gMINT: Gradiant-based Membership Inference Test applied to Image Models

Daniel DeAlcala Aythami Morales Julian Fierrez Gonzalo Mancera
Ruben Tolosana
Biometrics and Data Pattern Analytics Lab, Universidad Autonoma de Madrid, Spain
daniel.dealcala@uam.es aythami.morales@uam.es julian.fierrez@uam.es
gonzalo.mancera@uam.es ruben.tolosana@uam.es

Abstract

Membership Inference Test (MINT) is a method designed to determine whether specific data samples were used during model training. This paper explores the use of gradient-based information for MINT (gMINT). In particular, we take advantage of the gradient updates applied to each weight during the learning process, which we refer to as Weight Modifiers. We systematically analyze different types of Weight Modifiers and propose various approaches to process them effectively. Our experiments are carried out on multiple state-of-the-art architectures and four benchmark datasets, demonstrating that our method achieves near-perfect detection (100% in most scenarios). These results highlight the effectiveness of Weight Modifiers as a key cue for Membership Inference, further advancing the field of model auditing and privacy assessment in AI systems.

1. Introduction

In recent years, with the rapid rise of AI, various countries have intensified efforts to develop legislation that ensures that AI respects and safeguards the rights of citizens [46, 47]. Membership Inference Test (MINT) [7] aligns with these initiatives, serving as an auditing tool aimed at fostering more transparent and trustworthy AI [35]. The goal of MINT is to determine whether specific data have been used during training of a model. (A working MINT demonstrator was recently released [8][1].) This capability enables the detection of the illicit use of private [13, 15, 48], sensitive [28], or copyrighted data [14]. Another potential application of MINT is in international competitions, where improper data usage could unfairly improve model performance [9, 11, 41]. Backed by existing legislation, MINT has significant potential as a high-impact tool.

[1] https://ai-mintest.org/

MINT is rooted in Membership Inference Attacks (MIAs) [42], which aim to extract confidential information from a model. Although both MINT and MIAs share foundational principles, their different objectives lead to different environmental conditions, methodologies, and outcomes. MIAs, being classified as attacks, cannot assume direct access to the model being attacked [18]. Instead, they rely on partial knowledge of the target model to train the so-called shadow models - replicas designed to approximate the behavior of this target model and serve as the foundation for the attack [30]. This represents a key difference in environmental conditions: MINT is an auditing tool that, under the framework of existing legislation, grants access to the original model, reinforcing its role in regulatory compliance.

These differences in environmental conditions can lead to different methodologies and outcomes [7]. In this paper, we specifically analyze the role of gradient information in MINT by introducing gMINT. Although MIAs do not show clear advantages in leveraging gradients [29], we demonstrate that, in the context of MINT, gradient utilization provides a significant benefit.

The main contributions can be summarized as follows.

- We propose the use of gradient-based information, specifically Weight Modifiers, for the Membership Inference Test (gMINT), leveraging the updates applied to each model weight during training.
- We explore different types of Weight Modifiers and introduce multiple methods to process and analyze them effectively.
- We perform a comprehensive comparison between our approach (gMINT) and previous MINT methods, demonstrating that Weight Modifiers provide a more effective cue for Membership Inference.
- We conducted extensive experiments on four benchmark image datasets and multiple state-of-the-art architectures for image recognition, achieving near-perfect member-

ship detection in most scenarios. (Related experiments applying gMINT in order to assess whether a given text was used for training LLMs can be found elsewhere [26]).
- Our results highlight the potential of Weight Modifiers as a powerful tool for model auditing, contributing to the advancement of privacy assessments in AI systems.

The article is structured as follows. Sect. 2 reviews previous approaches to Membership Inference and the use of gradients in model auditing. Sect. 3 provides the mathematical formulation of gradients and their role in neural network training. Sect. 4 formally defines the Membership Inference Test (MINT) and details how gradients, specifically Weight Modifiers, are incorporated into our gMINT framework. Sect. 5 outlines the data partitioning strategy for training and evaluation, describes how Weight Modifiers are processed and analyzed, and details the architectures and experimental setup used. Sect. 6 presents the performance of our gMINT in different settings, including comparisons with previous MINT methods. Finally, Sect. 7 analyzes the implications of our findings and potential directions for future work and Sec. 8 presents the conclusions.

2. Related Works

In this work, we work on MINT [7, 8], demonstrating that leveraging gradients in this framework is highly beneficial. MINT is rooted in MIAs [42], where the advantages of using gradient information are not evident.

2.1. Membership Inference Attacks (MIAs)

Shokri *et al.* (2017) [42] introduced MIAs, demonstrating that an attacker could extract sensitive user information (e.g. medical records or shopping preferences) from a trained model. MIAs operate under the assumption that the attacker has partial knowledge of the target model, such as its architecture, training process, and certain statistics of the original dataset. Based on this knowledge, shadow models are trained to mimic the behavior of the target model [18, 30]. These shadow models are then used to infer which data samples were part of the training set, on the premise that given a sufficient number of shadow models [18, 42], the attacker can approximate the behavior of the target model.

Most MIA approaches focus on leveraging the model's output [24, 37, 43, 49] in what are referred to as black-box MIAs. Nasr et al. [29] later introduced white-box MIAs, which exploit intermediate model information, such as hidden activations or gradient updates. However, white-box access has not shown significant advantages over black-box MIAs, giving comparable results when using intermediate activations [29, 36] and only marginal improvements when incorporating gradient information [23, 29].

MIAs remain a challenging task, and multiple studies have explored their inherent complexity [34, 39]. MIAs have proven to be quite adaptable, extending their reach to a wide variety of models. For example, MIAs have been extended to language models [3, 25], diffusion models [10], and audio models [27, 40].

