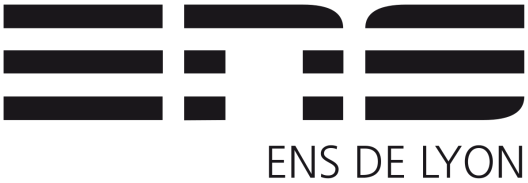

École normale supérieure de Lyon

Ioan-Tudor Cebere

M2 Informatique Fondamentale

MIAmorization: Blackbox Membership Inference Attack via Memorization

28th June 2022

Department of Computer Science

Abstract

Membership Inference Attack (MIA) is a class of attacks against machine learning models that try to leak an essential privacy property of a given sample: *Has this sample been used at training time?* Existing attacks have a high cost of running and are impractical against overly parametrized models. In this work, we explore the possibility of building an efficient MIA by exploiting unwanted memorization in subpatches of information using only inference queries to the model, highlighting an efficient privacy risk in the blackbox setting without any knowledge of the underlying model.

1 Introduction

Through large datasets availability and advances in hardware, machine learning has experienced incredible attention in both academic research and industrial applications. While some fields reap the full potential of machine learning, areas like healthcare observe remarkable but slower progress due to the lack of publicly available data. Thus, *why not use private and personal data to train a machine learning model*? An extensive line of work [RG20] has proven that machine learning models offer no privacy guarantees. They are susceptible to privacy attacks, making them incompatible with common privacy law.

Creating efficient and powerful privacy attacks against machine learning models is one approach to understanding and mitigating privacy leakages. Through them, we can validate and debug privacy claims. Privacy attacks are used to evaluate how tight privacy guarantees (e.g. Differential Privacy [DR14]) are [Nas+21], given the capabilities and assumptions for different attackers. Through them, we can quantify and compare the privacy leakage between different threat models in which the knowledge and power of an attacker vary. Consider the scenario in which we can create an adversary that violates a given bound on the privacy of a given model. In this rare case, there can be two possibilities: flawed implementation or incorrect theory [Tra+22].

One of the fundamental privacy leakages is: Was a given sample part of the training set of the targeted model or not? This is the focus of a Membership Inference Attack (MIA), which receives as input a model and a sample and predicts if the example was used by the model at training time or not. MIAs received much attention because they are considered a fundamental sign of privacy leakage. If an attacker cannot decide if a sample was present or not, it can not infer any other privacy properties against that sample.

Unfortunately, standard approaches to performing membership inference attacks involve a significant overhead. For example, the seminar work of [Sho+16] uses up to 100 retrained models, while state-of-the-art methods like [Car+21] use between 128 and 256 retrained models. The performance of these attacks is correlated with the number of retrained models. These attacks, even if they present excellent performance, cannot cover the following applications easily:

1. When testing privacy bounds [JUO20], the experiment must be rerun multiple times to ensure that the attacker was not "lucky", usually more than 1000 trials of the attack. Even if possible optimizations are described in [Car+21], we can't overcome the high costs of model retraining. The reported times of testing privacy-preserving models using MIA in [Nas+21] was 3000 GPU-hours on small models and datasets.
2. When trying to evaluate the privacy leakage of modern over parametrized models that have high training time and hardware costs. In our work, we can exploit, for example, ViT-L, a model with 300M parameters, with an estimated train cost of more than 1.000-core days on TPUv3 [Zha+21].

Retraining this model often implies very high costs, rendering the current solutions ineffective.

Contributions. Our work focuses on designing an attack that can infer membership with a running time independent of the training time of the underlying model, making it suitable for huge models and privacy testing purposes. In our work, we focus on the computer vision classification task. We take the targeted sample and remove the information directly associated with its associated label, keeping the context of that sample. Afterwards, we analyze how confident is the model while presenting subparts of both the context and the entire image to the model. Suppose we find that the model keeps relatively high confidence in the targeted label before and after removing the information associated with the label. This is one of the key components of our attack, proving label leakage in the context of a given label. We call this phenomenon memorization. Our approach is designed to unveil traces of unwanted memorization in the context of the given label only through inference capabilities. If the attacker can observe memorization with high confidence, it outputs that the sample was used at training time.

Our contribution is as follows:

1. Identify use cases in which current MIA approaches are not suitable.
2. Describe a novel approach to MIAs independent of the underlying model.
3. Discuss the difficulties of running this attack at scale and formulate future augmentations.
4. Discuss possible mitigations of the attack.
5. Identify the scenarios in which the limitations of our approach render the attack ineffective.

We have run our attack on different computer vision models trained on [Den+09], and our best results were achieved against Visual Transformers models like [Tou+21a] and [Tou+21b], with an accuracy of the attack between (62-64)%, and AUC 0.64-0.65, and a TPR@LOW FPR of (1-2.5)%.

The report is organized as follows: Section 2 formalizes the privacy problem, Section 3 describes our novel attack, Section 4 explains the evaluation methodology, Section 5 presents the empirical results of the attack, and Section 6 compares our work to existing literature.

2 Background

We begin the background by familiarizing the reader with some Machine Learning and privacy results that we target in our work.

2.1 Machine Learning

A neural network used for classification $f_\theta : \mathcal{X} \to [0,1]^n$ is a function which is parametrized by θ that maps a sample $x \in \mathcal{X}$ to an n-dimensional output, which we assume is a probability distribution generated by softmax. This function f_θ is generated through training on dataset D sampled from the underlying

data distribution $\mathcal{D}$, the training process is denoted by $\mathcal{T}(D)$. While the literature on how to train neural networks is vast, most of them rely on mini-batch Stochastic Gradient Descent [Lec+98]:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{|D_{batch}|} \sum_{(x,y) \in D_{batch}} \nabla l(f_{\theta_i}, x, y) \tag{1}$$

Where α is the learning rate, $D_{batch} \xleftarrow{\$} D$ and $\theta_0 \xleftarrow{\$} \mathcal{N}(0,1)$. We observe that the training process is stochastic, as the model training depends on the initial parameters θ_0, which are usually sampled from a normal distribution and the randomness in selecting the mini-batches. One important observation is that the model can perfectly fit the training data with enough parameter capacity.

Next, we will introduce some privacy-preserving machine learning terminology.

2.2 Privacy of Machine Learning

A machine learning system used in critical scenarios cannot leak sensitive information about the underlying individuals that participated in model training. The class of attacks that exploit the privacy aspects of machine learning models are called privacy attacks. Privacy attacks have been an emerging field of study in centralized and federated learning [McM+16]. One way of classifying the attack is by the how the attacker interacts with the model:

- **Blackbox Privacy Attacks**: The attacker has API access to the model, similar to having a model exposed in the cloud, with no knowledge of intermediary activations/model weights. Even more, in our work, we are interested in the **constrained** blackbox privacy attacks, in which we assume the attacker does not know the model architecture and its hyperparameters either.
- **Whitebox Privacy Attacks:** The attacker has access to intermediary activations and weights, similar to when the attacker can download the model and perform inferences on his hardware, being able to update or change the model weights.

