Anti-Neuron Watermarking: Protecting Personal Data Against Unauthorized Neural Model Training

Zihang Zou\textsuperscript{1} Boqing Gong\textsuperscript{2} Liqiang Wang\textsuperscript{1}
\textsuperscript{1}University of Central Florida \textsuperscript{2}Google Research

Abstract

In this paper, we raise up an emerging personal data protection problem where user personal data (e.g. images) could be inappropriately exploited to train deep neural network models without authorization. To solve this problem, we revisit traditional watermarking in advanced machine learning settings. By embedding a watermarking signature using specialized linear color transformation to user images, neural models will be imprinted with such a signature if training data include watermarked images. Then, a third-party verifier can verify potential unauthorized usage by inferring the watermark signature from neural models. We further explore the desired properties of watermarking and signature space for convincing verification. Through extensive experiments, we show empirically that linear color transformation is effective in protecting user’s personal images for various realistic settings. To the best of our knowledge, this is the first work to protect users’ personal data from unauthorized usage in neural network training.

1 Introduction

Recent advances on machine learning (ML) techniques have put personal data in great risks. For example, in the scandal of “Cambridge Analytica”\cite{1}, millions of users’ personal data are collected without consent to train machine learning models for political advertising. To protect personal data and privacy, legislation are put in place such as Europe General Data Protection Regulation (GDPR\cite{2}, effective in May, 2018), California Privacy Act (CCPA\cite{3}, effective in Jan, 2021), and China Data Security Law (CDSL\cite{4}, effective at Jul, 2021), to name a few. Such regulations\cite{2} require personal data shall be “processed lawfully, fairly and in a transparent manner” and can only be used “adequately, relevantly and limited to what is necessary in relation to the purposes (‘data minimisation’)”. However, even with strict legislation, the approach to verify potential personal data breach is still missed, especially regarding advanced machine learning techniques from a user’s perspective.

This paper studies personal image protection (PIP) from unauthorized usage in deep neural network (DNN) training. The need for PIP arises when users expose their images to digital products and cloud services. In the era of big data and deep learning, a critical concern is that DNN learners may violate users’ intents by using their data for training without authorization. It becomes worse when DNN models may consequently leak private user information\cite{5, 6, 7, 8}. However, how can common users know that their images, which might be a tiny portion of the training set, have been used to train a DNN?

Conventional PIP relies on digital watermarking\cite{9, 10, 11, 12, 13, 14}, which is a prevalent approach to prevent the duplication of images without permission. By imprinting special patterns, such as signatures, logos, and stamps, digital watermarking enables users to track and identify unauthorized copies of their data.

However, can the watermarking suffice the PIP against unauthorized DNN learners? The learners may only use user images to propagate gradients during training and do not release them at inference time at all. One inspiring observation is that some DNNs do “memorize” certain training examples\cite{5, 15, 6} in various ways. However, to take advantage of DNNs’ memorization ability for the PIP purpose, we have to answer three questions at least. Will the models memorize a user’s watermarked images in particular — what if they memorize others? What kinds of watermark have a better chance to be memorized? If a DNN has memorized a user’s watermarked images, how can the user verify it?

In an effort to answer those questions, we present our studies on anti-neuron watermarking for PIP against adver-
saral DNN learners. Figure 1 illustrates the problem raised by the paper. Initially, a user watermarks his/her images using a private signature before sharing the images to the public domain. An unauthorized learner then collects the user’s watermarked images, along with images from many other users, to construct a training dataset to train a DNN image classifier. As a result, given the original unwatermarked image, if we can recover the private signature from the neural classifier, then we can claim that the learner has used the watermarked images to train the classifier. Such as verification procedure can be conducted by a third-party neutral arbitrator to provide an official judgement to the user and the learner by checking whether the recovered private signature matches the user’s signature.

To achieve the above goals, we propose to use linear color transformation (LCT) as our anti-neuron watermarking. It applies to the hue space and the resulting images remain as appealing as the original ones visually, so it is difficult for the learner to detect such watermark. The images shown in Figure 1 at the user and public domain demonstrates an example of the original image and the watermarked one, respectively. Moreover, the LCT method is resilient to common image augmentation techniques used in training neural models. For watermarking, a user simply applies LCT on images with a private signature. For verification, the verifier can iterate over the signature space, to recover the signature on user images that leads to minimum loss of DNN models. This combination of the watermarking and verification is simple and yet surprisingly effective. We validate it via four datasets and five DNN architectures in various realistic settings.

We summarize our main contributions as follows. PIP is introduced against unauthorized DNN learners, which is especially important as the DNN models make their way into the digital world. We propose an anti-neuron watermarking, along with a verifier, by effectively taking advantage of the DNNs’ memorization of training data. We also conduct thorough experiments to verify the effectiveness of the LCT based anti-neuron watermarking and facilitate future research.

2 Related Work

Watermarking is an imprinting technique to declare ownership of objects for a long history. It can be traced back to paper marking at 1282 in Italy, where a watermark was created via changing the thickness of the paper. Digital watermark is later introduced by [12] to code an undetectable digital watermark on gray scale image. [13] trains a neural network for watermarking using conventional approach. [16] adds digital watermarks to video frames with neural networks. [14] proposes an automated and robust image watermarking based on deep neural networks.

Recently, watermarking are used to protect the intellectual property of machine learning models [17, 18]. This techniques work similarly as trojan attacks [19] or backdoor attack [20], where models are fine-tuned with constructed samples to learn objective behaviors. Membership inference determines if a certain sample is inside a target dataset. Inference attack was first proposed for the attacking and defense on medical datasets where users’ medical records are extremely sensitive. By comparing genomic data with the statistical information of the training dataset, the presence of certain users can be inferred by attackers [21, 8] later introduce membership inference attack (MIA) into machine learning models. Such MIA trains a binary classifier to predict membership, on top of several shadow models being trained with the same data distribution as training. Alternatively, [22] use the average of training error as the threshold to perform MIAs. [23] improve this threshold with bayes optimal classifier to search for the best threshold using samples from both training and testing.

As MIAs determine whether given data samples belong to training set or not, it is tempting to perform MIAs for personal data protection. However, as the training data distribution is unknown for common users, neither shadow models [8] nor threshold [22, 23] can be obtained and thus MIA methods fail to work in protecting personal data.

Besides, the purpose of personal data protection is to design a watermarking function, along with a verification method to protect user personal data from unauthorized usage, not defending MIAs [24, 25]. In the study [22] on membership inference, they discuss the membership advantage is not necessary if there exists features as a prior knowledge that can be used to distinguish data, (e.g. unique id for each image). This follows same idea of watermarking and verification discussed in the paper. However, [22]’s study does not study what watermarking technique can be used to against neural modeling training. They assume the all users can substitute their original data with an identifier via an arbitrary function G [26] and this substitution would not interfere the embedding identifiers. This might work if neural networks can perfectly memorize every details of training data, including the identifier. However, as shown empirically from our experiments, this is not true. Different users with the same watermarking function would interfere each other heavily, as neural models can learn the data augmentation during training [27].

