Defending Against Membership Inference Attacks

Eva Azazoglu, Brown University, CSCI2980 Research Report, May 2025

Abstract— **Membership Inference Attacks (MIAs) threaten the privacy of data used in machine learning (ML) models as they allow adversaries to infer whether a specific data point was part of the model's training set. MIAs pose a particular risk in privacy-sensitive domains like healthcare or finance. They extract sensitive information by exploiting the generalization gap caused by overfitting in ML models and can even exploit black-box models. This project investigates the effectiveness of three defense methods against MIAs: dropout, label smoothing, and confidence score masking. It tests these defenses both individually and in combination against the MIA pipeline provided by the *ML-Leaks* framework [1].**

Building on the *ML-Leaks* framework, this project adapts a CNN-based attack pipeline using the CIFAR-10 dataset, and it evaluates various defense combinations by measuring attack accuracy, target model utility, and balanced accuracy. In this setup, dropout most effectively reduced attack accuracy but significantly harmed model utility, while label smoothing offered a more favorable privacy-utility tradeoff as it preserved target accuracy while reducing MIA effectiveness. Confidence score masking, on the other hand, provided minimal additional protection when it was combined with the above regularization techniques. The study's results suggest that combining multiple defenses leads to diminishing returns and that the careful selection of individual high-impact defenses such as label smoothing can meaningfully improve models' robustness against MIAs.

I. Introduction

Machine learning (ML) models are becoming ubiquitous across industries and geographies, where they have come to play integral roles in tasks ranging from speech recognition to disease detection. Among their various applications, ML models increasingly manage complex tasks in domains that handle sensitive data, such as healthcare and finance [2]. The rise of Machine Learning as a Service (MLaaS) provides additional access to ML models, as companies expose their models via APIs [3]. Naturally, it is imperative that we maintain the privacy of such sensitive data and that we ensure its proper protection.

However, ML models have an innate tendency to memorize their training data and overgeneralize when faced with unseen information (known as overfitting), which leaves them vulnerable to a range of attacks aimed at compromising their data integrity. As industries increasingly integrate ML models into their handling of sensitive data, these exploitations pose a growing threat to data protection. Some of these potential attacks include model extraction attacks, attribute inference attacks (model inversion attacks), property inference attacks, as well as membership inference attacks [3].

This research project focuses in particular on **membership inference attacks** (MIAs), which allow the attacker to infer whether a specific data record was part of the training data of a given model [4]. As mentioned above, MIAs are especially problematic in MLaaS offerings, as the exposure of companies' ML models through APIs makes these models vulnerable to MIAs and similar attacks [3]. These privacy violations are particularly dangerous in fields like healthcare, where MIAs can lead to breaches in medical confidentiality [2]. There are additionally important legal concerns to consider, as MIAs can lead to violations of privacy laws such as the European Union's General Data Protection Regulation (GDPR) which could consider the inferences made through MIAs as "unlawful disclosures of personal data" [3].

Attackers use MIAs to infer membership by exploiting differences in how models behave on training versus non-training data, as models often exhibit lower loss values and higher prediction confidence for training data. MIAs are effective precisely because they exploit an inherent weakness of ML models, which is their tendency to overfit on training data and therefore exhibit significant performance differences between training and test data [5]. The success of MIAs is directly linked to the model's generalization error, which is the difference in a model's performance on a training versus a non-training sample that arises as a result of overfitting [4]. Generalization error plays a critical role in MIA success, as higher generalization errors generally make the model more predictable and the difference between training and non-training data will therefore become more easily recognizable [4].

Since the core mechanism of MIAs functions by comparing model outputs (such as prediction probabilities or confidence scores) on different data points, these attacks pose a particularly widespread risk as they also function on *black-box ML models*, which are models where the attacker does not have access to their internal parameters but rather can only observe the model outputs [4]. This, in turn, means that even attackers with limited knowledge or computational power can execute successful MIAs on widely used and available models, which highlights the vulnerability of ML models against these attacks and the importance of implementing strong defenses to mitigate

them.

A. Organization

This report is organized as follows. Section II outlines the conceptual process of MIAs and their defenses by synthesizing existing work. Section III presents the empirical research methodology of this project, while Section IV presents the experimental results and Section V offers a discussion of findings, limitations, and future research directions.

II. Existing Work

This section outlines the principal components of MIAs and synthesizes existing work on MIA mechanisms (including membership scores, attack techniques, and shadow models) and on defense methods. The rationale behind this section is that order to be able to successfully defend against and mitigate MIAs, it is important to clearly understand the core mechanisms involved in these attacks themselves and how previous research has engaged with them.