Privacy attacks can also be described by the knowledge of the attacker regarding the data, namely:

- **Active Attacker:** The attacker is part of training the private model, already knowing a share of the data on which the model was trained.
- **Passive Attacker:** The attacker is an observer. He does not know any of the training data.
- **Supervised Attacker:** Knows a data sample's actual training label(s).
- **Unsupervised Attacker:** Does not know the true label of training samples.

Unless specified otherwise, we study our privacy attack in a constrained black box setup with an active and supervised attacker. Next, we are interested in introducing the membership inference game.

2.3 Blackbox Membership Inference Attacks (MIA)

The goal of a blackbox MIA is to predict if a given sample was used or not at training time by a model f_θ. In this case, f behaves like an oracle for the actual parameters θ, as the attacker does not have access to θ directly. The protocol is described in Protocol 1. If we use the assumptions in red in the protocol, we drop the **constrained** blackbox MIA setting, in which we assume that the attacker does not know how the model was trained. Another remark is that we can't have an arbitrarily large number of queries in the constrained blackbox MIA, as this could lead to a model stealing attack [Tra+16].

We observe that creating an attacker in this setup is equivalent to creating a distinguishing function $\mathbb{A} : (f_\theta, \mathcal{X}, \mathcal{T}) \to \{0, 1\}$ or $\mathbb{A} : (f_\theta, \mathcal{X}) \to \{0, 1\}$ if we work in the constrained blackbox setting. The goal is this function is to answer the question if $\mathcal{X}$ was used when creating the parameters θ behind f_θ.

The game from the adversaries' point of view in the constrained blackbox setting is depicted in Fig. 1. Next, we will discuss one of the main tools we want to exploit when creating our proposed $\mathbb{A}$, memorization in neural networks.

Protocol 1: Blackbox Membership Inference Attack protocol. Adapted from [Car+21].

Parameters: Data Distribution $\mathcal{D}$

Inputs: Challenger: train function $\mathcal{T}$.
Adversary: target model f_θ, target sample x, distinguish function $\mathbb{A}$, train function $\mathcal{T}$
Evaluator: challenger ground truth b, adversary prediction b'

Challenger Phase:

1. Sample a training dataset $D \sim \mathcal{D}$.
2. Train machine learning model $f_\theta \leftarrow \mathcal{T}(\mathcal{D})$
3. Flip a coin $b \xleftarrow{\$} \{0, 1\}$.
4. If b is 0, sample $x \sim D$. We denote this as **IN** sample.
5. If b is 1, sample $x \sim \mathcal{D} - D$. We denote this as **OUT** sample.
6. Send the attacker f_θ and x.
7. Send the evaluator b.

Attacker Phase:

1. Run distinguishing algorithm $b' \leftarrow \mathbb{A}(f_\theta, x, \mathcal{T})$
2. Send the evaluator b'.

Evaluator Phase:

1. Attacker wins if $b == b'$.
2. Challenger wins if $b \neq b'$.

2.4 Memorization

Modern over parametrized machine learning models in the supervised framework is known to overfit the training data heavily. One impactful example is made by [Zha+16] in the experimental section, where they randomly flip the labels on the train set, and the machine learning model still manages to reach 90% on the train set. While the previous example is a degenerate one to illustrate the memorization process, this scenario in which machine learning models create an internal mapping between the data and the label is empirically well studied but theoretically not fully understood yet.

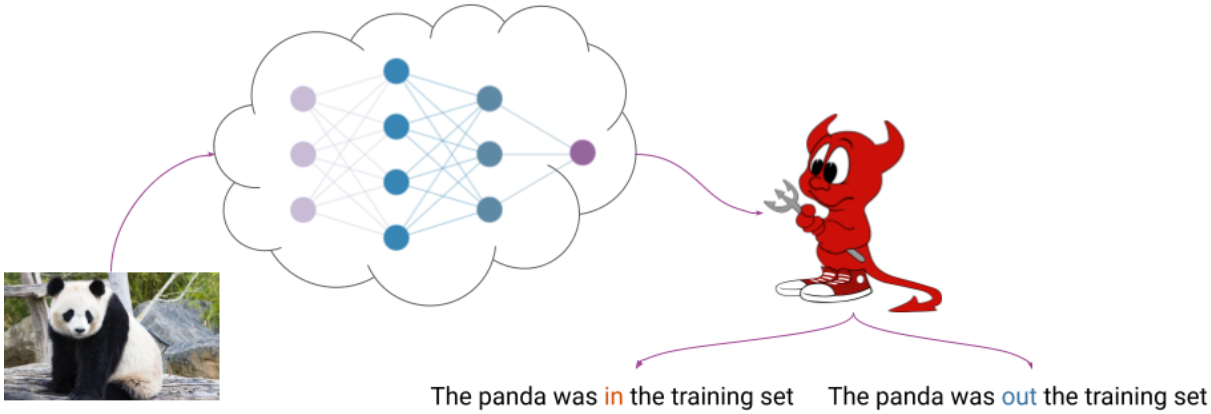

Figure 1: **The Attack setup, in a constrained black-box environment**. The attacker must leak training samples using only a public API, and without knowing the model architecture or the underlining hyperparameters.

The phenomenon when single samples are memorized partially or entirely by an algorithm is described as memorization. We underline the critical difference between overfitting and memorization: Overfitting is a global phenomenon that characterizes some properties across the entire training dataset. This is an inherent property of using gradient descent to train deep neural networks described in section 2.1 - a parametrized enough model with adequate training steps will reach 100% accuracy on the training set.

Memorization is a local effect of a model concerning a given piece of input data. An example of memorization [Car+18b] in a language model would be to feed it the following prompt: "Tudor Cebere's phone number:" and the model to answer "+40746571123". This model might still generalize well and have appealing properties for practitioners, but it could violate privacy.

Recent theoretical results have shown that memorization and overfitting on training data are needed to achieve a near-optimal f_θ [Fel19] classifier. Another important theoretical result is that memorization will happen only for the shortail of the data distribution. In other words, we can't expect all samples to present signs of memorization.

Two types of memorization are of interest - correct memorization, which helps machine learning models learn spurious features (e.g. a bird cage for the bird class) and unwanted memorization - the type of memorization that assigns random information to a given label just to fit the training data properly (e.g. the face of a fisherman and the fish label). In other words, the machine learning model will associate a part of the information with the label without actually being able to distinguish the targeted label correctly.