Besides membership inference, attackers can infer sensitive attributes via invasion attack [6, 28]. Such kind of attack exploits intermediate features or gradients to reconstruct the inputs or private properties. One defense strategy to such threats is to obfuscate intermediate features. For example, Xiang et al. [29] conceal intermediate features by rotating the features in a complex space. Or more directly, Orekondy et al. [30] remind users the risk of exposing certain sensitive attributes via a risk prediction model. Differential privacy is a conventional privacy preserving technique that adds random noise to data to confuse attacker while keeping data distribution unchanged [31, 32]. As a result, attacker cannot infer useful information from the data. This is generic settings that are used widely, covering federated learning [33], casual learning [34], scalable systems [35] and etc. Federated learning considers the privacy issues in distributed learning settings [36, 37, 33]. Instead of collecting user data on a central server, federated learning keep user data locally to protect user privacy. By aggregating either gradients or parameters from local clients, the server gathers
knowledge from user to train the models without uploading user data. However, recent studies [38, 39, 40] reveal that this distributed design cannot prevent privacy leakage because man-in-the-middle intruder can invert user data with gradient information.

**Data anonymization** removes or replaces privacy information from original data. Wu et al. [41] degrade action video frames to hide personal information. Ren et al. [42] anonymize each person’s face in pixel-level with adversarial training. Uittenbogaard et al. [43] exploit depth and multi-view imagery to remove foreground objects from street views. Speciale et al. [44] replace 2D points features with 2D line features, which provide sufficient information for camera calibration but hide sensitive information from original images. Oh et al. [45] and Shan et al. [46] add imperceptible perturbation to original images to confuse recognizers as adversarial samples [47].

### 3 Approach

#### 3.1 Problem Definition

In this paper, we consider image classification as a case study for PIP, without loss of generality. Denote by $D_u$ the set of images owned by a common user $u$. Suppose the user $u$ exposes $D_u$ online, e.g., by sharing them on social media or backing them up to cloud. For the purpose of verifying potential breach of personal data proprietary, the user watermarks images before uploading them to the cloud. Denote by $D_u^*$ the set of watermarked images.

An unauthorized learner uses the user’s data $D_u^*$, along with many others’, to construct a training set $D$ to train a deep CNN classifier $f$ without acquiring the user’s permission. It is reasonable to assume that the user’s data $D_u^*$ is only a small portion of the whole training set $D$.

Our goal is to design a watermark, along with a neutral third party verification method $\mathcal{V}$, to verify a potential breach of a user’s personal images proprietary:

$$
\mathcal{V}(f, D_u^*, D) = \begin{cases} 
\text{True} & \text{if } D_u^* \subseteq D \\
\text{False} & \text{Otherwise}
\end{cases}
$$

where the verifier returns true if and only if the user’s watermarked images $D_u^*$ are a part of the training set $D$ for the classifier $f$.

Note that our goal is to protect user personal images, which are unique and distinguished among identifiable users, as defined in [2].

#### 3.2 The Verification Method $\mathcal{V}$

**Recovering “watermark” or “watermarked signatures”?”**

Recent studies show that DNNs can “memorize” some training examples in various ways [5, 15, 6], and one can recover certain meaningful low-resolution images from CNNs [6]. Hence, it is tempting to conduct verification by recovering the user $u$’s images from the CNN $f$. However, there are many challenges with this approach. First of all, the CNN $f$ may memorize some training images but not this user $u$’s. Moreover, even if the model happens to memorize some of this user’s images, the recovery success rate is low. Existing works [6] can recover semantically meaningful images from some CNNs, but they do not resemble any training images, to the best of our knowledge. Finally but not the least, it incurs high computation cost, often by many iterations of gradient descent, and assumes that the CNN classifier $f$ is a white box, disclosing its architecture and parameters.

Instead of using visual watermarks directly, the user can make the watermark as a secret signature that indicates a special transformation to images, and meanwhile simple for computing because the user has full control and knowledge of her/his images $D_u$ and the watermark. Hence, instead of recovering images or visual watermarks, we design a verification method $\mathcal{V}$ by inferring the watermark secret signature added to the user’s images, not the images per se.

By working as an arbitrator independent from the user and neural learner, the verifier $\mathcal{V}$ infers a secret signature from the CNN classifier $f$ and the user’s images $D_u$ without knowing the original watermark signature. If the inferred signature matches the original one well, we say that the classifier $f$ is most likely trained using the user’s images $D_u^*$.

Denote by $g_k : D_u \mapsto D_u^*$ the user $u$’s watermarking function parameterized by $k \in K$, where $K$ is the bounded space of all the possible signatures for watermarking. Suppose the user watermarks every image $I \in D_u$ by choosing the watermarking signature $k^*$, so $D_u^* = \{g_{k^*}(I), \forall I \in D_u\}$. Further, denote by $y$ the class label of image $I \in D_u$ and the watermarking function $g_{k^*}(I)$ does not change the image’s class label. Next, we recover the watermarking signature $k^*$ by the following function

$$
\hat{k} \leftarrow \arg \min_{k \in K} \sum_{(I,y) \in D_u} L(f(g_k(I), y)),
$$

where $L$ is a loss (e.g., cross-entropy) for classification.

During the verification procedure, it is important to note that we do not feed the watermarked images $D_u^*$ to the deep classifier $f$. Instead, we supply the user’s original images along with the watermarking function. $k^*$ can be considered a secret signature held by the user. If $\hat{k} \approx k^*$, i.e., the secret signature can be discovered by the verifier without knowing it before, the deep learner likely has $D_u^* = \{g_{k^*}(I), \forall I \in D_u\}$ in its training set; otherwise, the classifier $f$ should not reach the minimal loss coincidentally at the watermarking signature $k^*$ chosen by the user.
Figure 2 illustrates the above watermarking and verification processes.

3.3 Signature Space of Watermarking

In real practice, a verifier needs to quantitatively determine if \( k \approx k^* \). One common approach is to set up a threshold \( \tau \) such that if \( |k - k^*| < \tau \), we say user data has been used for training classifier. The verifier can further split the bounded signature space \( K \) into \( N \) equal size \( 2\tau \) non-intersected intervals to construct a space with \( N \) possible values. Considering one value related to no watermarking, the number of valid watermark signatures would be \( N - 1 \).

Since a user’s personal data are unique, the user’s signature does not need to be unique, thus, each user can freely choose a signature. Some readers might be curious what if there is a large number of users, for example, 1 million users but with a small \( N \), would a user’s signatures be guessed out easily? What if two users coincidentally choose the same signature? The answer to the first question is no. In reality, well-trained neural classifier could generalize well on user data distribution. If a user’s watermarked data are not used in the training, given the original unwatermarked data, the recovered signature from the well-trained classifier would approach no watermarking. As a result, the expectation of inferring user signatures correctly without learning user data would be close to zero. For the second question, recall the verification only requires a user’s personal unwatermarked data, which are distinguished from the others, thus the verification is an independent process, and the signatures can be recovered independently. However, if many users exploit the same watermarking function, recover accuracy could degrade, as shown in Figure (4). Hence, different watermarking are preferred to avoid interfere between users.