A. Attack Mechanism

The key assumption on which MIAs operate is that models behave differently for training versus non-training data (as described in Section I) [5]. In general, the model that is exploited in the attack is referred to as the *target model*, while the attacker's classifier model that infers memberships is known as the *attack model*.

Membership score calculation. The core mechanism of MIAs (and the primary objective of the attacker) is the calculation of *membership scores* for given inputs, which are scores that indicate whether a sample was part of the training set for a model. These scores are calculated based on model output metrics such as prediction confidence, the negative loss, and probability distributions [2], where higher scores indicate that the sample is likely part of the training data.

While the evaluation in this project operationalizes this membership score calculation without a specific formula (as we see in Section III), a mathematical abstraction of this process can clarify this process; the simplest illustration for membership score calculation can be found in the *loss-based attack* technique. Thus, for a given input x and a target model M_θ with parameters θ, the loss-based membership score $S(x, y)$ is given by:

$$S(x, y) = -L(M_\theta(x), y) \qquad [2], [4], [5]$$

Here, a lower loss value (therefore a higher negative loss) indicates that the model has "memorized" the training data and therefore exhibits significantly lower loss for training data than for non-training data.

MIA techniques. While the loss-based membership inference formula is the most accessible example, there are several different types of MIA techniques, each focusing on a different metric to compute membership scores while resting on the same underlying principle of membership score calculation. In addition to loss-based attacks (which we have just seen), important techniques include attacks based on *confidence* [2], [3], [5], *metric* [3], *decision boundary* [3], *entropy* [3], [4], *prediction correctness* [3], and *binary classifier* [3].

This project focuses in particular on confidence-based attacks, in which the attacker observes the model's prediction confidence for a given input based on the assumption that models tend to be more confident when they predict labels for their training data [3]; here, higher confidence typically indicates that the input was part of the training set for the model [2].

Shadow models. A common way to mount MIAs is through shadow model training. Shadow models are trained with the specific goal of replicating the target model's behavior, in order to simulate MIAs to allow the attacker to train the attack model effectively [3]. A shadow model D_{shadow} is trained using a shadow dataset that is split into known member and non-member samples:

$$D_{shadow} = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) \qquad [3]$$

Once the shadow model is trained, the attacker then uses its outputs to train a classifier (the attack model) for predicting membership on the target model. Therefore, shadow models significantly enhance the attack's accuracy as they allow attackers to estimate how the target model would treat training versus non-training data [4]. Shadow model training is also a key facilitator of MIA's success against black-box MIAs, as shadow models provide an opportunity to train the attack model on a model that mimics the target model's behavior but whose training data and parameters are known, therefore allowing for finetuning before applying the attack model to the black-box target model [3], [4]. To synthesize the process, the formula for membership prediction using a classifier trained using a shadow model is as follows:

$$\mathrm{M}(x) = \text{Classifier}(\text{Shadow Model Output}(x))$$

where $\mathrm{M}(x)$ represents the predicted membership (based on the membership score) of input value x.

B. Defenses

As we have established, MIAs exploit model overfitting and predictable behavior to infer whether a data point was part of the training set. Numerous defense methods have been proposed and implemented to work towards building more robust protections against these attacks; these

defense methods can be broadly categorized into *model-level defenses* and *output-level defenses*, many of which focus on mitigating the core weakness that is overfitting. This project explores 3 complementary defenses ranging across both categories: dropout, confidence score masking, and label smoothing, and it evaluates their effectiveness alone and in combination with each other. For the sake of concision and relevance, this section focuses only on the defense methods we use in our empirical methods, but further defense methods are provided in this footnote.[1]

First, it is important to distinguish between model-level and output-level defenses as they apply to this project:

- **Model-level defenses** are regularization-based and aim to reduce overfitting in order to minimize behavioral differences between training and non-training data. *Dropout* and *Label smoothing* belong in this category.
- **Output-level defenses** aim to obfuscate the model's output to conceal membership clues. *Confidence score masking* belongs in this category.

Dropout. Dropout is a classical regularization method that aims to reduce overfitting in a model. Dropout involves randomly deleting (dropping) units during training, which prevents the model from relying too heavily on any individual training sample and therefore reduces the likelihood of memorization [4]. Dropout helps mitigate the risk of MIAs by making the model less reliant on individual training data points and more on general patterns in the data [5]. This helps models generalize better (since it reduces their tendency to overfit/memorize the training data), which lowers the generalization gap and therefore makes it harder for the attacker to distinguish between training and non-training data [2].