2.2. Membership Inference Test (MINT)

MINT [7] is an auditing tool designed to detect potential data misuse, not an attack aiming to extract sensitive information. Instead of being an adversary, MINT operates under an auditing entity that seeks to verify whether unauthorized data has been used. Unlike MIAs, MINT is supported by various legal regulations and, since it is not classified as an attack, assumes access to the original model.

The work [7] conducted experiments on face recognition models, demonstrating that MINT detection is feasible even for complex models trained on millions of images. Furthermore, [6] explored various training factors that can influence detection, either positively (enhancing transparency) or negatively (concealing unauthorized data usage). MINT has also been recently studied on LLMs [26] to infer whether a given text was used to train them.

3. Gradients in Machine Learning

Let W be the set of parameters of a neural network, where each weight W_j contributes to the network output $f_W(d)$, given an input sample d from the training set, defined as (d, y), where y is the corresponding label.

The loss function $L_W(d)$ is defined as:

$$L_W(d) = l(f_W(d), y) \tag{1}$$

Intuitively, this function quantifies the discrepancy between the prediction of the model and the actual label. The function l varies depending on the task at hand (e.g., mean squared error, cross-entropy, etc.).

The general objective of training is to minimize this loss function for each training sample d. Optimization algorithms achieve this by updating the parameters W in the opposite direction of the loss function's $L_W(d)$ gradient, thereby moving towards its minima.

Each weight W_j directly influences the network's output and, consequently, the loss function $L_W(d)$. As a result, this loss function is differentiable with respect to each weight, allowing us to compute the gradient of the loss with respect to any given weight:

$$\nabla W_j(d) = \frac{\partial L_W(d)}{\partial W_j} \tag{2}$$

Thus, the learning process consists of updating each weight via backpropagation, adjusting its value in the opposite direction of the gradient. Although the specific update rule depends on the optimization algorithm used, a general formulation is as follows.

$$W_j^{t+1} = W_j^t - G(\nabla W_j(d)) \tag{3}$$

G represents a term that depends on the gradient ∇W_j as well as other hyperparameters specific to the optimization algorithm. We refer to this term as the *Weight Modifier*.

4. gMINT: Motivation and Architecture

MINT is an auditing tool that enables an auditing entity (e.g., a public institution such as the EU/US, a private organization, or even the owner of the model) to assess the data used during the training process of a model. It requires access to the model and, using specific extracted information, trains what is referred to as the MINT Model. This MINT Model is responsible for detecting which specific data samples were used during the model training.

The authors of the original MINT [7] examined both black-box and white-box settings. In the black-box scenario, only the model's output is accessible, while the white-box setting allows access to internal information such as intermediate activations, revealing how the model processes input data [29]. To take advantage of this, they performed feedforward passes over the data and extracted intermediate activations at different points in the network (near the input, intermediate, and near the output). Based on this information, they trained the MINT Model as a binary classifier to determine whether a given data sample was part of the model training set.

4.1. Motivation: gMINT vs MINT vs MIAs

The authors of MINT [7] demonstrated that white-box access provides clear advantages in detection performance compared to black-box access, while in MIA white-box access did not produce significantly superior results [29]. This difference comes from the fact that MINT operates under different environmental conditions. Since MINT is considered an auditing tool supported by current legislation [46, 47], it assumes some level of cooperation from the model developer, allowing technologies to be designed with access to the original model. In contrast, MIAs are considered attacks, making access to the original model (and thus white-box access) much harder to assume. Combined with the limited performance improvements observed under the white-box environment, this has led to a reduced focus on such settings in MIAs compared to MINT.

Gradients, in particular, showed slight improvements over black-box access in MIAs [29]. However, due to the adversarial nature of MIAs, obtaining access to this information is highly challenging, which has limited both the adoption and the study of gradient-based approaches. In MINT, on the other hand, the starting premise is fundamentally different, as access to the original model (and thus to its gradients) can be assumed. This key difference, along with the superior detection performance observed for MINT in white-box settings (unlike MIAs), has motivated our exploration of gradient-based information within the MINT framework, resulting in the gMINT approach proposed here.

4.2. Elements and Architecture

In the following, we define the key elements and terms used throughout this work. Fig. 1 illustrates the key elements of our gMINT approach.

MINT aims to detect which data samples were used to train a model, referred to as the Audited Model. To determine whether a given sample was used in the training of the Audited Model, another model is trained using information extracted from it. This secondary model is called the MINT Model, which is a binary classifier that determines whether a data sample was part of the Audited Model training set. In gMINT the extracted information consists of the Weight Modifier (Sect. 3).

- Audited Model $M(\cdot|W)$: The model under audit, for which we aim to determine whether a specific sample was included in its training set. This model is defined by its architecture and parameters W.
- MINT Model $T(\cdot|\theta)$: A model trained using information extracted from the Audited Model, which determines whether a given sample was part of the Audited Model's training data. It is defined by its architecture and parameters θ.
- Training Data $\mathcal{D}$: The dataset used to train the Audited Model M.
- External Data $\mathcal{E}$: Data samples that were not used to train the Audited Model M.
- Weight Modifier $G(\nabla W_j(d))$: The information extracted from the Audited Model when processing a data sample d, used to train the MINT Model T.

4.3. Weight Modifiers: Motivation

After extensive experimentation, we concluded that using the gradient $\nabla W_j(d)$ directly as input to the MINT Model T does not produce competitive performances in MINT application. The raw gradients do not provide enough information to distinguish between samples used in training $d \in \mathcal{D}$ and those that were not $d \in \mathcal{E}$.

We therefore propose using the Weight Modifier $G(\nabla W_j(d))$ instead of the gradient $\nabla W_j(d)$. This Weight Modifier directly represents how much each weight W_j of the model M is updated for a given data sample $d \in \mathcal{D} \cup \mathcal{E}$. Depending on the specific optimization algorithm, this Weight Modifier applies different operations to the gradient $\nabla W_j(d)$ (see Sect. 4.4).

