Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,
Arbeitsgruppe ID Management

The Influence of Training Parameters and Architectural Choices on the Vulnerability of Neural Networks to Membership Inference Attacks

Oussama Bouanani

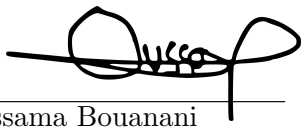
oussama.bouanani@fu-berlin.de
Matrikelnummer: 5218302
Betreuerin: Franziska Boenisch
1. Gutachter: Prof. Dr. Marian Margraf
2. Gutachter: Prof. Dr. Gerhard Wunder

Berlin, den 28.04.2021

Eidesstattliche Erklärung

Ich versichere hiermit an Eides statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 28.04.2021



Oussama Bouanani

Abstract

Machine Learning (ML) classifiers have been gaining in popularity and use in recent years as they offer the ability to solve complex decision-making problems in a variety of fields when trained with the appropriate data. These ML models however pose a privacy threat when they learn from private data. Membership Inference Attacks (MIAs) present a privacy breach where given a data record and a query-access to a target classifier, an adversary attempts to determine whether the record was a member of the target's training dataset or not. Previous MIA-relevant work handled various aspects of implementing the attacks and evaluating it but lacked in studying in the impact that individual architectural choices have on the vulnerability of a models to these attacks. Therefore it is the interest of this thesis to explore the role that training parameters and architectural options play in the resiliency of a model to MIAs. In this work, a series of experiments are executed on one baseline model where for each experiment, the baseline is trained on different variations of one single training parameter and different MIA types are launched on each variation to measure the impact of each architectural choice on the baseline's exposure to the attacks. Results show that architectural choices that tackle overfitting by decreasing the gap of a model's performance between training and validation data provides more protection against MIAs. An analysis in the most consistent tendencies in all experiment variations highlights the generated loss values from a target model's prediction as the main factor that allows a MIA to determine the membership status of a data record, as results also show that MIAs are more successful when they are able to learn to differentiate between the distributions of loss values between member and non-member samples.

Contents

1. Introduction

1.1. Motivation

Machine Learning (ML) classifiers are models that are able to learn consistent patterns in large dataset with the goal of making efficient predictions for data records that it has not encountered during training. This technology offers a wide variety of utility that includes solving computer vision problems [70], analyzing health data [4], speech recognition [24] and advertising [8]. In recent years, ML has been gaining in use and more organizations are now opting to use ML as a service (MLaaS) platforms, provided by big players such as Google [5], Microsoft [12] and Amazon [22], to outsource their complex ML training tasks, making ML a key player in the decision-making process of modern businesses.

This increase in popularity has been also met with a concern for privacy, especially in the case where sensitive and personal data is used to train a model and the model leaks information about the private data it trained with. Recent research [1], [46], [41] [40] has investigated these privacy leaks and identified a variety of potential attacks that a malicious party can deploy. These privacy breaches include *Model Inversion* [23] attacks that aim to reconstruct data records used in the training of a target model, *Property Inference* [47] attacks that attempt to extract global trends and sensitive properties of a training set of a model and Membership Inference Attacks (MIAs) which is the main focus of this thesis. MIAs manifest in the ability of an adversary to determine whether a particular record has been used to train a target classification ML model or not, usually with the adversary only being able to query the target model with individual data records and receive predictions as a response. In the context of personal medical or financial records, this membership knowledge can be leveraged by malicious parties to harass or part-take into discriminatory behavior against the persons tied to the data records.

Introduced by Shokri et al. [55], different experimental environments were set up to understand what type of information the attacks were relying on in addition. Past research has been also involved in offering mitigation strategies to protect from MIAs which ranged from architectural and training choices with the aim of improving the model's predictive performance on data outside of its training set. Past experiments targeting MIAs handled different ML model architectures, a variety of

datasets and various training choices specific to each work. This has naturally led to different results, in terms of the efficiency of the deployed MIAs which eventually led to researchers suggesting various counter measures, which are tied to specific experimental setups. A major aspect that lacked was the studying of the impact that individual training and architectural choices can have on the model's overall resilience to MIAs. Therefore, it is the goal of this thesis to focus on MIAs and specifically how the architecture or training processes of a model can increase or decrease its vulnerability to these attacks.