While dropout does affect the model's accuracy (which is natural since it makes changes to the training data itself), past studies such as Salem et al. have shown that this effect is not strictly negative and that, in some cases, the regularization imposed by dropout even improves after applying dropout [1]. However, this effect on accuracy is a tradeoff of dropout that is important to monitor as we apply it to mitigate MIAs.

Label smoothing. Label smoothing is another classical regularization method that aims to improve the generalizability of an ML model [3]. It works by replacing one-hot labels with a smoothed distribution that blends the true class with a uniform distribution over all classes, which allows it to address both overfitting and overconfidence. [6]. Thus, instead of assigning a probability of 1 to the correct class and 0 to all others, label smoothing replaces the target distribution with a mixture of the ground-truth and a uniform prior; typically, it assigns 0.9 to the true class and 0.1 distributed evenly among all others [7]. This prevents the model from producing very peaked output distributions (i.e. overly confident predictions) with large logit gaps between the predicted and non-predicted classes [7], [6]. This, in turn, reduces the generalization gap and therefore makes it more difficult for the attacker to distinguish between training and non-training data.

However, a key tradeoff introduced by label smoothing is that it may hurt model performance if the labels are correct, especially in cases where we may want confident predictions; this is because label smoothing ultimately affects the model's confidence across all predictions, even for correct ones [6]. If label smoothing is applied too aggressively, this could even lead to underfitting, which is why it is important to track and control the tradeoff as we apply label smoothing.

Confidence score masking. Confidence score masking aims to mitigate the effectiveness of MIAs by hiding the true confidence scores returned by a classifier model. There are several widely used variants of confidence score masking, ranging from adding noise to providing only limited prediction labels [3]. However, this project applies a type of masking in which the target classifier does not provide a complete prediction vector, but rather provides top-k confidence scores to the attacker [3]. For example, in a classification problem of 10 classes, the model would only provide the largest 3 confidence scores when the attacker queries the input record [3]. Once the top-3 confidence values are extracted, the remaining 7 values in the prediction vector are set to zero; however, the resulting sparse vector with only 3 non-zero values would violate the properties of a probability distribution as the top 3 values no longer sum to 1. Therefore, confidence score masking applies renormalization by dividing each of the top-3 retained values by their sum, so that the prediction vector again sums to 1 and the output thereby still behaves like a probability distribution, but with the fine-grained confidence detail hidden from the attacker.

Since this is an output-level defense, the key advantage of confidence score masking is that there is no need to retrain the classifier model, since this method is simply implemented on the output prediction vectors [3]. However, a significant drawback of confidence score masking arises from the fact that it only reduces the ostensibility of the output confidences but does not actually reduce the generalization gap. As long as this generalization gap exists, a simple prediction-correctness MIA will always

[1] Adversarial regularization [[3], p.10]; Adversarial training [[4], p.274] [[2], p.3]; Algorithmic stability [[5], p.273]; Batch normalization [[5], p.272]; Differential privacy [[3], p.8] [[4], p.270]; Difficulty calibration [[2], p.3]; Ensemble models [[3], p.10] [[4], p.275]; L1 and L2 regularization [[3], p.9] [[5], p.273] [[4], p.273]; Model pruning [[5], p.272]; Testing defenses with shadow models [[3], p.10]; Weight decay [[5], p.273]

achieve better attack performance than random guessing [3]. Therefore, while confidence score masking can make it more difficult for an attacker to distinguish between training and non-training samples, it does not fully remove the patterns of difference that result from the model's generalization gap.

C. Contribution

While existing work explores various defense methods in rigorous detail, defense methods are mostly evaluated in isolation. While this allows us to gain a deep understanding of the merits and tradeoffs of individual defense methods, it leaves us with unanswered questions on the potential of combining defenses to achieve synergies in defending against MIAs, but also to reveal new tradeoffs or risks. The contribution of this project lies in its approach of evaluating combinations of defense methods (dropout, label smoothing, and confidence score masking) to take a step towards answering these important questions. In particular, this project works on defense methods against MIAs that target ML classifier models and therefore explores defense methods that work well with precisely these classifiers.

The particular methods that this project uses were chosen for several reasons. First, all three defenses target different vulnerability layers in the classifier model:

- **Dropout** targets internal memorization in the model and aims to reduce memorization.
- **Label smoothing** mitigates overconfidence at the model's output layer and aims to reduce output confidence and sharpness.
- **Confidence score masking** controls model output exposure at inference time and aims to limit exposed signals in the output.