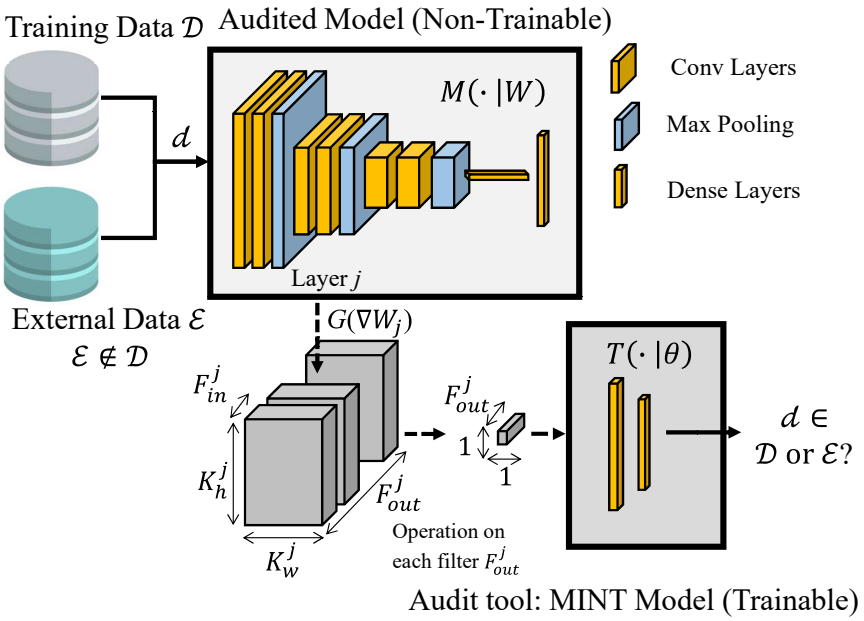

Figure 1. Workflow of gMINT where we use the Weight Modifiers from different layers of the Audited Model. These Weight Modifiers are transformed into vectors by applying different operations to train the MINT Model.

4.4. Weight Modifiers: Computation and Use

The Weight Modifier depends on the optimization algorithm and can have very simple analytical expressions, such as in SGD [2]:

$$G(\nabla W_j(d)) = \eta \nabla W_j(d) \tag{4}$$

where η is the learning rate.

Or more complex ones, such as in Adam [20]:

$$G(\nabla W_j(d)) = \eta \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon} \tag{5}$$

where $\hat{m}$ and $\hat{v}$ are terms dependent on the gradient $\nabla W_j(d)$, η is the learning rate, and ϵ prevents division by zero.

This analytical function can be used to obtain the model modifier. It requires knowledge of the learning algorithm and its hyperparameters. However, the modifier can be obtained more easily by rearranging Eq. 3, leading to the following definition.

$$G(\nabla W_j(d)) = W_j^t(d) - W_j^{t+1}(d) \tag{6}$$

Thus, we can compute this modifier by calculating the difference between the weights when the Audited Model M processes a sample d. Data d cannot be processed in batches, as this would yield the batch Weight Modifier instead of the Weight Modifier for the individual sample d.

4.5. Weight Modifiers: Dimensionality Reduction

Weight Modifiers conceptually serve as indicators of how much each weight in a model changes. Consequently, for each data sample, the amount of Weight Modifiers information available is equal to the number of parameters in the model. While using all the Weight Modifiers is feasible for small models, in larger architectures like those employed in this work, it becomes necessary to select a subset.

The filters of a model are structured as $(F_{out}^j, F_{in}^j, K_h^j, K_w^j)$, where F_{out}^j represents the number of filters in the layer j, F_{in}^j denotes the number of channels in each filter, and (K_h^j, K_w^j) corresponds to the dimensions of the filter, typically with $K_h^j = K_w^j$. Even extracting the Weight Modifiers information from a single filter results in $F_{out} \times F_{in} \times K_h \times K_w$ values per data sample d. For example, in a relatively small layer, this could be $64 \times 64 \times 3 \times 3 = 36,864$ values. While it is possible to train a MINT Model that directly processes all this modifier information from a filter, we explore more efficient alternatives for analyzing it. Specifically, we consider three methods for incorporating this information into the MINT Model:

- *Mean Vector Method*: Computes the mean value for each filter F_{out}^j by averaging the matrix (F_{in}^j, K_h^j, K_w^j), resulting in a size vector F_{out}^j that captures the mean Weight Modifier for each filter.
- *Min Vector Method*: Selects the smallest value within

each filter, generating a vector of size F_{out}^j that contains the minimum Weight Modifier associated with every filter.
- *Max Vector Method*: Retrieves the highest value from each filter, producing a vector of size F_{out}^j where each element corresponds to the maximum Weight Modifier found in that filter.

In this work, we analyze the impact of selecting modifier information from different filters at various depths within the model. Specifically, we extracted the Weight Modifiers from three different layers: one filter near the input of the model (Entry Setup), another at a mid-level depth (Middle Setup), and one close to the final layers (Output Setup). Finally, we also explore the effect of combining all of them (Combined Setup).

5. Experimental Protocol

Once all terms have been defined, we now describe the different components of the experimental protocol followed.