1.2. Contributions

Initially, basic ML-related terms are introduced including the architectural details, training process and parameters that present the aspects handled in the experiments. The focus here is centered around NNs and particularly Convolution NNs (CNNs) [53] since that is the main architecture that will be considered in experiments. CNNs are characterized by their use of Convolution and Max Pooling layers and known by their wide-use in computer vision problems [11]. It is important to note that the scope of this thesis covers the impact of NN architecture and training processes on the success of MIAs meaning that techniques such as Differential Privacy (DP) [20], which manipulates training data by introducing randomness to it, are not to be considered. An overview of ML-related privacy attacks is also provided including a detailed take on MIAs and the concepts it revolves around.

A deep dive into the most relatively recent research then explores the developed MIA techniques and the most interesting findings. Starting with the introduction of this type of attacks by Shokri et al, this work discusses different techniques of the attack, the limitation of their implementations, suggested counter-measures and factors deemed by the relevant research to have influence on a model's vulnerability to MIAs.

The thesis then evaluates the known MIA frameworks in order to select one that offers high-performing attacks which can be directly launched on models built in TensorFlow (TF) 2 [60]. This research led to TF Privacy Tools [61] which is a library built by the TF developers where privacy-assessing and protective tools are implemented. Further details on the implementations and configurations of the attacks are discussed in Chapter 4.

A baseline model is then defined and experiments are carried out where MIAs are launched on 94 trained variations of the baseline, where each experiment only deals with one single architectural or training aspect. Experimental results have shown that certain training choices could indeed have an influence on the model's vulner-

ability to MIAs. However, this influence of the architectural decisions manifested more clearly in a variety of consistent tendencies among all experiments. MIAs were generally more successful in inferring the membership status of misclassified data especially in models whose performance on unseen data suffered in comparison to the performance on training data. Overall, one of the main findings in this work is that a model was more prone to MIAs if it had significant differences in the produced loss values between member and non-member instances which the attack learns allowing it to set a separating decision boundary for the loss accordingly.

2. Background

2.1. Machine Learning Background

Let T be a classification machine learning (ML) model that trains over a dataset D_s composed of features X and labels Y by optimizing a set of learnable parameters that are responsible for its classification decisions. X holds features that T can use to find patterns and learn with while Y represents the classes that each element in X can belong to. Each data record only has one class. In a binary-classification problem, there are only two classes while a multi-classification problem can have $N > 2$ classes. Y is usually represented via a list integers that uniquely identify the classes.

D_s is split into two subsets: Training dataset D_{train} used to train T and D_{test} used to determine the model's performance on unseen data, D_{test} can be referred to as either the test or validation dataset. The model's task is to map a data record $x \in X$ to a class $y_{pred} \in \{0, 1, ..., N-1\}$. It can achieve this by feeding inputs from X_{train} and tuning its parameters with a goal to reach a state where the predicted class by the model matches the true class of the data record such that for a data record $(x, y) \in (X_{train}, Y_{train}), y_{pred} = y$. T learns by finding statistical patterns in X_{train} that it can leverage to identify specific features that helps it predict the correct class. After training for multiple iterations, T then attempts to predict the classes of records from X_{test} and its general performance (referred to as test or validation accuracy) is then evaluated depending on its success in its correct classification of validation instances. Training T is part of *Supervised Learning* which is a category of ML learning types that relies on labeled data that represent the possible prediction values unlike *Unsupervised* Learning where data is unlabeled. A major aspect in supervised learning is avoiding overfitting, it prevents the model from generalizing well to the problem it is solving. This happens due to the presence of noise in the training dataset, in the form of learnable patterns that do not really represent the true properties of the data and its corresponding classes. Several techniques can be employed to reduce overfitting [68].