It is not necessary to have enormous \( N \) based on the above discussion. But since the classifier’s training data distribution is unknown to users, a sufficient large \( N \) is still preferred to against training data distribution being coincidentally biased to some signature instead of \( k_j \). Trade-off occurs here as users would want a larger \( N \) to ensure highly convincing verification while a large \( N \) leads to a smaller \( \tau \), which makes it more difficult to distinguish a watermark signature from nearby signatures.

One simple way to increase signature space is to divide each user data into \( M \) splits \((D_u = \{D_u^1, D_u^2, ..., D_u^M\})\) and independently apply the \( M \) signatures to these splits correspondingly. During the verification, each signature is inferred independently using one data split by eq. (2). Since there are less watermarked data for each signature, users could enlarge \( \tau \) by \( L \) times, where \( L \times \tau \) is expected to well distinguish the adjacent signatures. Under this setting, the signature space for each user could increase exponentially, \( i.e., (N/L - 1)^M \).

A large signature space also benefits memorization of user signatures. According to the study [15] on memorization, deep neural classifier must memorize atypical examples to perform well on the less frequent examples during inference. Since watermarking shift data distribution via signature from a large space, watermarking is highly likely to lift user images into lower density region and thus being better memorized by neural models.

**Computational cost of signature inference.** For a space of \( N \) signatures, the minimum computational cost to infer a user’s signature by querying is \( O(N) \), which is linear and should be acceptable for verification. In the case of data splits, the signature space expands exponentially but the computational cost remains linear as \( O(MN/L) \). Besides from querying, we can apply other optimization approaches to reduce the computational complexity. For example, we can use gradient based approach to solve eq. (2) efficiently. And the computational cost could be less than \( O(N) \) if the gradient can target the inferred signature quickly.

3.4 Properties of Watermarking Function \( g_k \)

To make anti-neuron watermarking work for eq. (2), several properties are required. First of all, the watermarking function \( g_k \) shall preserve the images’ major properties of interest. For example, for a user portrait or selfies, \( g_k \) should not change the identity of user. Besides, the watermarking function \( g_k \) shall be resilient to common image augmentations used to train deep neural models. The watermarking signature should survive against common image augmentations. Furthermore, the signature space \( K \) of the watermarking function \( g_k \) shall be bounded, such that the verifier can solve the optimization problem in eq. (2) or simply iterate all possible signatures. Another efficient and common optimization method is to use stochastic gradient descent, as an alternative. Finally, we optionally require the watermarking function invertible, which allows a user to remove the watermarks and recover the original images without losing any information of the original images.

3.5 Linear Color Transformation

Based on the properties of watermarking functions discussed above, we propose Linear Color Transformation (LCT) as an effective watermarking method to protect users’ images against unauthorized neural model training. Color-based transformations such as hue have been widely used in natural images to change color properties, which well satisfies the above requirements of watermarking function \( g_k \).

Our anti-neuron watermarking is based on hue, a natural property of color that can be represented by an angular position on a color wheel or axis on chromaticity diagram. To perform hue transformation, we can convert images from RGB color space into YIQ color space by the following matrix:

\[
T_{YIQ} = \begin{bmatrix}
0.299 & 0.587 & 0.114 \\
0.596 & -0.275 & -0.321 \\
0.212 & -0.523 & 0.311
\end{bmatrix}
\] (3)

In YIQ color space, hue is represented by two dimensional coordinates, forming a chromaticity diagram. Explicitly, a hue change by \( h \) will be conducted by rotating the color diagram around \( Y \) axis with the following rotation ma-
trix, where \( \theta = \frac{h \pi}{180} \).

\[
T_h = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix}
\] (4)

Hence, for every pixel \( v = [v_r, v_g, v_b]^T \) in image \( x \), we watermark images by changing hue through a rotation in YIQ color space:

\[
v' = T(h)v,
\] (5)

where \( T(h) = T_{YIQ} \cdot T_h \cdot T_{YIQ}^{-1} \) and \( h \) is in \([0, 360]\).

**General LCT.** Similar to hue, any LCT based watermarking can be represented by an arbitrary \( 3 \times 3 \) matrix \( T_a \) instead of the fixed hue YIQ matrix \( T_{YIQ} \), along with a rotation matrix \( T_h \). The above watermarking function \( T(h) \) would become \( T(h) = T_a \cdot T_h \cdot T_a^{-1} \), and \( T \) is an identity matrix when \( h = 0 \).

### 3.6 Personal Data Protection and General Data Protection

In this paper, we apply our anti-neuron watermarking on "Personal Data Protection" and assume each user’s images can be distinguished from the others. “Personal Data” is well defined by legislation such as [2], requiring user data shall be “relating to” identifiable natural person.

Readers might be curious what if adversary exploits the hue-transformation as watermarking and find a signature on arbitrary images that leads to minimal loss of arbitrary neural models? Can adversary accuse data privacy breach for these models? The answer is no. According to our previous analysis on signature space, the signature inferred on arbitrary images is highly likely to be no watermarking, a signature value excluded from valid watermarking signatures. On the other hand, adversary cannot claim data propriety on arbitrary data as they cannot prove the data are legally “relating to” adversarial users.

It is also worth noting that our goal is not “general data protection” but “personal data protection. It is still a challenge to verify data breach in circumstances when users data can be identical among users. We leave this challenge for future improvements.

### 4 Experiments

#### 4.1 Setup

We evaluate the proposed watermarking in image classification on Cifar-10 / Cifar-100 [48]. Tiny ImageNet datasets [49]. Our experiment includes the following processes from users, unauthorized learners and verifiers.

**Datasets.** Cifar are widely adopted datasets with 50,000 training samples, 10,000 testing samples for 10 and 100 classes, respectively. Tiny ImageNet is a selective subset of ImageNet, containing 100,000 training samples, 10,000 validation samples and 10,000 testing samples for 200 classes. Each sample is \( 3 \times 64 \times 64 \). Since testing labels are not publicly available, we report models’ validation accuracy for models’ performance. CUB-200-Bird is a high resolution \( 448 \times 448 \) fine-grained dataset containing 200 bird species, with 5994 training samples and 5794 testing samples.

**Watermarking Personal Data by Users.** A portion of randomly chosen images from training set is defined as user’s personal data. User data could contain samples from any class. Each user image is watermarked by given \( 3 \times 3 \) LCT function by a given signature (i.e., value of hue adjustment) in the space of \([30, 60, ..., 330]\), with \( \tau = 15 \) by default, followed by a pixel value clipping. A clipping is needed because LCT could cause overflow on some pixels’ values. The clipping operation in theory will break the differentiable property of watermarking function and thus hinder gradient based optimization (i.e., stochastic gradient descent). However, we find empirically that this operation does not affect signature inference. Hence, we conduct clipping after the LCT in all our experiments.