A natural next question would be *"How to evaluate the memorization of a given sample?"*. Existing work in natural language processing tries to use canary in data to see if they are being memorized [Car+18a]. The correct answer would be that we don't know how to measure and quantify memorization properly. Existing techniques, like influence functions from robust statistics, have not been proved fruitful when measuring strongly non-convex objective functions like those used by neural networks. Even if they would

work, they usually require additional knowledge about the data or access to the model itself, making this approach impractical for an attacker. Thus, we rely on empirically evaluating memorization given our attack vector. Our contribution is a novel way of measuring memorization. Current state-of-the-art blackbox MIAs use such a heuristic for assessing the memorization and impact of a given sample, explained in section 6, which rely on model retraining.

3 Attack Description

Algorithm 2: Gaussian 4D Distinguisher

Input: model f_θ, target sample (x_t, l_t), IN data X_{in}, OUT data X_{out}, $ncrops$, r, h

Function `CreateSample(`l_i, c_i`):`
- $\mu_l = mean(l_i)$
- $\sigma_l = std(l_i)$
- $\mu_c = mean(c_i)$
- $\sigma_c = std(c_i)$
- return $\mu_l, \mu_c, \sigma_l, \sigma_c$

Function `FitGaussian(`l, c`):`
- $\mu_l, \mu_c, \sigma_l, \sigma_c \leftarrow [], [], [], []$
- **for** $i \in |X_{in}|$ **do**
  - $\mu_l[i], \sigma_l[i], \mu_c[i], \sigma_c[i] = CreateSample(l[i,:], c[i,:])$
- $\mu_\mathcal{N} = [mean(\mu_l), mean(\mu_c), mean(\sigma_l), mean(\sigma_c)]$
- $covar_\mathcal{N} = covar(\mu_l, \mu_c, \sigma_l, \sigma_c)$
- return $\mathcal{N}(\mu_\mathcal{N}, covar_\mathcal{N})$

Function `ComputeLoss(`X`):`
- $l, c \leftarrow [], []$
- **for** $(x_i, l_i) \in X$ **do**
  - $ctx_i \leftarrow \Gamma(x_i, l_i)$
  - **for** $j \in \overline{i, ncrops}$ **do**
    - $x_i^j \xleftarrow{\$} \Lambda(x_i, r, h)$
    - $ctx_i^j \xleftarrow{\$} \Lambda(ctx_i, r, h)$
    - $l[i,j] \leftarrow \Phi(D(f_\theta(x_i) \,||\, l_i)))$
    - $c[i,j] \leftarrow \Phi(D(f_\theta(ctx_i) \,||\, l_i)))$
- return l, c

// Training phase

$in_l, in_c =$ ComputeLoss(X_{in})

$\mathcal{N}_{in} =$ FitGaussian(in_l, in_c)

$out_l, out_c =$ ComputeLoss(X_{out})

$\mathcal{N}_{out} =$ FitGaussian(out_l, out_c)

$target_l, target_c =$ ComputeLoss$([(x_t, l_t)])$

$sample =$ CreateSample$(target_l, target_c)$

return $\dfrac{p(sample|N_{in})}{p(sample|N_{out})}$

The attack relies on the property of neural networks to be able to associate random data with random

labels. We assume that samples that were memorized at training time will have different behaviour when we remove the information of the actual label of the image compared to examples that weren't seen at training time. We describe the complete sample processing on which we rely our attack in Figure 2. The image is part of the training set of Imagenet [Den+09]. We will denote this as the **IN** sample. The end-to-end algorithm is described in Algorithm 2.

We assume the existence of a function $\Gamma(x, l)$ that can identify and remove the semantic information associated with the label l from the sample x. Such a function is realistic to consider; we can imagine that the attacker can handcraft the output of this function, use a segmentation model or have an already annotated dataset for this task. This function is presented in step 1 of Figure 2.

Another property we will exploit is that we want to find specific subpatches of information that yield memorization. We empirically observed that memorization might not be immediately evident when using the complete information. We will use an augmentation technique to be able to leverage memorization traces fully. This includes randomly applying a subset of the following transformations: resize, crop, horizontal flip, and vertical flip. We experimented with other functions like perspective transformation or noise augmentation, which proved counterproductive. Another remark is that the randomness of the transforms is preserved between the image with and without the label. In other words, the same collection of transforms will be applied to both. This part of our attack feature engineering is described in step 2 of Figure 2.

Indeed, the constraint of having an active attacker is significant, as it assumes **knowledge**, which in some cases might be impractical. Nevertheless, novel applications like federated learning allow individuals to contribute their local data in computation or data crowdsourcing platforms in which users can sell their data.

We take a model f_θ pre-trained on Imagenet([Den+09]), and we analyze the probability of the removed label resulting from the inference of both samples (with and without the semantic information of the label) through f_θ. We display this step as step 3 in Figure 2, but we plot the confidence histogram only for the removed label.

We are looking for traces of memorization. The intuition for this sample is that the model had to see it beforehand, and it was possible to see it only at training time. Therefore, an attacker using this information can solve the membership inference problem. The main difficulty of running this attack is the existence of spurious features or features that help classify a given label. An example of a spurious feature for a provided label is the existence of a bird cage in photography. Such a feature will create correct and useful memorization, as it helps generalization. Our goal would be to find subpatches that yield the same high probability.

After generating the two distributions of subimages, the attacker runs inference on them, resulting in multiple n-dimensional softmax arrays. To reduce dimensionality, we compute a divergence metric between the output of the model and the ground truth. In our work, we have used cross-entropy between two discrete distributions, p and q:

$$D(p||p) = -\sum_{x \in \mathbb{X}} p(x) * \log(q(x)) \tag{2}$$

where $x \in \mathbb{X}$ are all the possible events. As an augmentation to the loss distributions, we apply a trick first observed in [Car+21]. We are interested in increasing the distance between the divergences distributions. For each divergence computed, we apply $\Phi : R- > R$:

$$\Phi(x) = \phi(e^{-x}) \tag{3}$$

Where ϕ is the logit function, the inverse of the softmax function, defined as $\phi : [0, 1] \to R$:

$$\phi(x) = \log(\frac{x}{1-x}) \tag{4}$$

The result of this augmentation can be observed in step 4 of figure 2. At this point, we have two distributions that describe the behaviour of our model when presented with partial information that contains and does not contain a targeted label. Our attack relies on taking these two distributions' mean and standard deviation. We will denote μ_l, σ_l the mean and standard deviation of the distribution of the losses that contain the label (blue distribution in Figure 2), μ_c, σ_c, the mean and standard deviation of the distribution that does not contain the label (orange distribution in Figure 2). We will denote $(\mu_l, \sigma_l, \mu_c, \sigma_c)$ sample's attack features.

On top of these features, we would like to craft our distinguishing algorithm. We assume the knowledge of both some training samples and non-training samples. We extract the features of both classes, creating **IN** and **OUT** sets for this attack. On top of each of these two sets, we fit a multivariate Gaussian, denoted $\mathcal{N}_{in}$ and $\mathcal{N}_{out}$. When deciding if a given sample was part of the training set or not, we compute it's attack features and we do likelihood testing in both of these two distributions and output $\frac{p((\mu_l,\sigma_l,\mu_c,\sigma_c)|\mathcal{N}_{in})}{p((\mu_l,\sigma_l,\mu_c,\sigma_c)|\mathcal{N}_{out})}$.