One of the major aspects of training your machine learning model is avoiding overfitting. The model will have a low accuracy if it is overfitting. This happens because your model is trying too hard to capture the noise in your training dataset. By noise

it is meant the patterns that do not represent the true properties of your data, but random chance. Training T is part of *Supervised Learning* which is a category of ML learning types that relies on labeled data that represent the possible prediction values unlike *Unsupervised* Learning where data is unlabeled.

Deep Learning (DL) is a form of ML that uses a type of ML models called artificial Neural Networks (NNs). A NN is a connection of a series of artificial neurons where one can represent it as a graph composed of nodes representing the neurons and edges representing the connections between the neurons. Each NN has one input layer where input data is fed, an output layer that returns the model's prediction (In our case of a multi-classification NN, the predicted class or predictions of the probability that the input record belongs to each possible class), and thirdly one or more hidden layers in-between. The more hidden layers a NN has, the deeper it is. The learnable parameters of a NN are the weights assigned to the connections between neurons which can be increased or decreased to tune the final predictions of the NN.

A Convolution Neural Network (CNN)[53] is a deep neural network designed for specific tasks such as image classification [11] or object detection. In CNNs, there are two main parts. The first being the Convolution layers that split various features of an input image where each neuron in the layer is responsible for analyzing a small region or feature. The goal here is to allow the neurons in Convolution layers to extract the features that allow to identify each class from other classes. A Convolution layer is usually combined with a non-linear activation function [44] and Max Pooling layers. The latter layer is used to reduce the output of the Convolution layer with the goal of retaining the most important extracted information. Both the Convolution and Max Pooling layers have a kernel or filter size that represents the size of the region that each neuron in the layer would get its input form. The kernel plays the role of a sliding window that goes through an input. This combination of layers is typically repeated multiple times in a CNN model successively before the second main part, the Fully Connected (FC) or Dense layers block, is reached. In a FC layer, all outputs from the previous layer are connected as input to each neuron of the FC layer, unlike a Convolution layer where each neuron gets input that represent a specific region from the output of its previous layer. The FC layers use the output of the convolution layers to predict the class that the input sample belongs to. The output of the FC layers block is a vector of probabilities of the likelihood of the input sample belonging to each class.

2.2. CNN Architectural Choices and Training Parameters

Experiments in this thesis, formally described in chapter 5, involve handling a variety of architectural choices and training parameters and techniques related to NNs and CNNs specifically. This section aims to introduce the concepts behind these architectural choices.

Training Epochs. A training epoch is where the entire training dataset is passed forward and backward through the whole model once. This allows to train the network by optimizing the learnable parameters in order to improve the prediction accuracy of training samples. The optimal amount of training epochs depends mostly on the complexity of the neural network and the training dataset, usually the more complex one of these factors is, the more epochs are required to reach a good general performance. However, too many training epochs would lead to overfitting, where the model is much better at predicting samples that it has seen than ones that it had not encountered during training.

Batch Size. During training, the training dataset is split into small batches that such that each batch of training samples is passed forward and backward through the model once during each epoch. According to recommendations in works from Masters et al. [39] and Bengio et al. [3], using mini-batches has shown to provide improved generalization performance while also being able to leave a smaller footprint in memory.

Activation Functions. Activation functions introduce nonlinearity to a NN model. A neuron that receives input through a previous layer uses an activation functions to calculate the weighted sum to the input it was provided, adds a bias to compute the output of that neuron which is fed to neurons in subsequent layers. The weights of the input sum and the bias are all parameters that are tuned during the training of the model, they represent the learnable parameters in a NN. The values of these weights and biases in the network allow to determine which neurons are "activated" when they are fed a certain input by computing an output value that influences the computations of activation functions in subsequent neurons.

Loss Functions. ML models learn by evaluating how good its predictions are for a set of training data and updating its learnable parameters accordingly. Loss functions are used to quantify the errors made by the model in its prediction meaning that the higher the loss is, the more predictions deviate from the true values. An effective

learning procedure decreases the loss value within each training epoch as the model becomes better at producing predictions.