The goal of applying defenses to different vulnerability layers on the same model is to evaluate the effects of targeting multiple levels of the model on attack success. Second, these defense methods present relatively low-risk tradeoffs; while other defenses (such as differential privacy and adversarial training) offer stronger theoretical guarantees, this often comes at the cost of significant utility loss or implementation complexity. In order to be able to evaluate the merits of multiple defense methods, we chose simpler and widely applicable methods that allow us to control the privacy-utility tradeoff as much as possible.

III. Methodology

This section introduces the *ML-Leaks* model introduced by Salem et al. [1] and the basis it provides for this project, before moving to present the CIFAR-10 dataset and its applications for the project. Then, this section describes the experimental setup along with implementation and evaluation.

Two central objectives drive the empirical methods design:

1) **Reducing the attack success rate.** This is the driving motivation for implementing defenses, as our ultimate goal is to prevent data extraction and privacy violations, which we can most immediately measure by measuring the attack success rate (i.e. how accurately the attack model classifies a sample correctly).
2) **Minimizing the privacy-utility tradeoff.** Each defense method affects the target model's performance and thus creates a tradeoff between ensuring increased privacy but also reducing the model's utility. We hope to ensure that building robust defenses against attacks still leaves us with functional models that can perform adequately in their given tasks, so minimizing the privacy-utility tradeoff is a key consideration as well.

A. The ML-Leaks Model

ML-Leaks (Salem et al., 2019) [1] presents a generalized framework for MIAs that reduces prior assumptions on attacker knowledge and models a range of increasingly relaxed threat scenarios. It does so by improving on the original Shokri et al. [4] attack, which was the first work to formally demonstrate and define MIAs against ML models in a reproducible and generalizable format. Specifically, *ML-Leaks* demonstrates that only a single shadow model and a single attack model are sufficient to launch successful MIAs, which significantly reduces the cost and complexity of the attack. The *ML-Leaks* framework supports multiple types of target models (including convolutional neural networks (CNNs) and MLPs), and additionally, it evaluates MIAs across 8 diverse datasets (including CIFAR-10 and CIFAR-100); this flexibility makes the framework both model- and data-independent and widely applicable to different formats.

The attack model in *ML-Leaks* is a binary classifier, and it is trained on the outputs of a shadow model that mimics the target model's behavior by predicting membership based on confidence scores (posterior probabilities), according to the shadow model training process described in Section II-A. Salem et al. introduce 3 adversarial types:

1) **Adversary 1** has access to data from the same distribution as the target model's training data and knows its model architecture. It trains a shadow model based on this information to mimic the target model closely.
2) **Adversary 2** has access to data from a different distribution as the target model, and may also use a different model architecture. This is a weaker adversary with less knowledge of the target model but still trains a shadow model to mount the attack.

3) **Adversary 3** trains no shadow model at all and instead relies on simple heuristics such as prediction confidence or loss for membership inference. While this adversary is the simplest and weakest of the three, it is also the most realistic in terms of resource constraints.

***ML-Leaks* in this project.** This project uses Salem et al.'s work as a foundational paper because *ML-Leaks* offers a reproducible, modular, and broadly applicable pipeline for testing membership inference threats and defenses, as established above. This makes *ML-Leaks* an ideal foundation for implementing and evaluating multiple defense combinations. This project adopts *ML-Leaks*' Adversary 1 setup and thus uses a single shadow model and attack model. The project's implementation reuses core architecture from the *ML-Leaks* GitHub repository,[2] specifically its CNN target model and the attack pipeline. The choice to reuse this core architecture to enable direct comparison to baseline results, and also to reproduce the confidence-based attack mechanism that is instrumental for this project.

Additionally, the decision to focus on CNNs is aligned with Salem et al.'s evaluation on CIFAR-10/CIFAR-100 on convolutional networks. Since this project also adopts the CIFAR-10 dataset [8] (as discussed below), the choice of CNNs allows for a clear replication of the MIA setting and defenses. Finally, this project implements dropout as one of our defense methods following the proposal of this method in *ML-Leaks*, where Salem et al. apply dropout to input and hidden layers [1] to reduce overfitting and therefore the generalization gap that is so central to how MIAs exploit ML models.

Limitations of *ML-Leaks*. In the selection of *ML-Leaks* as the basis for this project, an important limiting consideration was the framework's age; it is over 6 years old, which in the rapidly evolving field of machine learning means that it cannot incorporate more recent attack models. However, despite its age, *ML-Leaks* remains an influential and widely reproducible foundational work in the field for black-box MIAs, and the clear experimental structure it offers allows for an ideal environment for isolated and combined testing of multiple defense mechanisms.