**Training Neural Classifiers Using Unauthorized Data by Learners.** Unauthorized learners train neural classifiers using above watermarked user data along with other training data. Images are converted from \([0, 255]\) into \([0, 1]\), randomly cropped, randomly horizontal flipped and normalized following common practice [50, 51]. Training data would be randomly shuffled and trained in mini-batches. All classifiers are trained from scratch for 90 epochs. The initial learning rate is 0.1 and decays 0.1 for every 30 epochs. In Cifar experiments, models’ architectures are slightly adjusted [52] to fit in small resolution.

**Inferring Watermark Signature by Verifiers.** Given suspicious neural network models, verifiers determine whether the models were trained using watermarked data following eq. (2). During signature inference, each image is converted from \([0, 255]\) into \([0, 1]\), then watermarked and normalized. Given clean (unwatermarked) images from users, watermark signature can be recovered using one of the following two approaches. 1) **Grid search approach:** verifiers can enumerate all signatures and add them to clean images, then perform grid search on every \( 2\tau \) degree to recover the signature. If the signature is well memorized by the classifier models, loss will reach minimum when the current enumerated signature equals or closely approximates to the signature used by the user. We iterate all the possible signatures in our grid search experiments. The signatures are generated by dividing the whole signature space into \( N \times 2 \tau \) intervals. 2) **Gradient search approach:** when the model is accessible, verifier can watermark clean images with a random initial watermark signature, and then use gradient search to infer the signature by descending the gradient \((\nabla_k L)\) of signature. This technique infers watermark signature more precisely than grid search. For the settings, unwatermarked user data are combined into one mini-batch and stochastic gradient descent is used for optimization over signatures. The initial learning rate is 0.1 and decays 0.1 each 100 epochs, with 300 epochs in total. To avoid local minima, we select initial values from all possible signatures and report the signature that lead to minimal loss. Comparing to grid search, this optimization approach is much more computational expensive and thus our results will be mainly on grid search for simplicity.

**Source Code.** Our source code is publicly available
at https://github.com/zzzucf/anti-neuron-watermarking. All our experiments are implemented by Pytorch. Readers can freely explore our proposed watermarking approach with the code supplied.

4.2 Effectiveness of Watermarking

In the following sections, by default we consider a single user watermarks his/her data with a signature and we show empirically under what circumstances this signature can be memorized and inferred. The experiments on multiple users are presented later in Section 4.3.

Different Quantity of Watermarked Samples. We first study how many data are desired for watermarking. Different quantity of samples are watermarked in this experiment. We adopt ResNet50 [50] and use 60 (hue adjustment) for signature. The grid search result is shown in Figure 3 (a, b, c) using eq. (2) and it’s visually clear that most of the models achieve minimum loss nearby the watermark signature, within the range of matching $60 \pm 15$ ($\text{signature} \pm \tau$). However, with less watermarked data (i.e., less than 5 samples), inferring signature misses as other signatures achieve minimum loss. We also show gradient descent result for Tiny ImageNet in Table 1. Being trained by classifier, user’s watermarked images achieve lower loss than the clean images. And watermarking does not affect classification performance for getting similar testing accuracy. These results show that given sufficient data, neural classifier could memorize watermark signature on user’s data pretty well.

| # of Data | Model Acc | Watermark Loss | Clean Loss | Inferred signature |
|----------|-----------|----------------|------------|--------------------|
| 10,000   | 55.6      | 0.019          | 0.161      | 59.0 ✓             |
| 1,000    | 55.8      | 0.016          | 0.541      | 56.4 ✓             |
| 100      | 54.9      | 0.019          | 0.668      | 59.5 ✓             |
| 10       | 54.5      | 0.017          | 0.329      | 48.9 ✓             |
| 5        | 54.6      | 0.001          | 0.397      | 60.99 ✓            |
| 1        | 55.9      | 0.004          | 0.006      | 17.0 ×             |

Table 1. Inferred signatures for models trained with different quantity of watermarked data on Tiny ImageNet. The watermark signature is 60 with $\tau = 15$.

Different Watermark Signatures. We verify whether different watermark signatures work equivalently. We randomly select 100 as user data for Tiny ImageNet and apply different signatures for watermarking. We present the grid search result for different signatures on Tiny ImageNet in Table 2. The result shows that the watermarking for single user does not affect models’ training as models’ accuracy are similar for different signatures. User watermarked images achieve lower average loss than original images, indicating unauthorized training models can memorize watermarked images in “some way”. And we achieve minimal loss nearby watermarking signature, bounded by predefined threshold $\tau = 15$. Besides, we show the grid search results in Figure 3 (d, e, f) for different datasets. From these figures, we observe the all inferred signature (marked in square) match original watermark signatures, so we can conclude that our approach is general to different data distribution.

| Watermark signature | Model Acc | Watermark Loss | Clean Loss | Inferred signature |
|---------------------|-----------|----------------|------------|--------------------|
| 60                  | 54.9      | 0.019          | 0.668      | 59.5 ✓             |
| 120                 | 55.6      | 0.070          | 0.895      | 120.2 ✓            |
| 180                 | 53.8      | 0.030          | 1.189      | 178.8 ✓            |
| 240                 | 56.3      | 0.025          | 1.118      | 242.1 ✓            |
| 300                 | 55.7      | 0.016          | 0.839      | 306.1 ✓            |

Table 2. Watermarking for different signatures for ResNet50 on Tiny ImageNet.

Different Neural Classifier Architectures. We also evaluate the proposed watermarking for different neural classifier architectures, including Alexnet [51], VGG [53], ResNet [50], Wide ResNet [54] and DenseNet [55]. We use the same settings of watermarking (60 is the watermark signature) for different architectures. As shown in Figure 3g, all inferred signatures match, implying that our watermarking approach works well against a large variants of DNN models. We also present the gradient search result for different architectures on Tiny ImageNet in Table 3.

| Architecture | Model Acc | Watermark Loss | Clean Loss | Inferred signature |
|--------------|-----------|----------------|------------|--------------------|
| Alex         | 38.0      | 2.189          | 3.175      | 56.9 ✓             |
| VGG          | 57.3      | 0.640          | 1.151      | 52.8 ✓             |
| Res          | 54.9      | 0.019          | 0.668      | 59.5 ✓             |
| Wide Res     | 56.6      | 0.005          | 0.738      | 58.4 ✓             |
| Dense        | 61.5      | 0.114          | 0.838      | 58.8 ✓             |

Table 3. Watermarking for different architectures on Tiny ImageNet.