Optimization Algorithms. Tuning the weights of a ML model depends on computing a loss function on the predictions of training samples after feeding them (usually batch-wise) into the model as part of the forward pass and then computing the gradients for each learnable parameter according to the obtained loss. These gradients represent the direction of change to update the weights during the backwards pass. The parameters are updated according to the gradient by a rate set via a hyperparameter called learning rate. This type of optimization algorithm is called gradient descent [48]. One major issue that faces gradient descent is local minima trapping which means achieving a local minima for the loss value for which the gradient is zero, meaning weights will no longer be updated and the absolute global minimum value that would allow to tune the weights to get the best possible performance out of the whole network can no longer be obtained. To avoid local minima trapping, different optimization algorithms [7] have been proposed.

Data Augmentation. The goal of augmenting training data samples is to artificially inflate the training dataset by applying transformations that preserve the features that define each sample's class, this is especially useful when only a limited amount of training samples is available. These transformations can be categorized into groups: Geometric and Photometric. Geometric data augmentation alters the geometry of the input image with the aim to make T invariant to changes in position and orientation of samples in D_{train} while Photometric transformations amend the color channels with the objective of making T invariant to changes in lightning and color in training images.

Weight Initialization. Learnable parameters are the weights and biases spread across nodes in the layers of the NN. Since the ML model learns by tuning the values of these parameters during training with the goal of reaching a set of weights that allow for useful predictions. Initializing these parameters means setting a starting point for the learning and optimization process.

Weight Regularization. Regularizing weights aims to avoid overfitting by shrinking the weights towards zero to discourage a model from learning complex patterns that are attributed to the training data records only which can diminish a model's general performance. Weight regularization techniques involve adding a term multiplied by a regularization parameter λ to the loss function of the ML model and the weights are updated according to the now appended loss during the backwards passes in training

which effectively decays the weights towards zero. The larger the value of *lamda*, the more weight values will be penalized where for $lambda = 0$, the regularization has no effect at all.

Batch Normalization. Normalizing the input data of neural networks to zero-mean and constant one-unit standard deviation is beneficial [52] for the training of neural networks. Batch Normalization (BN) extends the latter idea further by applying normalization across batches of training data across hidden layers in a NN. BN solves an instability problem in deep NNs [29] by normalizing the outputs of activations to avoid having them become too large which causes large updates in the backwards pass during training. BN layers include a momentum parameter [69] that controls the strength of the normalization, which is usually the most effective when used with small batch sizes.

Dropout. Dropout is yet another regularization technique for NNs. When implemented, it randomly disables the activation of individual nodes while allowing others to remain active during each forward pass in training. This trains the NN to not rely heavily on a particular set of nodes to make its prediction and that can be reflected by having a NN to rely on the whole network by avoiding the assignment of high weights to certain nodes. Dropout is implemented per layer, it can be used on any or all hidden layers and the input layer but not the output one. The probability of keeping or hiding each node is determined via a threshold that can be individually set as a hyper-parameter for each Dropout layer. It is worth mentioning that this technique is only used during training as it is later removed for prediction. The original paper [27] that proposed Dropout proposed using it on FC layers but not on Convolution layers while more recent research [45] was successful in proving the utility of Dropout with low threshold values in $[0.1, 0.3]$ applied after the activation function of each Convolution layer.

2.3. ML Privacy Attacks Background

Before presenting some of the most relevant threats that ML models face, basic data-privacy concepts would be defined. An Adversary or Attacker A is a malicious entity that attempts to extract sensitive information from a target model T. Let D_s be the dataset used to train T and let (x, y) be a data record that belongs to the population of data from which the elements of D_s were sampled from. The type of access that A has to T can differ:

- Black-Box [14] (query) access: A feeds T features x of a data record and T

returns an output y_{pred} as a vector of size N which includes the probabilities that x belongs to a class in $\{0, 1, ..., N-1\}$. In this case, A has no knowledge about the training or implementations of T and it can only rely on providing input, which can be synthetically generated to serve specific purposes, and interpret the outputs to gain knowledge about sensitive information.