Using likelihood ratio testing is not a new approach in membership inference attacks; citelira used univariate Gaussians to distinguish between losses directly. While crafting the attack, we experimented with various distinguishers like XGBoost, Neural Networks, and Decision Trees. We have stuck with the likelihood ratio test, as it provides an interpretable result that works with insufficient data. It leaves the opportunity to develop a one-sided likelihood ratio test only based on the $\mathcal{N}_{out}$, removing the need for

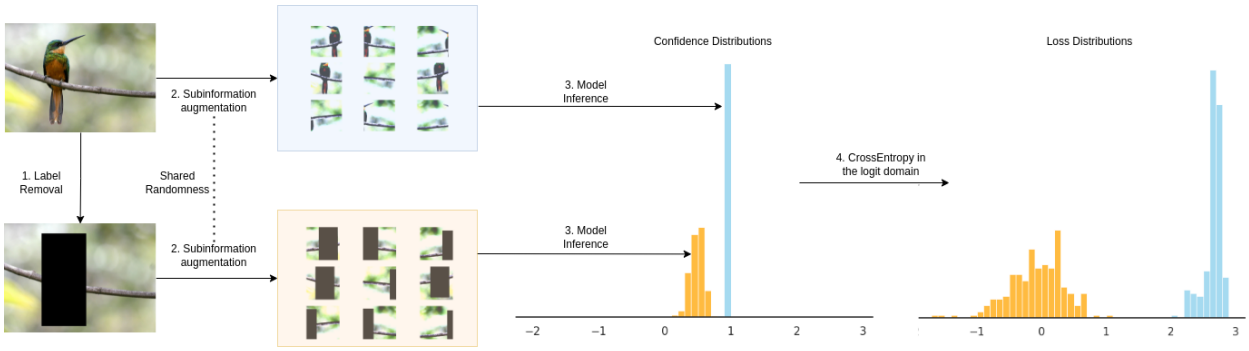

Figure 2: **Feature creation for the attack**. The attacker's feature creation step receives as an input an image and its label. It outputs two distributions that reflect the model's performance on the full image(blue) and the masked version of the image(orange).

an active attacker with knowledge of the training data. Our contribution is finding a way to correctly resume the information provided by the two distributions of losses for each sample. We have validated plenty of theories, trying to study the kurtosis of the distributions, their moments, and the number of outliers they have. Still, they provided little or no improvement over the current attack.

4 Attack Evaluation Methodology

Before diving into the details of the attack, a detailed description of what we expect from it, what we run it, and how we evaluate it is needed.

4.1 Metrics

We analyze our attack as a binary classification problem. Our attacker classifier needs to predict zero if the queried sample was an IN sample or one if it was an OUT sample. We assume an evaluation phase in which the attacker outputs a list of scores for a list of test samples, and the evaluator outputs the performance of the attacker in terms of True Positives, (TP) True Negatives (TN), False Positives (FP), False Negatives (FN).

Accuracy. Accuracy is a measure of observational error that describes how far a set of predictions are from the ground truth. Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

ROC curve. The Receiver Operator Characteristic curve solves the problem of picking a threshold on distinguishing the scores outputted by the attacker by varying the threshold. Imagine that the attacker outputs a set of monotonic scores describing the experiment's outcome. We are interested in picking the best threshold the attacker could pick to decide if a sample was IN or OUT of the training set. By doing

this, we assume that the attacker is a worst-case scenario one. Assume $f_{IN} : [0, \inf] -> \{0, 1\}$ be an oracle function that tells us if a score outputted by an attacker is part of class IN and $f_{NIN} : [0, \infty] -> \{0, 1\}$ be an oracle function that outputs if a score is not part of class IN.

$$TPR(T) = \int_T^{\infty} f_{IN}(x)\, dx \tag{6}$$

$$FPR(T) = \int_T^{\infty} f_{NIN}(x)\, dx \tag{7}$$

The ROC curve plots TPR against FPR values at different thresholds, describing the outcome of an attack independent of picking the threshold value.

AUC score. This score is used in the machine learning community to describe the probability of a given classification being correct by checking that the score outputted by the binary classifier of an IN sample is smaller than the score of an OUT sample. In other words, it measures how separable two classes are. Assuming that our attacker $f : \mathcal{X} \to [0, \infty]$, the AUC of f is computed as:

$$AUC(f) = \frac{\sum_{x \in IN} \sum_{y \in OUT} 1[f(x) < f(y)]}{|IN| x |OUT|} \tag{8}$$

TPR @ LOW FPR. As in [Car+21], we argue that even the ROC/AUC can be misleading, and we would like to know the performance of the attack with high confidence. We are interested in analyzing the True Positive Rate (TPR) at a very small False Positive Rate (FPR). Our experiment section will analyze the TPR at a 10^{-3} FPR. We find this metric important as it evaluates how reliable the attacker can identify members of the dataset.

4.2 Datasets

Our attack will rely on finding parts of a given sample that shows signs of memorization on a given pre-trained model. To validate our hypothesis, we need to find a patch that is semantically uncorrelated to the given label that regurgitates unwanted information about the label itself. To do this, we need to know exactly where the label is in the image, and we need to be able to separate the label from its context. This introduces a limitation for our experimental part, as we rely on the existence of close-to-perfect segmentations. This assumption is not unrealistic for an ideal attacker in our threat model, but it adds too much complexity when empirically studying this phenomenon.

The primary dataset on which we validate our work is Imagenet, a well-established dataset for image classification with 1000 classes, 12 million images for training, and 100k images for testing. Out of these,

we can run our attack only on those present in the ImageNet Object Localization Challenge to identify the images' patches containing the targeted label. This will heavily restrict our dataset size to 200k training samples and 50k test samples. One of the major drawbacks of our work is that the segmentations are imperfect, and for our attack to work properly, good segmentations are needed. We report the following discovered issues:

- If there were multiple instances of the same class (for example, two husky dogs), only one would be labelled/removed. This ruins our modelling of how memorization should happen.
- The objects are not perfectly segmented. Parts of the targeted label would be left out, leading to label information leakage in the context.
- The segmented label is almost as big as the image itself, leaving no context for the attacker to analyze.

Due to the previously described issues, we created separate datasets of 2000 samples of good segmentations in which we evaluate our attack, which we will denote CleanSeg Imagenet.

4.3 Models

Our attack is demonstrated on six well-established pretrained models pretrained on Imagenet. The models are standard pretrained ones, found on hubs like torchvision [Pas+19] or timm [Wig19]. We picked them with various parameters, from relatively small models like an efficient net to over-parametrized transformer models.