Different Learning Capacity of Models. We further investigate whether model memorizes watermark signatures better when model has more learning capacity (e.g. more parameters, deeper or wider) by exploring the ResNet family. As shown in Figure 3h and Table 4, we can observe that as networks go larger and deeper, the loss decreases faster and reaches minimum at the watermark signature.

| Architecture | Model Acc | Watermark Loss | Clean Loss | Inferred signature |
|--------------|-----------|----------------|------------|--------------------|
| ResNet18     | 52.9      | 0.036          | 0.801      | 55.2 ✓             |
| ResNet34     | 53.6      | 0.007          | 0.679      | 55.8 ✓             |
| ResNet50     | 54.9      | 0.019          | 0.668      | 59.5 ✓             |
| ResNet101    | 54.6      | 0.004          | 0.736      | 58.3 ✓             |
| ResNet152    | 56.5      | 0.007          | 0.814      | 60.4 ✓             |

Table 4. Watermarking for different learning capacities on Tiny ImageNet.
Figure 3. The first row shows variant of loss for models trained with different quantity of watermarked samples on Cifar and Tiny ImageNet. The second row shows variant of loss for models with different signatures on Cifar and Tiny ImageNet. The third row shows results for different architectures different capacity on Tiny ImageNet and high resolution on CUB-200-Birds. Square marker indicates inferred signature (the point reaches minimum loss).

High Resolution Image. In Figure 3i, we present our result on CUB-200-Birds, fine-grained dataset with a high resolution of 448×448. We use pretrained ResNet50 models from ImageNet and conduct a transfer learning on CUB-200-Birds. Since the dataset only have less than 6000 images for training, we assume user have 60 images (1%) for watermarking. High intensity data augmentations [56] are being used to boost performance, including color jitter, random crop, random resize, random scale and random horizontal flip. Even under strong data augmentation, the result shows that it could memorize user’s watermark during training even if models are pretrained on other large scale datasets.

4.3 Signature Space Analysis of Watermarking

In this section, we show experiments by analyzing signature space of watermarking following Sec 3.3.

When Data Were Not Used to Train Neural Classifier. From eq. (2) and previous experiments, we empirically show that the inferred signature would match watermark signature if watermarked data have been used to train neural classifier. Here we show the inferred signature would approach no watermarking when watermarked data were not used for training. To this end, we use auxiliary unseen data from validation for inference, which have the same data distribution as training but have not been used for training. Pretrained models from Figure 3f are used to produce Table 5. Empirically, inferred signatures always approach 0 (i.e., identity transformation under eq. (4)) for untrained data. Considering the data is different and watermarking is always being used before exposure, different inferred signature value can distinguish models training with and without watermarked data.

| Watermark | 60 | 120 | 180 | 240 | 300 |
|-----------|----|-----|-----|-----|-----|
| Trained   | 59.5 | 120.2 | 178.8 | 242.1 | 306.1 |
| Untrained | 6.8  | 3.8  | 4.3  | 357.6 | 4.2  |

Table 5. Inferred watermark signatures would be different for trained vs untrained data. Notice that 357.6 is equivalent to -2.4 in angle expression and it is close to zero.

Increase Signature Space with Data Splits. We repeat experiment in Figure 3f by dividing 100 data of user into 10 splits and assign random signature to each split. The \( \tau \) is enlarged from 15 to 30. As a result, the signature space
of single user increases from 11 to $5^{10}$. From Table 6, we can observe that all the inferred signatures match watermark signature given $\tau = 30$ even with only 10 samples for each signature.

| Watermark | Inferred | Watermark | Inferred |
|-----------|----------|-----------|----------|
|           | 120      | 120       | 121.6    |
|           | 240      | 237.3     | 239.8    |
|           | 120      | 125.8     | 178.9    |
|           | 180      | 183.8     | 134.8    |
|           | 60       | 180       | 249.1    |

Table 6. Increase signature space with data splits.

Multiple Users with LCT-based Watermarking. Previous experiments are set with a single user, here we further explore multiple users under several settings. The training set of Tiny ImageNet is divided into 1,000 users equally. For the same LCT, we use hue as in eq. (3). For different LCTs, we sample $3 \times 3$ matrix from uniform distribution $T_a \sim \mathcal{U}(-1, 1)$ (uniform distribution) per user. Each user chooses a random signature from $\{30, 60, ..., 330\}$ and $\tau$ is set to 15. We report total matches on the average of 5 independent experiments.

From the result shown in Figure 4, we can observe that when 20% of users are exploiting watermarking, watermark signatures can be inferred correctly for almost all the users. As this ratio increases, the matching accuracy drops significantly if users use the same LCT. Meanwhile, if users use different LCTs, the matching accuracy remains above 80% even all users data are watermarked, being independent among users. We also evaluate a special case when adversary infer signatures using arbitrary LCT. Arbitrary LCTs ($T_{a'} \sim \mathcal{U}(-1, 1)$) only achieves 10% matching accuracy, which is 70% less when LCTs are given. Such results indicate users can use unique and user-specific watermarking for a better protection when other users also exploit watermarking.

![Figure 4. Matching accuracy for multiple users using watermarking.](image)

4.4 Properties Analysis of Watermarking

In the following sections, we analyze how different properties discussed in Section 3.4 affect watermarking.

**Resilience to Data Augmentation.** We evaluate our watermarking effectiveness against a variety of common data augmentations, especially applying the same hue transformation in training. We evaluate several widely adopted data augmentations including cutout [57], label smoothing [58], Gaussian noise [59], adversarial training [60] and differential privacy [32]. For the hue transformation for watermarking, we test color jitter [51], which include brightness, saturation, contrast and the same hue transformation we used for watermarking.

- **Random Crop** [51]: random crop is a widely used data augmentation for most neural models’ training [51, 53, 50, 56]. It randomly crops images into smaller resolution to reduce models’ overfitting on spatial location. For Tiny ImageNet, images are randomly cropped into $64 \times 64$ with a padding of 8. For Cifar, images are randomly cropped into $32 \times 32$ with a padding of 4. All our training includes random crop for best performance.

- **Horizontal Flipping** [51]: this is also a widely used data augmentation for image classification. We include horizontal flipping for all our experiments.

- **Cut Out** [57]: cutout removes random region of size $M \times M$ from images at each training iteration. We set the size $M = 8$ for our experiments. The experiment result is shown in Figure 5a.

- **Label Smoothing** [58]: label smoothing is also a widely used data augmentation in many tasks [61]. It reduces the probability of ground truth label (e.g., 100% cat, 0% dog, 0% duck) by a smoothing parameter $\alpha$ and assigns probability uniformly to other classes (90% cat, 5% dog, 5% duck). The smoothing parameter $\alpha$ is set to be 0.1 in our experiments. The result is shown in Figure 5b.

- **Gaussian Noise** [59]: this technique simply add noise to the input from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$ to increase models’ robustness. The $\sigma^2$ is set to be 0.1 in our experiments and the result is shown in Figure 5c.

- **Adversarial Training** [60]: Neural Networks is known be to vulnerable to adversarial attacks. [47, 60], and adversarial training are believed to reduce overfitting [60] and mitigate privacy leakage [24, 25]. We address this by training with adversarial samples generated from FGSM attack [47]. The epsilon is set to be 0.01 and the result is shown in Figure 5d.