B. Experimental Design

The experimental design takes the *ML-Leaks* framework as its basis and builds the defense methods upon this foundation. The design overview in this section covers the dataset selection, implementation details, and evaluation metrics.

Dataset. This project uses the CIFAR-10 dataset.[3] This dataset consists of 60,000 color images of size $32x32$ pixels across 10 classes, with 6,000 images per class and a standard train/test split of 50,000 and 10,000 respectively [8]. CIFAR-10 is widely used in ML research; this is not only thanks to its moderate complexity but also because it is suitable for image classification benchmarks, especially in CNNs, which makes it particularly suitable for this CNN-based project. This project uses the CIFAR-10 dataset in order to align with the original *ML-Leaks* evaluation, and to ensure results that are reproducible and comparable within a well-established benchmark dataset. Additionally, its manageable size and balanced class distribution also make CIFAR-10 a practical choice for conducting extensive experiments across multiple defenses without hitting significant time or resource limitations.

Implementation details. All code for this project builds upon the public *ML-Leaks* GitHub repository[4] and it introduces extensive modifications for the following goals:

1) **Implement defense methods.** This includes dropout as well as the new defense methods of label smoothing and confidence score masking.
2) **Improve experimental tracking.** This involves auto-saving model outputs as `.npz` files and clearly logging output and evaluation metrics per run.
3) **Introduce modular configuration.** This involves restructuring the codebase to enable toggling and combining defenses without rewriting any core pipeline logic.

The base CNN model and attack pipeline are preserved from the original code; this includes the shadow model training, attack model architecture, and confidence-based membership inference. **Dropout** is implemented as a model-level defense inserting dropout layers at both the input and fully connected layers of the CNN, which is in accordance with the original dropout proposal in *ML-Leaks*, and the dropout rate was fixed at 0.5 during training. The implementation relies on Theano 1.0.5 and Lasagne, which are legacy deep learning libraries that the original *ML-Leaks* codebase uses. Accordingly, the environment was set up using Conda on macOS ARM64 (Apple Silicon), with Python 3.8 to ensure compatibility with Theano and Lasagne. Similarly, and to resolve deprecation issues during setup, this implementation uses manually configured dependencies, including legacy versions of NumPy. The full implementation code and experiments are available at the project's GitHub repository.[5]

We implement **label smoothing** by modifying the target labels into smoothed soft-label vectors before training and using a smoothing parameter of 0.1. This implementation then involves updating the training function to accept and

[2]https://github.com/AhmedSalem2/ML-Leaks
[3]https://www.cs.toronto.edu/~kriz/cifar.html
[4]https://github.com/AhmedSalem2/ML-Leaks
[5]https://github.com/eva1azaz/CS2980-MIA-project

process one-hot vectors with adjusted probabilities. To reduce runtime during training with label smoothing, the smoothed labels are precomputed for the entire training set rather than generating them on the fly for each minibatch. To achieve this, we modify the `iterate_minibatches` function to return sample indices, which then allows each minibatch to directly index into the precomputed smoothed label array and thus significantly improves efficiency. Then, **confidence score masking** is implemented post-training by modifying prediction vectors to retain only the top-3 confidence values per prediction and then renormalizing the prediction vector. We recall from Section II-B that there are several widely used variants of confidence score masking. In this implementation, the top-3 confidence masking was chosen for its ability to retain partial predictive information by suppressing fine-grained leakage of confidence scores without fully discarding informative outputs, with the expectation that this variant in particular is most likely to achieve a balance between privacy and utility.

Finally, we evaluate all three defenses individually as well as in combinations (iterating through all possible combinations, e.g. dropout + masking, dropout + label smoothing, all three combined, etc.). While target and shadow models are retrained for each combination of defenses involving dropout and label smoothing, confidence score masking is applied post-training. When running these combinations, each run (target, shadow, attack) automatically saves output data and model parameters to uniquely named `.npz` files, so as to facilitate reloading each run for subsequent confidence score masking or analysis without needing to retrain. All models were trained for 50 epochs using a batch size of 100 and a learning rate of 0.01, unless otherwise specified. These exhaustive combinations allow us to measure the individual impacts of each defense method, as well as the synergistic impacts that combinations of defenses may have on both utility and attack success.