Model Name	**Number of Parameters**	**Top-1 Accuracy**
Densenet121 [Hua+16]	7M	75%
EfficientNet Small [TL21]	22M	83.9%
CaiT-224-Small [Tou+21b]	46M	82.7 %
EfficientNet Medium [TL21]	54M	85.1%
EfficientNet Large [TL21]	120M	85.7%
ViT-224-Large [Tou+21a]	303M	85.15%

Table 1: Description of models, their number of parameters and reported accuracy on test set.

.

When picking the ordering, we considered the number of parameters and the Top-1 accuracy on the ImageNet test set as the most important factors when deciding the models. Still, there are a lot of other key questions that have arisen which will lead to future work and investigations:

1. Has pre-training on a different dataset an impact on memorization or our attack?
2. Has initialization via Self-Supervised Learning (SSL) impacted memorization or our attack?
3. What kind of impact does regularization have on our attack?

5 Attack Evaluation

Figure 3 presents the main results of our work when evaluated on the curated segmentations ImageNet. Our attack adds an improvement between 2-7% over the baseline in accuracy and AUC. It, more importantly, provides a reliable way of obtaining a better than random TPR@LOW FPR in the range of (1-2.5)% for all attacked models. Our attacker does not perform better against highly parametrized machine learning models. Our best performance is achieved against a model based on visual transformers with a relatively small number of parameters, CaiT-S [Tou+21b]. We achieve 64% +-1% accuracy with a 0.66 +- 0.02 AUC score and a TPR of 1.9% at a small FPR. Next, we are interested in understanding how the attack compares to the baseline and how each attack component resulted in the current performances. In 4 we have displayed how the attack feature creation behaves on a few **IN** samples, the samples are picked from CleanSeg Imagenet.

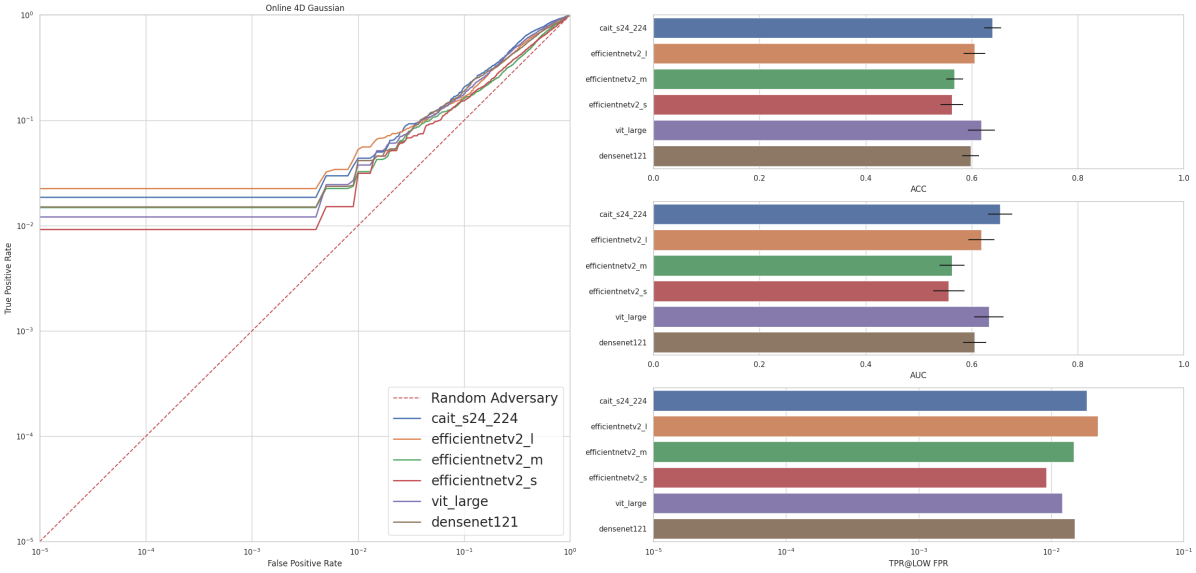

Figure 3: **Our proposed attack on CleanSeg Imagenet**. Best performances were achieved against CaiT, with an accuracy of 64%, AUC of 65% and TPR@LOW FPR of 1.8%.

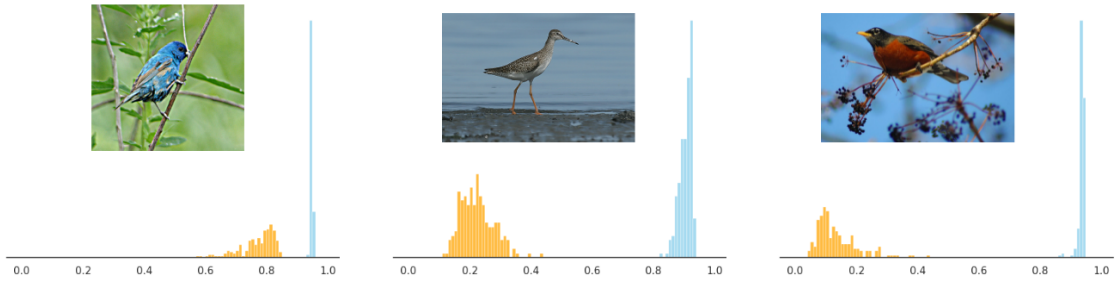

Figure 4: **IN samples probabilities for the full image and the label context.** We can observe how the context confidence distribution (orange) leaks information about label, even if removed.

5.1 Ablation study

Our attack has several components, and in our experimental section, we would like to present how they interact with each other and what the impact is on accuracy through an ablation study. As in the metrics evaluation case, we argue that AUC and ACC are essential metrics; one of the metrics we aim to improve is the TPR@LOW FPR, as we are not interested only in the worst-case scenario analysis of an attacker. All of our experiments have been run on CleanSeg Imagenet.

Baseline. The baseline evaluates the power of an adversary that tries to distinguish between IN and OUT samples based only on the confidence of the softmax output of the model on the ground-truth label of that sample. We use this baseline to understand an attacker's gain by exploiting partial information of the context. The algorithm is fully described in 3 and its performance is presented in 5. Even from the baseline, we can observe two interesting phenomena: not all models behave the same. Our top performance is achieved on the most over-parametrized model, ViT-G (0.58 ACC) - (0.60 AUC), and for the smaller models, we achieve negligible TPR@LOW FPR. We also observe the small models' poor performance at small FPR on the ROC curve as a worse metric than random guessing for the smallest models (densenet121, cait, efficientnet s, and m). This means this metric behaves worse than the random player in a low false-positive regime, making it unreliable outside the worst-case scenario. Next, we will introduce the likelihood ratio test to see its impact on the attacker. Compared to our approach, the baseline observes a drop in performance of (1-7)% in ACC and (0.02-0.07) in AUC.