- **Differential Privacy** [32]: differential privacy is a wide adopted privacy preserving technique in many real life application. By adding noise to the query results, user’s sensitive information cannot be inferred via querying. In our implementation, we add random noise sample from Gaussian distribution $\mathcal{N}(0, \sigma^2)$ to the output confidence. The $\sigma^2$ is set to be 0.1 and the result is in Figure 5e.

- **Color Jitter** [51]: color jitter randomly adjusts brightness, contrast, saturation and hue of input images. We apply high intensity color augmentation in our experiments. For each color properties, the value of adjustment is randomly sampled from [-288, 288], covering 80% of the range of transformation. In Figure 5f, we can see the
Figure 5. The variation of model loss for different data augmentations. Only the color jitter can significantly narrow down the loss difference between signatures.

Table 7. Watermarking for model trained with color jitter augmentation for ResNet50 on Tiny ImageNet. Loss difference between clean and watermarked samples are smaller comparing with Table 2.

| Watermark signature | Model Acc | Watermark Loss | Clean Loss | Inferred signature |
|---------------------|-----------|----------------|------------|--------------------|
| 60                  | 52.5      | 0.042          | 0.054      | 64.8 ✓             |
| 120                 | 52.5      | 0.102          | 0.244      | 108.3 ✓            |
| 180                 | 53.0      | 0.052          | 0.081      | 178.8 ✓            |
| 240                 | 53.0      | 0.113          | 0.132      | 254.4 ✓            |
| 300                 | 51.7      | 0.125          | 0.161      | 291.3 ✓            |

In Table 8, watermark signatures can be inferred correctly for the aforementioned data augmentations. This shows empirically that LCT is an effective watermarking approach because it is resilient to common data augmentation in neural networks’ training. Besides, we evaluate privacy preserving techniques such as differential privacy. Since we infer signature using all user images, noise added to the output would be removed by taking average.

Table 8. Inferred signatures for models trained with different data augmentations for ResNet50 on Tiny ImageNet.

| AUGMENTATION       | 60  | 120 | 180 | 240 | 300 |
|--------------------|-----|-----|-----|-----|-----|
| Cut Out            | 57.5| 122.1| 178.6| 240.3| 301.8|
| Label Smoothing    | 59.6| 116.9| 181.4| 239.3| 299.9|
| Gaussian Noise     | 58.3| 107.8| 187.1| 237.2| 309.9|
| Adv Training       | 57.5| 118.8| 183.5| 242.9| 298.6|
| Differential Privacy| 56.0| 117.5| 182.8| 240.6| 295.9|
| Color Jitter       | 64.8| 108.3| 178.8| 254.4| 291.3|

Less noticeable Watermarking. In the previous sections, we show that LCT based watermarking is effective against unauthorized neural learners, but may change the color property significantly in visualization, as illustrated in Figure 6. One may argue that such a kind of watermarking could be too visually obvious to be recognized by unauthorized neural learners. However, given data samples, by selecting proper watermarking signature, the watermarking could be difficult to be distinguished from stylish transformation or even unnoticeable to human being.

For example, nowadays users would often apply image filters to stylize images before publishing on social media, where filters are quite natural such as tuning the color of tree leaves from green to yellow, changing the color of sky from light blue to dark blue. The parameters of these images’ filters could be used as signature for watermarking. As stylized filters are widely used, it would be difficult for neural learners to distinguish whether it is watermarking or users’ preference.

Color-based transformation can be less noticeable when it is only applied on selective pixels and color channels. In particular, by following [13], we first generate a random binary...
string \( w \) with a fixed length \( T \), and then generate pseudo-random pixels’ positions as \( \rho_t = (i_t, j_t) \) for each element \( w_t, (1 \leq t \leq T) \). Finally we change blue color channel for these pixels as:

\[
B_{\rho_t} \leftarrow (2w_t - 1)\alpha L_{\rho_t},
\]

(6)

where \( \alpha \) is the hyper-parameter of watermarking intensity and \( L_{\rho_t} \) is luminance of pixel calculated by \( L_{\rho_t} = 0.299R_{\rho_t} + 0.587G_{\rho_t} + 0.114B_{\rho_t} \).

Different from [13], we use \( \alpha \) as the watermark key and pass the pseudo-random pixels’ locations and binary string \( w \) to the verifier for key inference. Empirically, this kind of watermarking can be memorized by neural learners but less noticeable to human. We present visual comparison of samples in Figure 7 with different value of \( \alpha \). As shown in Figure 7, the general appearance of watermarking remains visually unnoticeable. For \( \tau = 0.1 \) and watermark signatures \( 0.10, 0.30, 0.50, 0.12, 0.28, 0.44 \) are inferred, matching the watermark signatures. However, such kind of watermarking has its limits. It introduces noise to the images and the images can look noisy when the color theme are dominated by red or green. To solve this problem, one has to find some transformation that is not visible to human but easy to learn by neural classifiers. We leave this challenge as future improvements.

### 4.5 Memorization Analysis of Watermarking

We further explore empirically why watermarking is effective against neural classifier. We first compare our approach with two MIAs. Then we show watermarked data can be easier to be memorized than their original copies. And last we show when the watermarking signature were memorized during training.

**Watermarking Protects Users’ Data from Membership Inference Attacks.** To the best of our knowledge, this paper is the first work to protect user personal data from unauthorized neural model training, and there exists no comparable former approaches. One most related approach would be MIAs. However, MIAs require the prior knowledge of training data distribution that is not applicable in PIP. Here, we show that watermarking could hinder MIAs as a defense way with two black-box MIAs: \( \text{Adv}_{\text{std}} \) [22] and \( \text{Adv}_{\text{pow}} \) [23]. We following [62] experiment settings for these two attacks. \( \text{Adv}_{\text{std}} \) [22] only require \( N \) samples from training to calculate the average training loss as a threshold to predict given data membership. And \( \text{Adv}_{\text{pow}} \) [23] searches for the best threshold using \( N \) training samples and extra \( N \) testing samples. The \( N \) is set to be 50 for our experiments. These two thresholds based MIAs can be applied to predict membership when a few training or testing samples of models are known for the verifier.

We perform MIAs on Cifar-100 and Tiny ImageNet pre-trained models. The MIAs’ success rate is calculated by classifying membership using the above thresholds on testing set, with half of data from model’s training and half from testing. Since our training data contains watermarked data, we construct two testing sets respectively. For the training half, one takes non-watermarking data and the other takes watermarked data. The threshold is obtained from non-watermarking training data as the prior knowledge. As shown in Table 9, success rate decreases significantly when predicting membership for watermarked data. This is because watermarking lifts user data to rarer samples and lead
to lower confidence. As a result, the threshold obtained from training distribution would fail to work in classifying watermarked data.

|                        | Cifar-100 | Tiny ImageNet |
|------------------------|-----------|---------------|
| Accuracy (Train/Test)  | 99.6% / 74.0% | 100% / 53.0% |
| AdvPred (W/N)          | 50.5% ↓ / 58.8% | 55.0% ↓ / 61.0% |
| AdvPred (W/N)          | 59.0% ↓ / 65.8% | 70.0% ↓ / 71.0% |

Table 9. MIA success rate on watermarked (W) and non-watermarked (N) data.