C. Evaluating MIA success

There are several commonly used metrics to evaluate the success of MIAs, including balanced accuracy [2], AUC-ROC [2], and True Positive Rate at low False Positive Rate [2]. In addition to comparing attack accuracy across defenses, this project primarily uses **balanced accuracy** to evaluate attack success when faced with different defense methods. Balanced accuracy is the simplest method to evaluate attack efficacy, and it measures how often an attack correctly predicts membership on a balanced dataset of members (class 1) and non-members (class 0) by computing the mean of sensitivity (true positive rate, or recall for class 1) and specificity (true negative rate, or recall for class 0) [2]:

$$\text{Balanced Accuracy} = \frac{1}{2}(\text{Recall}_{\text{class 0}} + \text{Recall}_{\text{class 1}})$$

Balanced accuracy was chosen for its ability to compare fairly across defenses. While defenses such as dropout reduce both member and non-member confidence indiscriminately, others like confidence score masking target only members; balanced accuracy accounts for both sides of the inference and therefore provides a fairer metric for comparing these tradeoffs. Therefore, balanced accuracy reflects the success of the defense methods in narrowing the generalization gap that MIAs exploit, which makes it a valuable evaluation metric. Additionally, balanced accuracy can handle class imbalance (if there is not a perfect balance of member to non-member samples), and it also offers an improvement from using plain accuracy or precision, which could potentially misrepresent imbalanced datasets.

As discussed in Section II-B, this project aims not only to reduce MIA success but also to optimize the privacy-utility tradeoff when implementing defense methods. Therefore, this project also records accuracy scores for the target model as well as the shadow model and the attack model, and compares these scores across the different combinations of defense methods that are applied.

IV. Results

This section presents the results of the empirical study outlined in Section III, covering baseline results, each defense in isolation, and finally the combined defenses, and commenting on trends and observations. Section I summarizes the target and attack performance along with balanced accuracy across all the different defense configurations. After running the model on each combination of defense methods discussed in Section III, we recorded the most relevant values for this study:

1) **Target accuracy.** This measures the accuracy of the target model in classifying samples correctly among 10 classes; it is a core indicator of utility, where higher target accuracy means that the defense preserves accuracy more effectively.
2) **Attack accuracy.** This measures the accuracy of the attack model in distinguishing between training and non-training samples in the target model; it is a core privacy metric, where a lower attack accuracy indicates that the model has higher privacy.
3) **Balanced accuracy.** This measures how often the attack correctly predicts membership; it is an indicator of attack fairness across member/non-member classes, and is useful to detect class imbalance effects [2]. Balanced accuracy is computed from recall for class 1 and recall for class 0 according to the formula outlined in Section III-C.

Defense(s)	Target Accuracy	Attack Accuracy	Balanced Accuracy	Recall (class 1)	Recall (class 0)
None (Benchmark)	0.60	0.82	0.83	1.00	0.65
Dropout	0.49	0.51	0.51	0.69	0.33
Smoothing	0.60	0.72	0.73	1.00	0.61
Masking	0.59	0.80	0.81	1.00	0.45
Dropout + Smoothing	0.60	0.73	0.73	1.00	0.46
Dropout + Masking	0.58	0.81	0.81	1.00	0.62
Smoothing + Masking	0.60	0.74	0.74	1.00	0.48
All Three	0.60	0.73	0.73	1.00	0.46

TABLE I
RESULTS SUMMARY

4) **Recall for class 1 (members).** This is the true positive rate (sensitivity). It asks: out of all actual members, how many were correctly predicted as members?
5) **Recall for class 0 (non-members).** This is the true negative rate (specificity). It asks: out of all actual non-members, how many were correctly predicted as non-members?

The defense combinations for each run, along with their recorded accuracy values in each of the above categories, are recorded in Table I.

Benchmark. First, we consider the benchmark (with no defense) and note that the attack accuracy is at 0.82, which indicates a highly effective MIA (as the goal of the defense implementations is to reduce this value). The balanced accuracy at the benchmark is 0.83, which means that the attack model identifies members (class 1 recall) perfectly, while only identifying 65% of non-members (class 0 recall) correctly. This asymmetry further highlights that there is a large generalization gap in the target model and that this facilitates a strong attack. On the other hand, the target accuracy at the benchmark is 0.60, and we hope to keep this as high as possible to maintain the target model's utility and therefore minimize the privacy-utility tradeoff from the defense implementations.

A. Defenses in isolation

Dropout. As we can see, dropout alone significantly reduces attack accuracy, with the implementation of dropout reducing the attack accuracy from 0.82 to 0.51, which means that the attack model is barely performing better than random guessing (which would be 0.50). Looking at the balanced accuracy value of 0.51 (compared to 0.83 at benchmark), this confirms the effectiveness of dropout, as it reduces the model's generalization gap so well that the attack model struggles to distinguish members from non-members in both classes, leaving the attack model to barely more than random guessing. However, we note that dropout significantly reduces the target accuracy as well (from 0.60 to 0.49), which indicates that this increased privacy comes at the cost of model utility.