Algorithm 3: Baseline

Input: model f_θ, target (x, l)

return $f_\theta(x)[l]$

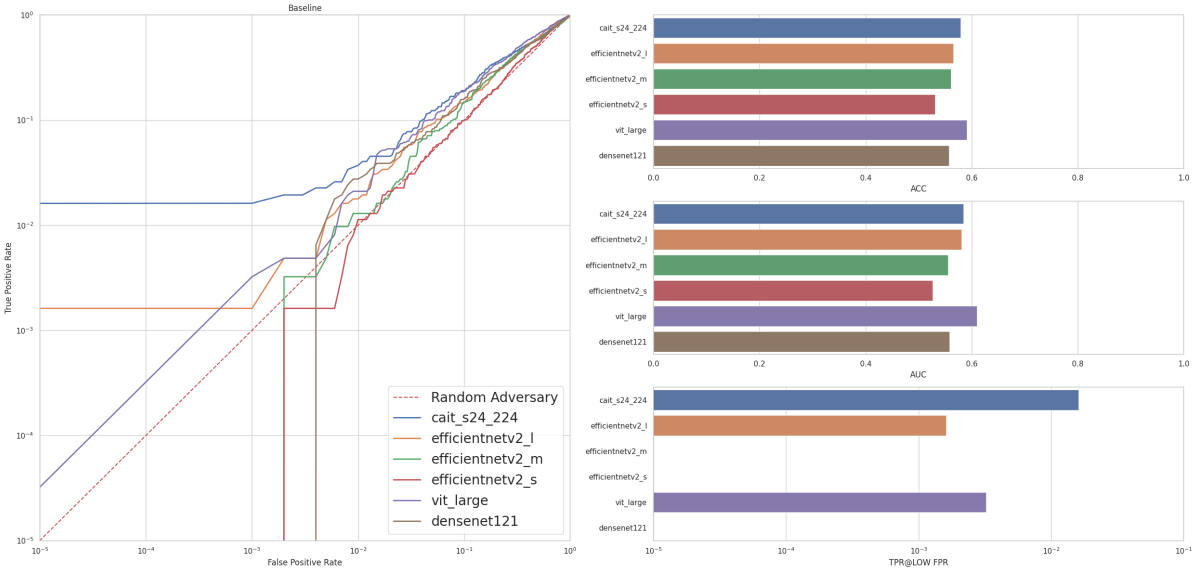

Figure 5: **Baseline Performance.** The baseline only outputs the confidence of the model on the targeted label on the full image. While it seems to have reasonable ACC/AUC, for some models the attack completely fails at TPR@LOW FPR.

Likelihood Testing. An immediate improvement over the baseline is by taking a similar approach as seen in [Car+21]. We would like to distinguish if a given loss is more similar to the IN or OUT dataset using distribution testing. Here, we introduce the active attacker assumption, in which we assume we know some data used as training time by the attacked model. Only by doing likelihood ratio testing on top of losses in the logit domain improves the overall performance of the attacker by 1-2% ACC and 0.01-0.02 AUC. More important, it can give a better than random TPR @ LOW FPR to all models, meaning that this metric is not detrimental to the attacker in the low-FPR regime.

Another argument worth discussing is why picking a Multivariate Gaussian distribution to fit and predict the four parameters $(\mu_l, \sigma_l, \mu_c, \sigma_c)$ for the likelihood ratio test in our proposed attack. We have empirically observed that on random data from ImageNet, for both IN and OUT distributions, Gaussians can be approximated with slightly different longtails, which will increase the discriminative capabilities of our attacker. Each dimension of the Multivariate Gaussian can be observed in 6. The performance of this attack is reported in figure 7.

Algorithm 4: Likelihood ratio attacker.

Input: model f_θ, target sample (x_t, l_t), IN data X_{in}, OUT data X_{out}

Function `FitGaussian(`$losses$`):`
  $\mu = mean(losses)$
  $\sigma = std(losses)$
  return $\mathcal{N}(\mu, \sigma)$
Function `ComputeLoss(`X`):`
  $losses \leftarrow []$
  for $(x_i, l_i) \in X$ **do**
    $losses[i] \leftarrow \Phi(D(f_\theta(x_i) \,||\, l_i))$
  return losses
% fit phase
$loss_{in} = \text{ComputeLoss}(X_{in})$
$\mathcal{N}_{in} = FitGaussian(loss_in)$
$loss_{out} = \text{ComputeLoss}(X_{out})$
$\mathcal{N}_{out} = \text{FitGaussian}(out_l, out_c)$
$target_loss = \text{ComputeLoss}(X_t)$
return $\dfrac{p(target_{loss}|N_{in})}{p(target_{loss}|N_{out})}$

Loss in the Logit Domain. One immediate question would be the usefulness of the attacker computing a divergence metric and projecting it into the logit domain. As described in 2, we can observe that the primary use case for this approach is to create more separable distributions. We observed that this approach also makes each of the two distributions closer to a Gaussian distribution. The attack gains (1-2%) accuracy and AUC when the loss in the logit domain is used.

Attack Performance Correlations. One of the hypotheses still under study when performing

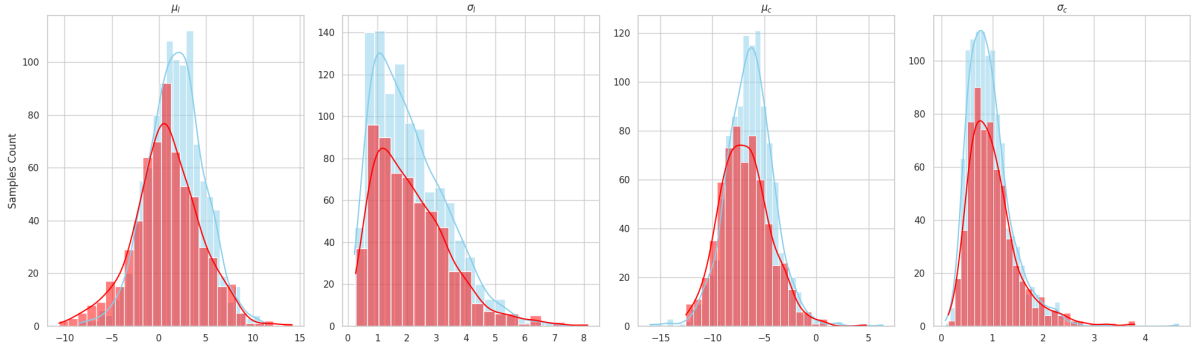

Figure 6: **Dimensions approximated by the Multivariate Gaussian.** Empirically, we observed that the four features picked for our proposed attack follow a Gaussian distribution.