Watermarking Improves Memorization of User Data.

“Memorization value estimate” (MAE) [15] measures the generalization gap (difference of predicted probability of ground truth labels between models trained with certain data ($P_{in}$) and models not trained with certain data ($P_{out}$)) to quantify the memorization ability of neural networks toward such data. If the MAE becomes higher after watermarking, it indicates the model tends to memorize watermarked data than original data.

Specifically, for training algorithm $A$ on a dataset $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, the amount of label memorization by $A$ on example $(x_i, y_i) \in S$ is defined as [15],

$$\text{mem}(A, S, i) := \left| \Pr_{h \leftarrow \mathcal{A}(S)}[h(x_i) = y_i] - \Pr_{h \leftarrow \mathcal{A}(S \setminus i)}[h(x_i) = y_i] \right|. \quad (7)$$

To estimate above memorization value on sample index at $i$, [15] firstly selects random subsets for $S$, with some subsets including sample $i$ and some exclude sample $i$. Then, $K$ models are being trained using these subsets, grouped into 2, one includes sample $i$ and the other excludes sample $i$. The memorization value estimate (MAE) is finally calculate by averaging the difference of $P_r$ between these two groups of models. In our experiments, we split the training set of Tiny ImageNet into 1,000 users and choose one split as user data (we calculate MAE on a collection user data instead of one sample). Then we randomly use 70% of users data to construct 20 subsets. 20 models are being trained correspondingly to calculate the final MAE. We fix the indices for this experiments and train 20 models with and without watermarking on user data. As shown in Table 10, MAE increases by 8% after applying watermarking, indicating that the watermarking increases memorization of user data.

|                | $P_{in}$ | $P_{out}$ | MAE  |
|----------------|----------|-----------|------|
| No Watermarking | 75.6%    | 48.8%     | 26.8%|
| Watermarking    | 58.4%    | 23.5%     | 34.8% (8.0% ↑) |

Table 10. MAE of user data increases after watermarking.

When Signature is being Memorized during Neural Model Training. One interesting problem for watermarking is when a signature is memorized by neural models.

From previous studies on MIAs [8, 22, 62, 63], user privacy information is being leak when model’s overfitting. As a result, it is likely for models to memorize watermarking signatures when models over-learn the watermarked data [63]. To explore the answer of this question, we infer signatures by grid search for different checkpoints of model during training. In Figure 8, we can observe the inferred signature reaches 0 at early stage of training and gradually reach 60 with more learning epochs. This result illustrates that the watermarking signature could be memorized before the end of training. This empirical results can also explain why data augmentation (watermarking) can be learned by neural models during training, according to the study [27] for neural generalization.

![Figure 8. The watermarking signature can be memorized before the end of training. The red point indicates inferred signature that achieves minimal loss over other signatures.](image)

4.6 General Watermarking

In our paper, we mainly explore LCT as an effective way for anti-neuron watermarking. Besides from LCT, we also evaluate a traditional verification approach by recovering watermark pattern and show how it fails. We further explore a geometrical watermarking that can be applicable. Certainly, there are other watermarking functions that can verify unauthorized neural model training and we leave this open question for future research.

Verification by Recovering Watermarking Pattern. As we briefly discuss in Section 3.2, we investigate a traditional watermarking technique by appending a special pattern (e.g., a sticker) on images. We evaluate this by training a ResNet50 on 100 randomly selective user samples with a cat pattern in Tiny ImageNet. Similar to model inversion [6], we use a learnable Gaussian variable on users’ images and minimize the classification loss to reconstruct watermark pattern with this variable.

As shown in Figure 9, although the reconstructed pattern can achieve 100% accuracy and minimal classification loss,
neural models cannot memorize such watermark pattern, as no meaningful pattern can be recovered. We believe that this could be caused by convolution operation where all spatial information of pixels are being ignored during training. This experiment indicates that it is difficult for neural learners to memorize such kinds of watermarks, comparing to the watermarks using color-based transformation.

**Geometrical Watermarking.** Besides color-based transformation watermarking, we also investigate the effectiveness of geometrical (e.g., rectangle mask) watermarking. Specifically, we can change the intensity of certain area by multiplying a rectangle mask on certain rectangle area of image \(x\) as:

\[
g_v(x) = x - x \cdot \text{mask} \cdot (1 - v),
\]

where mask is a rectangle binary masking located at \((a,b)\) (top left corner) with a size \(w \times h\), and \(v\) is the intensity of watermarking. Note that, here, the user secrets keys consist of different \(v\), different diagonal location as \((a,a)\), and different ratio of \(w/h\).

We evaluate this watermarking on randomly selected 100 samples from Tiny ImageNet. Results on different mask intensity values are reported in Figure 10a. As we can see, inferred keys match watermark keys for all the value selections. In Figure 10b, we apply rectangle watermarks on different diagonal position \((i,i)\), \(0 \leq i \leq 64\). The results illustrate that the watermark can be recovered. In Figure 10c, we fix the top left corner of the rectangle mask and change ratio between width and height. The results also illustrate that extract watermark achieve minimum loss when approaching watermark parameter.

**5 Conclusion**

In this paper, we introduce a new personal data protection problem against unauthorized neural model training. To protect user personal data, we propose an anti-neuron watermarking approach based on linear color transformation. By watermarking user’s images with private signature using LCT, potential privacy breach of user personal data can be detected and verified by a third-party neutral arbitrator. Through extensive experiments, we show empirically that LCT-based watermarking is effective in protecting user data in a various realistic settings.

![Image](image.png)

Figure 9. Illustration of watermark pattern (left), recovered pattern (middle) and watermarked images (right). Recovering small visible watermark fails in getting noise.