Label smoothing. While label smoothing does not lead to the same drastic reduction in attack accuracy as dropout, we can still observe a significant drop in attack accuracy at 0.72 compared to 0.82 at the benchmark. The balanced accuracy of 0.73 is a significant reduction from 0.83 at the benchmark, which indicates that the attack is meaningfully less effective across both classes (as we can also see in the Recall columns). While the drop in attack accuracy seems moderate, the balanced accuracy confirms that this really is a two-sided defense (affecting both members and non-members) and does not just skew predictions towards one class. Crucially, label smoothing maintains the target accuracy at 0.60 and therefore, unlike dropout, leaves the target model's utility intact.

Confidence score masking. While confidence score masking also largely maintains the target model's utility by barely affecting the target accuracy (0.59 compared to 0.60 at benchmark), it only reduces attack accuracy by a small amount (0.80 compared to 0.82 at benchmark) and therefore did not seem to have a significant impact on the attack mechanism. Significantly, the balanced accuracy for confidence score masking is barely lower than the benchmark, at 0.81 compared to 0.83 with no defense. Therefore, despite the slight reduction in attack accuracy, we can infer from the balanced accuracy score that the attack model

still performs almost perfectly on at least one class; this is indeed confirmed when we look at the recall columns, which indicate that the attack model performed perfectly on class 1 (members). This suggests that confidence score masking does not significantly reduce the generalization gap in the target model. That is not surprising, given that this defense only hides the confidence vector but does not affect the model's internal learning.

B. Defenses in combination

Dropout + label smoothing. The combination of dropout and label smoothing produces very similar results to label smoothing alone, with an attack accuracy just slightly above that of label smoothing (0.73 compared to 0.72), while maintaining almost exactly the same values for target accuracy and balanced accuracy as label smoothing. This seems to suggest that the combination of these two defense methods does not augment their individual impacts, and the influence of dropout specifically does not seem to be clearly reflected in this combination.

Dropout + confidence score masking. The combination of dropout and confidence score masking yields an attack accuracy of 0.81, which is higher than dropout alone (0.59) and around the same as masking alone. This result suggests that confidence score masking may dominate in this combination and the most substantial effect of dropout is not preserved here. For balanced accuracy, we also see that it is significantly higher than for dropout alone (at 0.81 compared to 0.51), which indicates that the attack model still performs well for at least one class (and we can see in the recall columns that it performs perfectly on members). Therefore, this combination does not improve over either individual defense method in isolation, and it may even suffer from conflicting or redundant effects.

Label smoothing + confidence score masking. The combination of label smoothing and confidence score masking produces nearly identical results to label smoothing alone, with attack accuracy and balanced accuracy both at 0.74 (compared to 0.72 and 0.73 respectively for smoothing alone), along with the same target accuracy as smoothing alone at 0.60. Here, it seems that label smoothing dominates and carries over most of the defense impact, whereas confidence score masking seems to contribute little additional protection once label smoothing has already weakened the model's confidence distribution.

All three combined. Interestingly, adding confidence score masking to the dropout + label smoothing combination does not improve results further, as we can see that the attack accuracy remains at 0.73 and balanced accuracy is also unchanged at 0.73. This absence of change underlines the observation that when confidence score masking is applied in addition to strong regularization methods like dropout and label smoothing, it has limited marginal benefit and thus diminishing returns.

C. General observations

In general, with respect to the privacy-utility tradeoff, we can observe that the strongest defenses seem to appear in combinations that include label smoothing. Among all the different evaluated combinations, label smoothing in isolation offered the most favorable privacy-utility tradeoff, as it reduced attack accuracy to 0.72, while preserving full target accuracy at 0.60 and also maintaining class balance with a balanced accuracy of 0.73. While dropout in isolation was the most effective method in suppressing attack accuracy, this came with a high cost to the target model's classification performance and thus comes with a sharp privacy-utility tradeoff. Not shown in Table 1 but still recorded in the model output were shadow model accuracy scores for each combination; these stayed constant around the range of 0.60-0.61 for all combinations except dropout in isolation, where it dropped to 0.46 and thus further reinforces our takeaway that the privacy benefits of dropout come with significant utility costs. Finally, confidence score masking offers minimal additional benefit when combined with the other two methods (which are both strong regularization techniques) and shows diminishing returns in combination; this suggests that the benefits of confidence score masking are more focused on the defense method on its own, or perhaps in combination with weaker defenses.