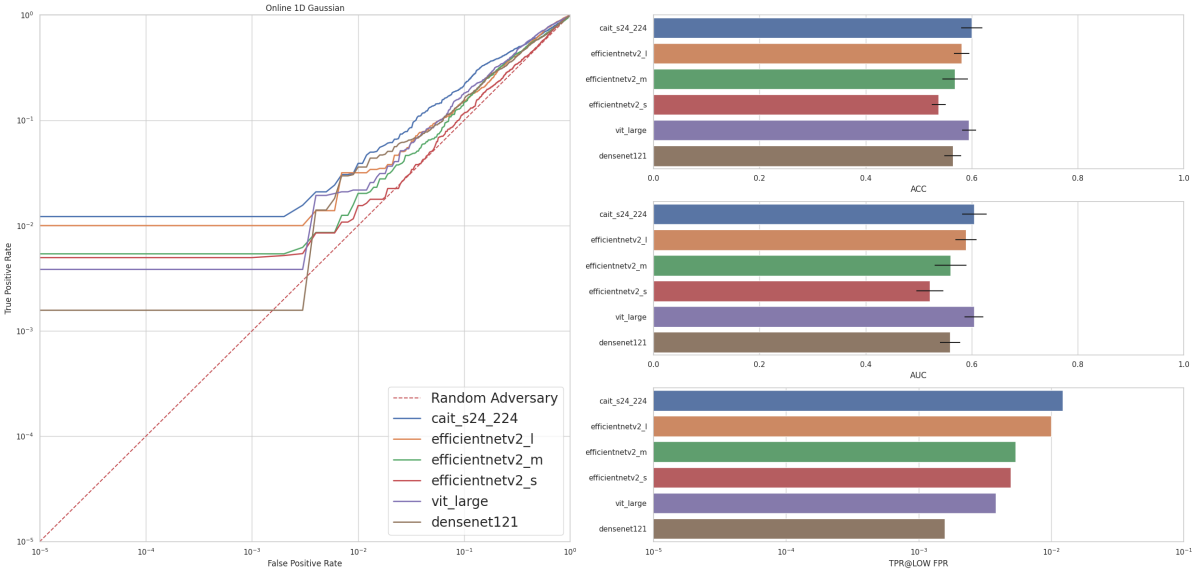

Figure 7: **Likelihood testing only attack performance.**. We can observe a strong impact on adding likelihood ratio testing to our attack compared to the baseline, at the cost of adding the active attacker assumption.

membership inference attacks is: **As models become more and more over parametrized, do they become more vulnerable against membership inference attacks?** Our attack does not provide evidence that this hypothesis is true. Our best attack is against CaiT, which is not the biggest model we tested against. Densenet121 proved vulnerable, even if it was our study's most under-parameterized model. We argue that to provide a compelling answer about why our attack delivered this performance, a deeper dive into the training methods of the models is needed. We believe that simple techniques like early-stopping or regularization can provide a massive impact, which is detrimental to our attacker.

Attack Runtime Performance. Our attack is designed to run for a dozen minutes on a CPU or GPU. The main cost of running this attack is the inference time of a given model. This attack implementation could be even more efficient through a vectorized implementation in a framework like JAX. We argue that even if our attack is not the strongest in literature, the fact that it is efficient to instantiate in practice might make it appealing for practitioners.

6 Related work

While the first instantiation of MIA was demonstrated on genomic data [Hom+08], the first MIA against a machine learning model was performed by Shokri through the shadow models approach [Sho+16]. Soon after, a connection between privacy testing and membership inference attack was formalized. This line of work produced great results (multiple citations) and approaches, which we can reexplore and reevaluate in our setting as future work. All of the mentioned approaches are based on shadow models.

Our work tries to improve black-box membership inference attacks' performance and running time with a supervised and active attacker. To our knowledge, no previous work attempts to address this problem without exploiting knowledge about the model architecture itself and retraining it with different subsets. In the following section, we will present the closest work to ours, which we want to improve directly.

Shadow Models. Most of the related work relies on the usage of shadow models [Sho+16], state-of-the-art approaches improving the running time by various optimizations. Shadow models are a class of privacy attacks in which an attacker tries to "shadow" a target model and his behavior on a target sample $x \in \mathcal{X}$ by training multiple models on various datasets sampled from the same underlying data distribution $\mathcal{D}$ and adding x to the training set in 50% of the cases. This will create two sets of models $\mho_{IN}$ and $\mho_{OUT}$. The goal of the attack is the decision if the target model f_θ behaves closer to $\mho_{IN}$ or $\mho_{OUT}$. The template for running an attack based on shadow models is described in 5. All shadow models attacks need to design a specific function β that needs to answer if model f_θ has been trained or not on sample x given the sets $\mho_{IN}$ and $\mho_{OUT}$. The original shadow model paper used a neural network for this distinguishing game; other approaches like [Car+21] used a likelihood ratio distinguisher. We mention

that the cost of running β is usually negligible compared to the cost of creating the shadow models. An expected running time of the algorithm is $\Theta(k * \overline{\mathcal{T}} + \overline{\beta})$, where $\mathcal{T}$ is the median cost of running the training function and $\overline{\beta}$ is the median cost of running the distinguishing function of the attacker.

Imagine trying to run an inference attack against an over-parametrized model like ViT-L, as we have run in the experimental section. The largest version of the model has an expected $\overline{\mathcal{T}}$ of 1.000 cores-day on TPU-v3. Beyond the running time, most researchers in privacy can't afford this hardware, not the time costs associated with running an experiment. We argue that our line of work is valuable as an alternative to testing and understanding the memorization made by over-parametrized models.

Algorithm 5: Shadow Model Attack Template

Input: model f_θ, target sample x, training function $\mathcal{T}$, number of shadow models k, data distribution $\mathcal{D}$, distinguishing function β

$\mho_{IN} = []$
$\mho_{OUT} = []$
for $i \in \overline{i, k}$ **do**
    $D \sim (\mathcal{D} - x)$
    $\mho_{OUT}[i] = \mathcal{T}(D)$
    $D' = D \cup x$
    $\mho_{OUT}[i] = \mathcal{T}(D')$
return $\beta(x, f_\theta, \mho_{IN}, \mho_{OUT})$

7 Limitations and future work

7.1 Limitations

Lack of context. Our approach of trying to extract signs of memorization from the context can be applied only when a given target label has a context. It is unclear yet how well this approach works on datasets with a very scarce or non-existent context, like brain MRI classification. We argue that creating an efficient attack for tasks that do not benefit from context information is an interesting and significantly more difficult task.

Spurious features. The proposed attacker's main difficulty is distinguishing between spurious features, the type of memorization that is beneficial for a given label and generalization, in the context and unwanted memorization, the kind of memorization that assigns random information to the provided label. We argue that this can be overcome with a carefully crafted attacker, as there is enough signal present for a human to decide only based on information coming from the sample if unwanted memorization happened or not. In the next part of this section, we argue how we tried to approach this using Self Supervised Learning.

Training data knowledge assumption. While not unrealistic, it is a strong assumption to assume that an attacker knows the training data of the underlying model. This would be a very limiting requirement

to carry out this attack, and it is worth exploring to try to remove it. In the future work part of this section, we describe how a passive attacker should be designed.