5.1. Databases and Models

We used four different image datasets, chosen to exhibit diverse characteristics in terms of image resolution, number of classes, and dataset size:

- **MNIST** [22]: This dataset consists of 60K training samples and 10K test samples, containing simple images of 10 digits (classes) in low resolution (28×28) and a single grayscale channel.
- **CIFAR-10** [21]: With 50K training samples and 10K test samples, CIFAR-10 features images from 10 classes with a significantly higher level of realism. The images have 32×32 pixels and three color channels (RGB).
- **GTSRB** [44]: This dataset includes 39K training samples and 13K test samples, comprising fine-grained classification data with 43 classes. The images vary in resolution, ranging from 15×15 pixels for the smallest to 222×293 pixels for the largest and three color channels (RGB).
- **Tiny-ImageNet** [5]: Containing 100K training samples and 20K test samples, Tiny-ImageNet is a subset of ImageNet with 200 classes and 64×64 pixel-resolution images with three color channels (RGB).

The models used include some of the leading state-of-the-art architectures: MobileNet [17], ResNet50 [16], ResNet101 [16], DenseNet121 [19], Xception [4], and a Vision Transformer (ViT) [1].

5.2. Data Division

Fig. 1 illustrates the gMINT process. The Audited Model M is trained in the dataset $\mathcal{D}$, while the MINT Model T is a binary classifier, trained with the Weight Modifiers, that distinguishes between data used for training $\mathcal{D}$ and external data $\mathcal{E}$ that were not part of the training process.

Initially, the data set (e.g. MNIST, CIFAR-10, etc.) is split into two equal parts. One half, denoted as $\mathcal{D}$, is used to train the Audited Model M, while the other half remains unused during training M and is considered external data $\mathcal{E}$. Once the Audited Model M has been trained on $\mathcal{D}$, Weight Modifiers are extracted from both $\mathcal{D}$ and $\mathcal{E}$. These Weight Modifiers are then split, with 80% available to train the MINT Model T and the remaining 20% reserved for evaluation. The evaluation results are in Sect. 6 are presented as the accuracy in the binary MINT task. Since the evaluation set is balanced, the baseline accuracy is 50%, which corresponds to random choice.

The fewer samples required to train the MINT Model, the better, as this implies less collaboration from the developer; therefore, we also present experiments where the number of training samples is reduced. Specifically, we consider three scenarios: using 100% (D100) of the samples available to train, 10% (D10) and 1% (D1). The performance on the evaluation set is reported in Sect. 6.

5.3. Audited Model Training

The Audited Model is trained on the data $\mathcal{D}$ (Sect. 5.2), which corresponds to half of the total dataset. The remaining half, denoted as external data $\mathcal{E}$, is not used for training, but is still processed to extract Weight Modifiers. The Audited Model is trained for 50 epochs following an Early Stopping strategy. The architectures used are those described in Sect. 5.1.

5.4. MINT Model Training

For the methods presented in Sect. 4.5, the MINT Model is implemented as a fully connected network. The architecture consists of two fully connected layers with 128 neurons each, followed by another layer with 64 neurons, and a final output layer with a single neuron for binary classification. All models are trained for 100 epochs, selecting the point that achieves the best performance on the evaluation set.

6. Results

6.1. Comparison of Weight Modifiers

As explained in Sect. 4.5, the information contained in the Weight Modifiers is extremely abundant, making it necessary to select only a subset of the available information. To this end, we focus on extracting the Weight Modifiers from the filters of specific layers (Entry Setup, Middle Setup, Output Setup, and Combined Setup). Then, we apply different operations to these Weight Modifiers, specifically considering three methods: the Mean method, the Max method, and the Min method.

In Table 1, we present a comparison of the three proposed methods in the MNIST data set under the different scenarios explored in this work (including the different se-

tups that extract Weight Modifiers from different layers, and different data availability scenarios for training). We emphasize that all results in this section are reported in terms of the accuracy of the binary MINT task. We refer to this accuracy as MINT detection. Since the evaluation set is balanced, the baseline accuracy is 50%, which corresponds to random choice. The first notable observation is that all three methods achieve very promising results, reaching near 100% MINT detection in at least one scenario for every model. This suggests that more complex approaches (such as using 3D convolutional architectures to analyze raw Weight Modifiers or Transformer-based architectures) are in principle not necessary. A second key takeaway is that although all three methods achieve nearly perfect performance, the Mean method performs slightly better, especially for more complex architectures like DenseNet121 and Xception. Therefore, in the following section we report experiments only using the Mean method.

6.2. Weight Modifiers Setups and Data Availability

In Tab. 2, we present the results of our extensive experimental framework on gMINT. We analyze the different factors described in Sect. 5, including four datasets and six state-of-the-art architectures, while varying the data availability for training (D100, D10 and D1) and modifying the different setups used to extract the Weight Modifiers (Entry, Middle, Output, and Combined Setups).

The first factor we analyze is the extraction setup. According to the results in Tab. 2, the Entry, Middle, and Output setups yield very similar performance, achieving near-perfect MINT detection in most cases. Differences among these three setups are minor: in some cases, the Middle setup performs slightly better (e.g., GTSRB dataset with MobileNet and DenseNet architectures), while in others, the Output setup is slightly superior (e.g., TinyImageNet dataset with DenseNet121). The Entry setup only shows slightly higher performance for the Xception model on TinyImageNet under the D100 data availability scenario.

However, the best performance is clearly achieved with the combined setup, which leverages information from all three setups, reaching almost 100% accuracy in almost all scenarios. Although it is expected that aggregating information from multiple setups leads to superior results, there are three specific cases where this does not hold: GTSRB with DenseNet121 and ViT and TinyImageNet with Xception, particularly in the D10 and D1 data availability scenarios. This performance drop is likely due to the combination of limited training data and the higher dimensionality of the input of the combined setup, which makes it more difficult for the MINT model to correctly learn the separation between the samples $\mathcal{D}$ and $\mathcal{E}$.

The case of Xception on TinyImageNet represents a notable outlier, where Weight Modifiers seem to be extremely noisy, likely due to specific architectural components (such as separable convolutions) combined with the higher complexity of the TinyImageNet dataset. For these particular situations, future work exploring alternative Weight Modifier processing methods or more advanced MINT model architectures may be worth investigating. In Sect. 6.3 we propose an interesting future direction for these cases.