- White-Box [14] access: A has some information regarding T and in some cases D_s as well. This information can be in relation to the training parameters, architecture of T, global trends in D_s or any other insider insight. A can also query T to learn more through the outputs it provides but A usually caters their attack depending on the information available to them about T.

MIAs in this work are also limited to passive attacks where A does not change anything in T or its training procedure when it performs its attack, unlike active attacks where, for example, an adversary in a federated learning [66] setting is sending aggregate updates with an intent to push the collaborative learning into a direction that allows them to extract sensitive information from other parties participating in the same collab-learning.

The focus of this work is centered around MIAs. The task of a MIA is to infer whether a sample was used in the training of a target model, i.e. was part of its training dataset. In a black-box setting, attacker A can only query the target T and use the prediction outputted by T in order to differentiate between training set member and non-member samples. Figure 2.1 provides a simple illustration of a black-box MIA. In a white-box setting, A would have access to more information regarding the training and architecture of T and can therefore develop and build their attack with more insights on how T produces the prediction output. Membership inference represent a serious privacy breach. For example, when dealing with a health-related dataset which can include *Magnetic Resonance Imaging (MRI)* scans, a medical imaging technique used in radiology, or test results for certain individuals. If this data is used to train a classifier where A can successfully discover whether a specific record was used for training or not then this attack would inherently leak sensitive information about the individual tied to the data record. A successful MIA is an indication of information leakage through a model and this breach can also be a gateway to further privacy-breaching attacks.

As MIAs represent the main focus in this thesis, the next section that describes related work presents the most recent relevant work that dealt with building MIAs and evaluating their performance under different circumstances.

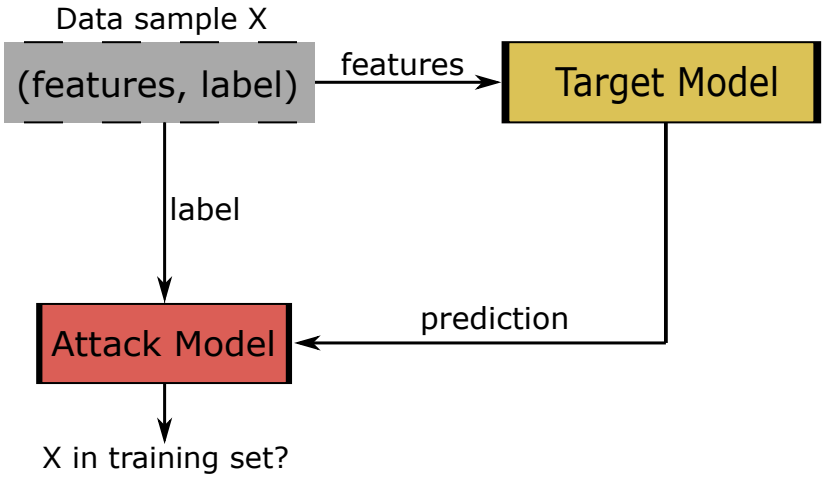


Figure 2.1.: Membership Inference Attack in a black-box setting. The attack queries the target model with features of a data point X and receives the prediction to use it alongside the label to make a decision whether X was used in the target's training dataset or not.

3. Related Work

3.1. Introduction of MIAs

MIAs were brought to the limelight when Shokri et al. [55] developed the first MIA against two classification models while only having black-query access to it. The first target model was constructed using a cloud-based MLaaS platform (for which architecture and hyper-parameters were not specified) and the second one was locally implemented as a CNN trained on a CIFAR dataset. The attack employed a technique called Shadow Training that involved creating multiple shadow models whose goal is to imitate the behavior of the target model but for which data membership status is known (since the data used to train the shadow model themselves is known). The attack model is then trained on the set of generated shadow models to learn to infer the membership status of individual data samples when fed to a particular target model. Figure 3.1 illustrates the process taken by the authors to put the MIA together.

