procedure.

D.5 Quality Evaluation

Table D.1 shows the FID and CLIPScore of all five surveyed watermarking methods without a watermark (first row), with a watermark (second row), after our adaptive noising attack (third row) and after our adversarial compression attack (fourth row). All results are reported as the mean value over three independent runs using three different secret watermarking keys. We observe that the degradation in FID and CLIPScores is statistically insignificant, as seen in Figure D.1.

D.6 Attack Efficiency

From a computational perspective, generating a surrogate watermarking key with methods such as RivaGAN or WDM is the most expensive operation, as it requires training a watermark encoder-decoder pair from scratch. Generating a key for these two methods takes around 4 GPU hours each on a single A100 GPU, which is still negligible considering the total training time of the diffusion model, which takes approximately 150-1000 GPU days [49]. The optimization of Adversarial noising takes less than 1 second per sample, and tuning the adversarial compressor's parameters takes less than 10 minutes on a single A100 GPU.

	TRW		WDM		DWT		DWT-SVD		RivaGAN	
	FID	CLIP	FID	CLIP	FID	CLIP	FID	CLIP	FID	CLIP
No WM	23.32	31.76	23.48	31.77	23.48	31.77	23.48	31.77	23.48	31.77
WM	24.19	31.78	23.43	31.72	23.16	32.11	23.10	32.15	22.96	31.84
A-Noise	23.67	32.15	N/A	N/A	23.55	32.46	22.89	32.50	N/A	N/A
A-Comp	24.36	31.87	23.27	32.01	23.16	32.17	23.06	31.92	23.25	31.86

Table D.1: Quality metrics before and after watermark evasion. FID⇓ represents the Fréchet Inception Distance, and CLIP⇑ represents the CLIP score, computed on 5k images from MS-COCO-2017. N/A means the attack could not evade watermark detection, and we do not report quality measures.

D.7 Double-Tail Detection

Jiang et al. [110] propose a more robust statistical test that uses two-tailed detection for multi-bit messages. The idea is to test for the presence of a watermark with message m or message $1 - m$ (all bits flipped). We implemented the double-tail detection described by Jiang et al. [110] and adjusted the statistical test in VERIFY to use double-tail detection on the same images used in Figure 7.4. Table D.2 summarizes the resulting TPR@1%FPR with single or double-tail detection. Since TRW [236] is not a multi-bit watermarking method, we omit its results.

Attack Method	**WDM**	**DWT**	**DWT-SVD**	**RivaGAN**
Adv. Noising ($\varepsilon = 1/255$)	100% / 100%	78.8% / 75.9%	100% / 100%	100% / 100%
Adv. Noising ($\varepsilon = 2/255$)	99.9% / 99.0%	79.1% / 75.0%	99.5% / 99.0%	99.9% / 99.9%
Adv. Noising ($\varepsilon = 6/255$)	68.0% / 68.7%	50.2% / 49.5%	**0.0%** / 23.3%	96.3% / 96.6%
Adv. Noising ($\varepsilon = 10/255$)	29.9% / 36.9%	**0.0%** / 57.3%	**0.0%** / 11.8%	67.0% / 63.3%
Adv. Compression	**2.0%** / **1.8%**	**0.8%** / **0.5%**	**1.9%** / **2.5%**	**6.3%** / **5.7%**

Table D.2: Summary of TPR@1%FPR using single-tail (left) and double-tail detection (right). We mark attacks bold if their TPR@1%FPR is less than 10%.

Table D.2 shows that double-tail detection increases the robustness of DWT and DWT-SVD against adversarial noising, which is the same effect that [110] find in their paper. We find that Adversarial Compression remains effective against all attacks in the presence of double-tail detection. Figure 7.4 shows that adversarial noising is highly effective against TRW but is ineffective against the remaining methods because an attacker has to add visible noise (see Figure 7.3).

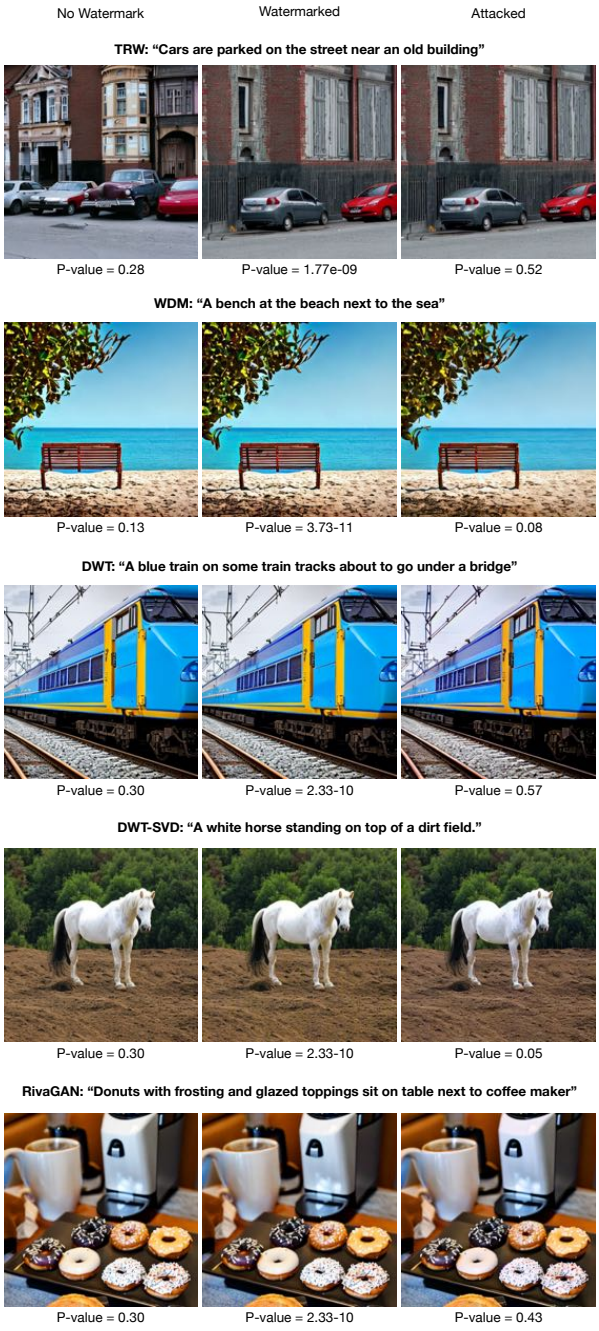


Figure D.1: Qualitative showcase of three kinds of images: non-watermarked, watermarked with mentioned technique, and attacked images with the strongest attack from Table 7.1. The p-values and text prompts are also provided.