The second factor we analyze is the availability of data for training. Although MINT typically assumes some level of collaboration from the model developer, the less data required, the more practical and applicable the MINT approach becomes. Therefore, the goal is to achieve the highest possible accuracy while relying on the smallest amount of data. As shown in the Tab. 2, even in the D1 scenario — where only 1% of the data used to train the Audited Model (500 samples for MNIST, 400 for CIFAR-10, 250 for GTSRB, and 900 for TinyImageNet) are available to train the MINT Model — the results are nearly identical to those obtained with D100, reaching close to 100% MINT accuracy. This is a highly promising result that highlights the strong applicability of this approach and opens the door to future research where MINT models could potentially be trained in an unsupervised manner.

6.3. Benchmarking gMINT against MINT

Finally, in Table 3, we present our gMINT results alongside those of the original MINT method under similar conditions [7]. As shown, our results significantly outperform MINT, demonstrating the superiority of the gMINT method. An interesting result appears with Xception trained on Tiny ImageNet, where MINT performs the best, while gMINT performs the worst. As discussed in [7], MINT tends to perform better with more complex models. This opens up the possibility of combining MINT and gMINT in certain cases to achieve better results where gMINT shows poorer performance, leaving a potential avenue for future research.

7. Discussion and Future Work

MINT is an auditing tool designed to identify which data samples were used to train a model, with the goal of improving transparency and trustworthiness from the user's perspective. In gMINT, we propose a method that leverages gradient information through the concept of Weight Modifiers (a function that depends on both the model's gradients and other hyperparameters) to perform this detection, achieving significantly better results than the original MINT method [7]. However, gMINT also comes with certain limitations. Specifically, obtaining Weight Modifiers is a more invasive process than simply accessing activations, as it requires knowledge of both the loss function and the optimization algorithm, as discussed in Sec. 4. In scenarios where such access is not possible, the use of MINT remains necessary. However, when this information can be obtained,

Table 1. Accuracy results for our gMINT are presented as the percentage of accuracy on the evaluation set. Since the set is balanced, random choice corresponds to 50%, while perfect classification is represented by 100%. Results are provided for different models and the MNIST dataset described in Sec. 5.1, considering the various Weight Modifier operations introduced in Sec. 4.5, and under different data availability scenarios.

	MEAN Method																	
gMINT	MobileNet			ResNet50			ResNet100			DenseNet121			Xception			ViT		
	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1
Entry Setup	100	99.94	99.91	100	99.80	99.65	100	100	100	52.32	50.00	50.00	100	100	100	100	100	100
Middle Setup	100	100	99.91	100	99.98	99.95	100	100	100	100	99.74	50.00	100	100	100	100	100	100
Output Setup	100	100	100	100	99.99	99.95	100	100	100	100	99.91	50.00	100	100	100	100	100	100
All Setup	100	100	100	100	100	99.96	100	100	100	100	100	100	100	100	100	100	100	100
	MAX Method																	
gMINT	MobileNet			ResNet50			ResNet100			DenseNet121			Xception			ViT		
	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1
Entry Setup	99.75	99.71	99.42	100	100	100	100	99.89	97.18	91.04	79.94	68.90	99.33	98.82	93.30	91.27	92.17	87.79
Middle Setup	99.93	99.86	99.84	100	100	100	99.98	99.87	98.56	91.66	91.83	86.37	99.94	99.94	99.81	100	100	100
Output Setup	99.91	99.86	99.79	100	100	100	98.97	98.75	98.75	96.25	95.92	85.84	99.94	99.93	99.81	100	100	100
All Setup	100	99.96	99.07	100	100	100	99.82	99.80	97.60	96.23	95.39	89.04	99.98	99.98	98.65	100	100	100
	MIN Method																	
gMINT	MobileNet			ResNet50			ResNet100			DenseNet121			Xception			ViT		
	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1
Entry Setup	100	100	100	99.99	99.98	99.93	100	100	99.96	87.20	87.00	69.72	99.37	97.31	90.29	99.79	99.79	94.77
Middle Setup	100	100	100	100	99.97	99.50	100	99.97	99.75	90.41	90.31	85.94	100	98.75	98.67	100	100	100
Output Setup	100	100	100	100	100	100	100	99.96	99.77	90.94	91.64	86.68	99.98	99.86	99.36	100	100	100
All Setup	100	100	100	100	100	100	100	100	99.94	91.65	91.04	85.82	99.98	99.86	99.30	100	100	100

Table 2. Accuracy results for our gMINT are presented as the percentage of accuracy on the evaluation set. Since the set is balanced, random choice corresponds to 50%, while perfect classification is represented by 100%. Results are shown for different models and datasets described in Sec. 5.1, under various data availability scenarios for training presented in Sec. 5.2 (D100 indicates access to 100% of the available data, D10 to 10%, and D1 to 1%) and for different setups of Weight Modifier extraction (Sec. 4.5). All results using the MEAN method for Weight Modifier dimensionality reduction.