7.2 Future work

Our future work will be concentrated on trying to understand and solve the above-mentioned existing limitations. Besides the limitations, another line of work worth exploring is designing mechanisms to protect against this attack.

Quality of segmentations. Due to the quality of existing segmentations for ImageNet, we could not fully evaluate the ideal case in which the attacker has access to a proper function (insert symbol). This has an impact on understanding how strong the attack is actually. One possible approach would be to switch to a dataset in which we know almost perfect segmentation like (cite dataset), but then we should train all the mentioned models. Another approach would be to craft an artificial dataset in which canaries are inserted in the images, and we evaluate the leakage made by those canaries, similar to the approach made in [Car+22].

Passive Attacker. The assumption that the attacker can obtain training data is a strong one. Shadow models bypass this assumption by simulating their own in and out samples through retraining. One theory that needs further investigation is - how much can we leak by doing only a one-sided likelihood ratio by only outputting $P[sample|\mathcal{N}_{OUT}]$. Our initial work in this direction proved that it significantly impacted the attack, but further investigations need to be done.

Context Informed Adversary. Currently, the attacker does not use information about the image or the subpatches of the image used. There is a loss of information there, as there might be an opportunity to differentiate if it was a spurious feature for the targeted label or its unwanted memorization. We started exploring this possibility by creating embeddings of the subpatches using Self Supervised Learning (SSL), using the SimCLR [Che+20] model. The main goal of embeddings would be to evaluate how unlikely an image patch gave the full image. Embeddings generated by SSL methods are trained to create a metric between samples - closer samples in the embedding space have more similar information than embeddings far away. Using this intuition, a patch containing a spurious feature should be closer than a patch that yields unwanted memorization, as spurious features should be in the "ecosystem" of the given label.

Experiment with different training resolutions. Usually, models trained on ImageNet are rescaled to a resolution of 224x224 and fed into the neural network. More recently, models started to train on resolutions of 384x384. An analysis would be interesting to evaluate if this impacts the memorization made by the model.

Mitigation techniques. Mitigation of this attack could be made via expensive techniques in terms of utility like private learning through DP-SGD of underlying models or via simple techniques like inference rate limiters or output perturbation methods. Understanding mitigations techniques is vital to understanding how to improve our current approach.

8 Conclusion

Defining powerful attacks is critical in assessing the quality and privacy of machine learning systems. This report investigates a new way of constructing membership inference attacks against machine learning models with an active and supervised attacker. Our new attack addresses a need for researchers that study the privacy of over-parametrized models or try to efficiently evaluate empirical privacy bounds that otherwise would be blocked by computational costs. We argue that this line of work would benefit both theories on understanding more properties of memorization in over-parametrized models and practice for empirical studies that rely on running attacks efficiently.

References

[Lec+98] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[Hom+08] Nils Homer et al. "Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays". In: *PLOS Genetics* 4.8 (Aug. 2008), pp. 1–9. DOI: 10.1371/journal.pgen.1000167. URL: https://doi.org/10.1371/journal.pgen.1000167.

[Den+09] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[DR14] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: http://dx.doi.org/10.1561/0400000042.

[Hua+16] Gao Huang et al. *Densely Connected Convolutional Networks*. 2016. DOI: 10.48550/ARXIV.1608.06993. URL: https://arxiv.org/abs/1608.06993.

[McM+16] H. Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: (2016). DOI: 10.48550/ARXIV.1602.05629. URL: https://arxiv.org/abs/1602.05629.

[Sho+16] Reza Shokri et al. *Membership Inference Attacks against Machine Learning Models*. 2016. DOI: 10.48550/ARXIV.1610.05820. URL: https://arxiv.org/abs/1610.05820.

[Tra+16] Florian Tramèr et al. "Stealing Machine Learning Models via Prediction APIs". In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 601–618. ISBN: 978-1-931971-32-4. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer.

[Zha+16] Chiyuan Zhang et al. *Understanding deep learning requires rethinking generalization*. 2016. DOI: 10.48550/ARXIV.1611.03530. URL: https://arxiv.org/abs/1611.03530.

[Car+18a] Nicholas Carlini et al. *The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks*. 2018. DOI: 10.48550/ARXIV.1802.08232. URL: https://arxiv.org/abs/1802.08232.

[Car+18b] Nicholas Carlini et al. "The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets". In: *ArXiv* abs/1802.08232 (2018).

[Fel19] Vitaly Feldman. *Does Learning Require Memorization? A Short Tale about a Long Tail*. 2019. DOI: 10.48550/ARXIV.1906.05271. URL: https://arxiv.org/abs/1906.05271.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[Wig19] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[Che+20] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. DOI: 10.48550/ARXIV.2002.05709. URL: https://arxiv.org/abs/2002.05709.

[JUO20] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. *Auditing Differentially Private Machine Learning: How Private is Private SGD?* 2020. DOI: 10.48550/ARXIV.2006.07709. URL: https://arxiv.org/abs/2006.07709.

[RG20] Maria Rigaki and Sebastian Garcia. *A Survey of Privacy Attacks in Machine Learning*. 2020. DOI: 10.48550/ARXIV.2007.07646. URL: https://arxiv.org/abs/2007.07646.

[Car+21] Nicholas Carlini et al. "Membership Inference Attacks From First Principles". In: *CoRR* abs/2112.03570 (2021). arXiv: 2112.03570. URL: https://arxiv.org/abs/2112.03570.

[Nas+21] Milad Nasr et al. "Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning". In: *CoRR* abs/2101.04535 (2021). arXiv: 2101.04535. URL: https://arxiv.org/abs/2101.04535.

[TL21] Mingxing Tan and Quoc V. Le. "EfficientNetV2: Smaller Models and Faster Training". In: *CoRR* abs/2104.00298 (2021). arXiv: 2104.00298. URL: https://arxiv.org/abs/2104.00298.

[Tou+21a] Hugo Touvron et al. *Going deeper with Image Transformers*. 2021. DOI: 10.48550/ARXIV.2103.17239. URL: https://arxiv.org/abs/2103.17239.

[Tou+21b] Hugo Touvron et al. *Going deeper with Image Transformers*. 2021. DOI: 10.48550/ARXIV.2103.17239. URL: https://arxiv.org/abs/2103.17239.

[Zha+21] Xiaohua Zhai et al. *Scaling Vision Transformers*. 2021. DOI: 10.48550/ARXIV.2106.04560. URL: https://arxiv.org/abs/2106.04560.

[Car+22] Nicholas Carlini et al. *Quantifying Memorization Across Neural Language Models*. 2022. arXiv: 2202.07646 [cs.LG].

[Tra+22] Florian Tramer et al. *Debugging Differential Privacy: A Case Study for Privacy Auditing*. 2022. DOI: 10.48550/ARXIV.2202.12219. URL: https://arxiv.org/abs/2202.12219.