**References**

[1] Wikipedia. Cambridge analytica, 2018.
[2] GDPR. Regulation (eu) 2016/679 (general data protection regulation) that is applicable as of may 25th, 2018 in all member states, is to the protection of natural persons with regard to the processing of personal data and rules relating to the free movement of personal data. https://gdpr-info.eu/, 2016.
[3] CCPA. California privacy act: https://oag.ca.gov/privacy/ccpa, January 2021.
[4] Standing Committee of the National People’s Congress. China data security law, July 2021.
[5] Nicholas Carlini, Chang Liu, Ulfar Erlingsson, Jernei 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.
[6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 1322–1333, 2015.
[7] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE symposium on security and privacy (SP), pages 739–753. IEEE, 2019.
[8] 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.
[9] Ingemar J Cox, Matthew L Miller, Jeffrey Adam Bloom, and Chris Honsinger. Digital watermarking, volume 53. Springer, 2002.
[10] Deepa Kundur and Dimitrios Hatzinakos. Digital watermarking using multiresolution wavelet decomposition. In Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP’98 (Cat. No. 98CH36181), volume 5, pages 2969–2972. IEEE, 1998.
[11] Zhaoxiong Meng, Tetsuya Morizumi, Sumiko Miyata, and Hirotsugu Kinoshita. Design scheme of copyright management system based on digital watermarking and blockchain. In 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), volume 2, pages 359–364. IEEE, 2018.
[12] Anatol Z Tirkel, GA Rankin, RM Van Schyndel, WJ Ho, NRA Mee, and Charles F Osborne. Electronic watermark. Digital Image Computing, Technology and Applications (DICTA’93), pages 666–673, 1993.
[13] Pao-Ta Yu, Hung-Hsu Tsai, and Jyh-Shyan Lin. Digital watermarking based on neural networks for color images. Signal processing, 81(3):663–671, 2001.
[14] X. Zhong, P. Huang, S. Mastorakis, and F. Y. Shih. An automated and robust image watermarking scheme based on deep neural networks. IEEE Transactions on Multimedia, pages 1–1, 2020.
[15] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. arXiv preprint arXiv:2008.03703, 2020.
[16] Maher El’arbi, Chokri Ben Amar, and Henri Nicolas. Video watermarking based on neural networks. In 2006 IEEE International conference on multimedia and expo, pages 1577–1580. Ieee, 2006.
Figure 10. Average loss of user samples with rectangle masking watermarking for Resnet50 on Tiny ImageNet.

[17] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 1–8. IEEE, 2018.

[18] Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin’ichi Satoh. Digital watermarking for deep neural networks. International Journal of Multimedia Information Retrieval, 7(1):3–16, 2018.

[19] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017.

[20] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access, 7:47230–47244, 2019.

[21] Nils Homer, Szabolcs Szlinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. PLoS Genet, 4(8):e1000167, 2008.

[22] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In 2018 IEEE 31st Computer Security Foundations Symposium (CSF), pages 268–282. IEEE, 2018.

[23] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. White-box vs black-box: Bayes optimal strategies for membership inference. In Kamlika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 8617. Springer, 2014.

[24] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhongqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 259–274, 2019.

[25] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 634–646, 2018.

[26] Juan A Garay and Rosario Gennaro. Advances in Cryptology—CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II, volume 8617. Springer, 2014.

[27] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. Communications of the ACM, 64(3):107–115, 2021.

[28] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 619–633, 2018.

[29] Liyao Xiang, Hao Zhang, Haotian Ma, Yifan Zhang, Jie Ren, and Quanshi Zhang. Interpretable complex-valued neural networks for privacy protection. In International Conference on Learning Representations, 2020.

[30] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Towards a visual privacy advisor: Understanding and predicting privacy risks in images. In Proceedings of the IEEE International Conference on Computer Vision, pages 3686–3695, 2017.

[31] 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.

[32] Cynthia Dwork. Differential privacy: A survey of results. In International conference on theory and applications of models of computation, pages 1–19. Springer, 2008.

[33] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pages 1310–1321, 2015.

[34] Shrutika Tople, Amit Sharma, and Aditya Nori. Alleviating privacy attacks via causal learning. In International Conference on Machine Learning, pages 9537–9547. PMLR, 2020.

[35] Hai Phan, My T Thai, Han Hu, Ruoming Jin, Tong Sun, and Dejing Dou. Scalable differential privacy with certified robustness in adversarial learning. In International Conference on Machine Learning, pages 7683–7694. PMLR, 2020.

[36] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), pages 571–582, 2014.

[37] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, pages 1273–1282. PMLR, 2017.

[38] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In Proceedings of
the IEEE conference on computer vision and pattern recognition, pages 4829–4837, 2016.

[39] Jonas Geiping, Hartmut Bauermeister, Hannah Drüge, and Michael Moeller. Inverting gradients—how easy is it to break privacy in federated learning? In Advances in Neural Information Processing Systems, 2020.

[40] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5188–5196, 2015.

[41] Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Haolin Jin. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In Proceedings of the European Conference on Computer Vision (ECCV), pages 606–624, 2018.

[42] Zhongzheng Ren, Yong Jae Lee, and Michael S Ryoo. Learning to anonymize faces for privacy preserving action detection. In Proceedings of the European Conference on Computer Vision (ECCV), pages 620–636, 2018.

[43] Ries Uittenbogaard, Clint Sebastian, Julien Vijverberg, Bas Boom, Dario M Gavrilu, et al. Privacy protection in streetview panoramas using depth and multi-view imageries. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 10581–10590, 2019.

[44] Pablo Speciale, Johannes L Schonberger, Sudipta N Sinha, and Marc Pollefeys. Privacy preserving image queries for camera localization. In Proceedings of the IEEE International Conference on Computer Vision, pages 1486–1496, 2019.

[45] Seong Joon Oh, Mario Fritz, and Bernt Schiele. Adversarial image perturbation for privacy protection: a game theory perspective. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 1491–1500. IEEE, 2017.

[46] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: Protecting privacy against unauthorized deep learning models. In 29th USENIX Security Symposium (USENIX Security 20), pages 1589–1604, 2020.

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

[48] Cifar. 2009. CIFAR Dataset, available at https://www.cs.toronto.edu/~kriz/cifar.html.

[49] kaggle. 2017. Tiny Imagenet, available at https://www.kaggle.com/c/tiny-imagenet.

[50] 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.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[52] Liu Kuang, 2021. pytorch-cifar, available at https://github.com/kuangliu/pytorch-cifar.

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[54] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.

[55] 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.

[56] Fisher Yu, Dequan Wang, Evan Shellhamer, and Trevor Darrell. Deep layer aggregation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2403–2412, 2018.

[57] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

[58] 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.

[59] Jeremy Cohen, Elian Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In International Conference on Machine Learning, pages 1310–1320. PMLR, 2019.

[60] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations, 2018.

[61] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? arXiv preprint arXiv:1906.02629, 2019.

[62] Yigicän Kaya and Tudor Dumitras. When does data augmentation help with membership inference attacks? In International Conference on Machine Learning, pages 5345–5355. PMLR, 2021.

[63] Congzheng Song and Vitaly Shmatikov. Overlearning reveals sensitive attributes. In International Conference on Learning Representations, 2020.

A Definitions of Terms from GDPR.

To help readers better understand the terms we used in the manuscript, we quote several term definitions from GDPR as following:

1. “Personal data” means any information relating to an identified or identifiable natural person (‘data subject’); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person.

2. “Processing” means any operation or set of operations that is performed on personal data or on sets of personal data, whether or not by automated means, such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction.

3. “Third party” means a natural or legal person, public authority, agency or body other than the data subject, controller, processor and persons who, under the direct author-
ity of the controller or processor, are authorised to process personal data.

(11) “Consent” of the data subject means any freely given, specific, informed and unambiguous indication of the data subject’s wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her.

(12) “Personal data breach” means a breach of security leading to the accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to, personal data transmitted, stored or otherwise processed.