	MNIST																	
gMINT	MobileNet			ResNet50			ResNet100			DenseNet121			Xception			ViT		
	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1
Entry Setup	100	99.94	99.91	100	99.80	99.65	100	100	100	52.32	50.00	50.00	100	100	100	100	100	100
Middle Setup	100	100	99.91	100	99.98	99.95	100	100	100	100	99.74	50.00	100	100	100	100	100	100
Output Setup	100	100	100	100	99.99	99.95	100	100	100	100	99.91	50.00	100	100	100	100	100	100
Combined Setup	100	100	100	100	100	99.96	100	100	100	100	100	100	100	100	100	100	100	100
	CIFAR-10																	
gMINT	MobileNet			ResNet50			ResNet100			DenseNet121			Xception			ViT		
	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1
Entry Setup	100	100	99.98	99.99	99.98	99.98	99.99	99.96	99.87	100	100	100	100	100	100	100	100	100
Middle Setup	100	100	99.98	99.99	99.98	99.98	100	100	99.87	100	100	100	100	100	100	100	100	100
Output Setup	100	100	100	100	99.99	99.98	100	100	100	100	100	100	100	100	100	100	100	100
Combined Setup	100	100	100	100	100	99.96	100	100	100	100	100	100	100	100	100	100	100	100
	GSTRB																	
gMINT	MobileNet			ResNet50			ResNet100			DenseNet121			Xception			ViT		
	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1
Entry Setup	99.20	99.21	99.22	96.92	95.92	94.33	98.87	98.37	98.68	99.99	99.93	94.39	99.98	99.98	99.88	98.00	94.58	89.24
Middle Setup	100	100	99.97	99.96	99.96	99.80	99.86	99.80	99.66	100	100	99.64	99.98	99.98	99.98	94.42	92.29	92.03
Output Setup	98.84	98.61	98.61	99.99	99.96	99.96	99.93	99.93	99.92	99.98	99.37	99.09	99.98	99.98	99.90	99.69	99.48	99.42
Combined Setup	100	100	99.97	100	99.96	99.96	99.94	99.92	99.92	99.98	99.98	98.32	100	99.98	99.94	99.73	98.36	98.33
	Tiny ImageNet																	
gMINT	MobileNet			ResNet50			ResNet100			DenseNet121			Xception			ViT		
	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1	D100	D10	D1
Entry Setup	100	99.99	99.96	100	100	100	100	100	100	53.40	50.00	50.00	82.13	73.33	70.60	100	100	100
Middle Setup	100	100	99.96	100	100	100	100	100	100	100	99.97	58.33	80.21	77.09	76.61	100	100	100
Output Setup	100	100	99.98	100	100	100	100	100	100	100	100	100	75.39	74.92	69.64	100	100	100
Combined Setup	100	100	99.99	100	100	100	100	100	100	100	100	100	77.3	72.57	69.52	100	100	100

gMINT allows for almost perfect detection of training samples, even when the developer provides only a very limited set of training data.

Furthermore, since computing gradients requires running both a feedforward pass and a backpropagation step, obtaining gradients inherently includes obtaining activa-

Table 3. Comparison of gMINT and original MINT in terms of accuracy across different architectures and datasets. The table reports the accuracy (%) achieved by both methods under identical evaluation settings, highlighting the generalization and robustness improvements of gMINT over the original MINT.

	MNIST					
Method	MobileNet	ResNet50	ResNet100	DenseNet121	Xception	ViT
MINT [7]	52.35	51.28	51.47	52.16	54.59	53.21
gMINT [**ours**]	100.00	99.96	100.00	100.00	100.00	100.00
	CIFAR-10					
Method	MobileNet	ResNet50	ResNet100	DenseNet121	Xception	ViT
MINT [7]	66.47	66.38	69.12	60.72	67.29	60.51
gMINT [**ours**]	100.00	99.96	100.00	100.00	100.00	100.00
	GSTBR					
Method	MobileNet	ResNet50	ResNet100	DenseNet121	Xception	ViT
MINT [7]	59.48	61.17	61.02	61.35	59.79	60.41
gMINT [**ours**]	99.97	99.96	99.92	98.32	99.94	98.33
	Tiny Imagenet					
Method	MobileNet	ResNet50	ResNet100	DenseNet121	Xception	ViT
MINT [7]	60.11	61.47	61.29	62.18	63.02	59.83
gMINT [**ours**]	99.99	100.00	100.00	100.00	69.52	100.00

tions, the key element used by MINT [7]. This direct connection between gradients and activations suggests that both methods could be naturally combined to improve performance in cases where gMINT alone is less effective, opening an interesting direction for future work on context-aware multiple classifier fusion [12, 32] for MINT. This combination approach of classifiers can also incorporate other internal information from the audited model [38] correlated with training data to help the MINT process.

Exploring unsupervised approaches is also a promising research line as these methods would not require the developer to share any training data. However, for these methods to work, it is still necessary that among the collected data, whether obtained by scraping the Internet, combining public datasets, or other means, some samples must have been used to train the model, even if we do not know which ones. This imposes a limitation, making unsupervised methods applicable only when there is reasonable certainty about the potential origin of the training data. Moreover, in most practical cases, the data that was not used for training $\mathcal{E}$ is likely to be much more abundant than the data that were $\mathcal{D}$. Designing effective unsupervised methods requires careful attention to heavily imbalanced scenarios, which makes this a challenging but valuable direction for future research.

Finally, two other research lines around MINT in our agenda are: 1) incorporating human help in the MINT process (e.g. efficiently curating part of the data [33] or the information extracted from the Audited Model), and 2) extending MINT from a binary classification problem to a richer output in a way of explainable AI [31, 45] trying to respond (to some extent) the question: How were these data used to train this model?

8. Conclusion

In this work, we present gMINT, a novel auditing tool designed to detect AI models' training data. Based on MINT, gMINT leverages Weight Modifiers (functions derived from gradients) to significantly improve MINT performance. Our extensive experimental evaluation demonstrates that gMINT consistently outperforms MINT, achieving near-perfect detection in diverse scenarios and architectures. Despite its strong performance, gMINT requires access to gradient information, which makes it slightly more invasive than MINT and potentially limits its applicability in some real-world settings. (A working MINT demonstrator was recently released [8][2].)

Addressing gMINT limitations, we identify several future research directions, including hybrid approaches combining MINT and gMINT, as well as the development of unsupervised methods that do not require access to training data. We believe gMINT takes a step forward toward more transparent and accountable AI models.