V. Discussion

This section discusses and contextualizes the results and key takeaways from this project, including interpretation of results, challenges and limitations, and directions for future research.

Interpretation of results. As we saw in the results analysis, a key observation is that additional defenses have diminishing returns and do not offer significant improvements over defense methods in isolation. For example, the results clearly show that confidence score masking does not offer additional protection when strong regularization methods like dropout and label smoothing are already applied (even though confidence score masking does improve privacy when applied in isolation, which highlights the diminishing returns of combining defenses). Adding output-level defenses to model-level defenses (like regularization) may thus have only a marginal benefit, if any.

When contextualizing results, it is additionally important to consider practical applications of defenses and particularly the privacy-utility tradeoff as a primary driver of defense method selection. As we know from the results analysis, dropout in isolation can significantly reduce attack

accuracy (and therefore improve privacy) but this comes with a utility cost (lower target accuracy) that would deter those who want to maintain functional ML models when implementing privacy protections.

Balanced accuracy (and the recall values that constitute it) provide additional clarity on these interpretations as they indicate whether defenses work equally well across members and non-members. This is an additional important consideration when evaluating defenses, as the balance between a model's performance on member and non-member classes is a key factor in determining fairness and analyzing the generalization gap of a model. Considering these priorities, label smoothing stands out as an effective defense method as it provides significant and balanced privacy improvements with minimal utility loss.

Challenges and limitations. The implementation of this project came with several technical challenges. First, training dropout and (especially) label smoothing was computationally expensive; while label smoothing optimizations somewhat circumvented this problem and reduced the runtime to a reasonable scale, this may have affected the performance of label smoothing and our subsequent analysis. Additionally, as discussed in Section III-A, the age of the *ML-Leaks* frameworks caused limitations with the legacy code, as Theano/Lasagne and *ML-Leaks*' older architecture posed compatibility challenges and limited some modern techniques (e.g. preventing the implementation of recent adversarial training techniques).

ML-Leaks' age of over 6 years, as well as the specialized implementation on CNNs and using CIFAR-10, limit the scope of applicability of these results to other domains (such as medical or financial data) and other model types (such as transformers). Additionally, this framework does not reflect more recent advances in the rapidly evolving field of ML and newly developed defenses. However, while these constraints do limit the generalizability of these specific results, the flexible and reproducible environment for isolated and combined testing of multiple defenses still allows us to gain valuable insights into the synergies of these defense mechanisms. Building on these insights, future works can apply them to more current attack frameworks and wider domains.

Future work. As just discussed, the flexibility of the *ML-Leaks* framework makes these results a valuable basis for further research. Future work could extend these experiments to newer MIA frameworks (such as difficulty calibration or RAPID [2]), and it could apply the defense combinations explored in this project to non-vision datasets, or even alternative new model types such as transformers or LLMs. In terms of evaluation metrics, finer-grained metrics such as True Positive Rate at low False Positive Rate [2] could be integrated into future studies to add an additional layer of evaluation to defense methods.

Additionally, this project's findings with reference to the marginal benefits of adding output-level defenses to strong model-level defenses (i.e. adding confidence score masking to dropout and/or label smoothing) opens the door for further experimentation. In particular, it could be valuable to test other output-level defenses such as different variants of differential privacy or adversarial noise injection, and to compare their effects to this project's observations on confidence score masking. Finally, and considering the pivotal role of the privacy-utility tradeoff in this project's methods and conclusions, future work could explore additional methods to optimize the tradeoff and continue building towards effective MIA defenses.

REFERENCES

[1] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, 2019.

[2] Y. He, B. Li, Y. Wang, M. Yang, J. Wang, H. Hu, and X. Zhao, "Is difficulty calibration all we need? towards more practical membership inference attacks," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1–20, [Online]. Available: https://doi.org/10.1145/3658644.3690316.

[3] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1–37, 2022, [Online]. Available: https://doi.org/10.1145/3523273.

[4] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, 2017, pp. 3–18.

[5] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, Oxford, UK, 2018, pp. 268–282.

[6] T. D. Science, "What is label smoothing?" https://towardsdatascience.com/what-is-label-smoothing-108de2ecdc6d, 2020, accessed: 2025-05-11.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[8] A. Krizhevsky, "Learning multiple layers of features from tiny images," https://www.cs.toronto.edu/~kriz/cifar.html, 2009, accessed: 2025-05-11.