9. Acknowledgement

This work has been supported by projects BBforTAI (PID2021-127641OB-I00 MICINN/FEDER), Cátedra ENIA UAM-VERIDAS (NextGenerationEU PRTR TSI-100927-2023-2), and Comunidad de Madrid (ELLIS Unit Madrid). D. DeAlcala is supported by a FPU Fellowship (FPU21/05785) from the Spanish MIU. G. Mancera is supported by FPI-PRE2022-104499 MICINN/FEDER.

[2] https://ai-mintest.org/

References

[1] Dosovitskiy Alexey. An image is worth 16 × 16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*, 2020. 5

[2] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993. 4

[3] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021. 2

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017. 5

[5] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *arXiv preprint arXiv:1707.08819*, 2017. 5

[6] Daniel Dealcala, Gonzalo Mancera, Aythami Morales, Julian Fierrez, Ruben Tolosana, and Javier Ortega-Garcia. A Comprehensive Analysis of Factors Impacting Membership Inference. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3585–3593, 2024. 2

[7] Daniel DeAlcala, Aythami Morales, Gonzalo Mancera, Julian Fierrez, Ruben Tolosana, and Javier Ortega-Garcia. Is my Data in your AI Model? Membership Inference Test with Application to Face Images. *arXiv preprint arXiv:2402.09225*, 2024. 1, 2, 3, 6, 8

[8] Daniel DeAlcala, Aythami Morales, Julian Fierrez, Gonzalo Mancera, Ruben Tolosana, and Ruben Vera-Rodriguez. MINT-Demo: Membership inference test demonstrator. In *AAAI Workshop on AI Governance: Alignment, Morality, and Law (AIGOV)*, 2025. 1, 2, 8

[9] Ivan DeAndres-Tame, Ruben Tolosana, Pietro Melzi, Ruben Vera-Rodriguez, Minchul Kim, Christian Rathgeb, Xiaoming Liu, Luis F. Gomez, Aythami Morales, Julian Fierrez, Javier Ortega-Garcia, Zhizhou Zhong, Yuge Huang, Yuxi Mi, Shouhong Ding, Shuigeng Zhou, Shuai He, Lingzhi Fu, Heng Cong, Rongyu Zhang, Zhihong Xiao, Evgeny Smirnov, Anton Pimenov, Aleksei Grigorev, Denis Timoshenko, Kaleb Mesfin Asfaw, Cheng Yaw Low, Hao Liu, Chuyi Wang, Qing Zuo, Zhixiang He, Hatef Otroshi Shahreza, Anjith George, Alexander Unnervik, Parsa Rahimi, Sébastien Marcel, Pedro C. Neto, Marco Huber, et al. Second FRCSyn-onGoing: Winning solutions and post-challenge analysis to improve face recognition with synthetic data. *Information Fusion*, 120:103099, 2025. 1

[10] Jinhao Duan, Fei Kong, Shiqi Wang, Xiaoshuang Shi, and Kaidi Xu. Are diffusion models vulnerable to membership inference attacks? In *International Conference on Machine Learning*, pages 8717–8730. PMLR, 2023. 2

[11] Meiling Fang, Marco Huber, Julian Fierrez, Raghavendra Ramachandra, Naser Damer, Alhasan Alkhaddour, Maksim Kasantcev, Vasiliy Pryadchenko, Ziyuan Yang, Huijie Huangfu, Yingyu Chen, Yi Zhang, Yuchen Pan, Junjun Jiang, Xianming Liu, Xianyun Sun, Caiyong Wang, Xingyu Liu, Zhaohua Chang, Guangzhe Zhao, Juan Tapia, Lazaro Gonzalez-Soler, Carlos Aravena, and Daniel Schulz. SynFacePAD 2023: Competition on face presentation attack detection based on privacy-aware synthetic training data. In *IEEE/IAPR Intl. Joint Conf. on Biometrics (IJCB)*, 2023. 1

[12] Julian Fierrez, Aythami Morales, Ruben Vera-Rodriguez, and David Camacho. Multiple classifiers in biometrics. Part 2: Trends and challenges. *Information Fusion*, 44:103–112, 2018. 8

[13] Luciano Floridi. Open data, data protection, and group privacy. *Philosophy & Technology*, 27:1–3, 2014. 1

[14] Jane C Ginsburg. Copyright, common law, and sui generis protection of databases in the united states and abroad. *U. Cin. L. Rev.*, 66:151, 1997. 1

[15] Ahmad Hassanpour, Majid Moradikia, Bian Yang, Ahmed Abdelhadi, Christoph Busch, and Julian Fierrez. Differential privacy preservation in robust continual learning. *IEEE Access*, 10, 2022. 1

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 5

[17] Andrew G Howard. MobileNets: efficient convolutional neural networks for mobile vision applications. In *Proceedings of the IEEE/CVF Confernce on Computer Vision and Pattern Recognition*, pages 21–26, 2017. 5

[18] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership Inference Attacks on Machine Learning: A survey. *ACM Computing Surveys*, 54(11s):1–37, 2022. 1, 2

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 5

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Toronto, ON, Canada*, 2009. 5

[22] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. 5

[23] Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated White-Box membership inference. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1605–1622. USENIX Association, 2020. 2

[24] Zheng Li and Yang Zhang. Membership leakage in label-only exposures. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, pages 880–895, 2021. 2

[25] Zhan Li, Yongtao Wu, Yihang Chen, Francesco Tonin, Elias Abad Rocamora, and Volkan Cevher. Membership inference attacks against large vision-language models. *Advances in Neural Information Processing Systems*, 37:98645–98674, 2025. 2

[26] Gonzalo Mancera, Daniel DeAlcala, Julian Fierrez, Ruben Tolosana, and Aythami Morales. Is my text in your AI model? Gradient-based membership inference test applied to LLMs. *arXiv preprint arXiv:2503.07384*, 2025. 2

[27] Yuantian Miao, Minhui Xue, Chao Chen, Lei Pan, Jun Zhang, Benjamin Zi Hao Zhao, Dali Kaafar, and Yang Xiang. The audio auditor: user-level membership inference in internet of things voice services. *Proceedings on Privacy Enhancing Technologies*, pages 209–228, 2021. 2

[28] Aythami Morales, Julian Fierrez, Ruben Vera-Rodriguez, and Ruben Tolosana. SensitiveNets: Learning agnostic representations with application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(6):2158–2164, 2021. 1

[29] 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 *Proc. of the IEEE Symposium on Security and Privacy*, pages 739–753, 2019. 1, 2, 3

[30] Jun Niu, Peng Liu, Xiaoyan Zhu, Kuo Shen, Yuecong Wang, Haotian Chi, Yulong Shen, Xiaohong Jiang, Jianfeng Ma, and Yuqing Zhang. A survey on membership inference attacks and defenses in machine learning. *Journal of Information and Intelligence*, 2(5):404–454, 2024. 1, 2

[31] Alfonso Ortega, Julian Fierrez, Aythami Morales, Zilong Wang, and Tony Ribeiro. Symbolic AI for XAI: Evaluating LFIT inductive programming for fair and explainable automatic recruitment. In *IEEE/CVF Winter Conf. on Applications of Computer Vision Workshops (WACVw)*, pages 78–87, 2021. 8

[32] Alejandro Peña, Ignacio Serna, Aythami Morales, Julian Fierrez, Alfonso Ortega, Ainhoa Herrarte, Manuel Alcantara, and Javier Ortega-Garcia. Human-centric multimodal machine learning: Recent advances and testbed on AI-based recruitment. *SN Computer Science*, 4(5):434, 2023. 8

[33] Alejandro Peña, Aythami Morales, Julian Fierrez, Javier Ortega-Garcia, Iñigo Puente, Jorge Cordova, and Gonzalo Cordova. Continuous document layout analysis: Human-in-the-loop AI-based data curation, database, and evaluation in the domain of public affairs. *Information Fusion*, 108: 102398, 2024. 8

[34] Shahbaz Rezaei and Xin Liu. On the difficulty of Membership Inference Attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7892–7900, 2021. 2

[35] Yao Rong, Tobias Leemann, Thai-Trang Nguyen, Lisa Fiedler, Peizhu Qian, Vaibhav Unhelkar, Tina Seidel, Gjergji Kasneci, and Enkelejda Kasneci. Towards Human-Centered explainable AI: A survey of user studies for model explanations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4):2104–2122, 2024. 1

[36] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. White-box vs black-box: Bayes optimal strategies for membership inference. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5558–5567. PMLR, 2019. 2

[37] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-Leaks: Model and data independent Membership Inference Attacks and defenses on Machine Learning Models. In *Proceedings of the Annual Network and Distributed System Security Symposium*, 2018. 2

[38] Ignacio Serna, Alejandro Peña, Aythami Morales, and Julian Fierrez. InsideBias: Measuring bias in deep networks and application to face gender biometrics. In *IAPR Intl. Conf. on Pattern Recognition (ICPR)*, pages 3720–3727, 2021. 8

[39] Avital Shafran, Shmuel Peleg, and Yedid Hoshen. Membership Inference Attacks are easier on difficult problems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14820–14829, 2021. 2

[40] Muhammad A. Shah, Joseph Szurley, Markus Mueller, Thanasis Mouchtaris, and Jasha Droppo. Evaluating the vulnerability of end-to-end automatic speech recognition models to membership inference attacks. In *Proceedings of Interspeech*, 2021. 2

[41] Hatef Otroshi Shahreza, Christophe Ecabert, Anjith George, Alexander Unnervik, Sébastien Marcel, Nicolò Di Domenico, Guido Borghi, Davide Maltoni, Fadi Boutros, Julia Vogel, Naser Damer, Ángela Sánchez-Pérez, Enrique Mas-Candela, Jorge Calvo-Zaragoza, Bernardo Biesseck, Pedro Vidal, Roger Granada, David Menotti, Ivan DeAndres-Tame, Simone Maurizio La Cava, Sara Concas, Pietro Melzi, Ruben Tolosana, Ruben Vera-Rodriguez, Gianpaolo Perelli, Giulia Orrù, Gian Luca Marcialis, and Julian Fierrez. SDFR: Synthetic data for face recognition competition. In *2024 IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG)*, 2024. 1

[42] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks against Machine Learning models. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 3–18, 2017. 1, 2

[43] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of Machine Learning Models. In *Proceedings of the USENIX Security Symposium*, pages 2615–2632, 2021. 2

[44] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1453–1460, 2011. 5

[45] Javier Tello, Marina de la Cruz, Tony Ribeiro, Julian Fierrez, Aythami Morales, Ruben Tolosana, Cesar Luis Alonso, and Alfonso Ortega. Symbolic AI (LFIT) for XAI to handle biases. In *European Conf. on Artificial Intelligence Workshops (ECAIw)*, 2023. 8

[46] The White House. Memorandum on Advancing the United States' Leadership in Artificial Intelligence., 2024. 1, 3

[47] Europen Union. Artificial intelligence act. *European Parliament: European Parliamentary Research Service*, 2024/1689, (Updated June 2024). 1, 3

[48] R. Veldhuis et al. *Privacy and Security Matters in Biometric Technologies*. Springer, 2025. 1

[49] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in Machine Learning: Analyzing the connection to overfitting. In *Proceedings of the IEEE Computer Security Foundations Symposium*, pages 268–282, 2018. 2