This technique comes with a limiting assumption that requires the shadow models to be trained on data that comes from the same distribution as the training dataset of the eventual target model. If data from the same distribution is not available then one can synthesize it either by using statistics from the population of the target model's training data or by using synthesizing techniques that involve querying the target model. During their experiments, Shokri et al. sampled from the same dataset, satisfying the assumption, to train the target and shadow models but they allowed no overlap between both training sets. In addition, the goal of the shadow models is to imitate the target model. Therefore, it is advantageous for the adversary to design shadow models to resemble the target as much as possible, depending on the information available regarding the target.

Notable results include achieving a 94% and 74% attack accuracy on a multi-classification model trained using the Purchases-Dataset [50] on Google's and Amazon's Machine Learning as a Services (MLaaSs) respectively as well as reaching a 70% attack accuracy when launching an attack on a target model trained using the Texas-Hospital-Dataset [9]. The efficiency of the MIAs did not suffer when shadow models were trained on synthetically generated data where the assumptions about the distribution of the target model's dataset were not completely accurate but also

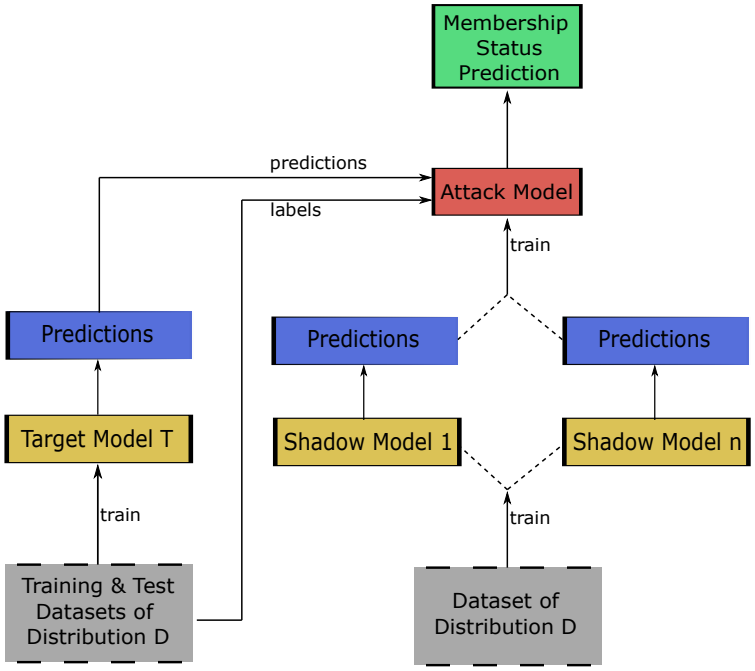


Figure 3.1.: Shadow Training technique as implemented by Shokri et al. [55]. The attack involves training n Shadow Models that attempt to imitate the behavior of the target model T to use for the training of the attack model.

not entirely random.

3.2. Relevant Findings

Role of overfitting. In Shokri et al's experiments, the authors observed that the more a target model overfits to its training data, the more it is susceptible to MIAs. To inspect the relationship between overfitting and the attack's performance, authors measured the train-test accuracy gap for each class and studied it along with the attack precision per class. A bigger gap on one class indicates that the model is overfitted on data elements that belong to that class and plots showed that these bigger gaps are associated with higher attack precision. However, it was stressed that while overfitting can play an important factor, it is not the only factor that can make a model vulnerable to MIAs as the structure and type of the model can also contribute to the problem. As a mitigation measure, the effect that various techniques had on the vulnerability of a model to MIAs was studied. Restricting the

prediction vector to the top k classes or the top one class or using regularization (namely Dropout) with the goal to reduce overfitting both failed at dropping the attack's accuracy to a "safe" rate.

Role of datasets and influential data points. Long et al. [37] were more interested in studying how resilient well-generalized models can actually be. In their experimental setup, they assumed black-box access to a well-generalized target model and selected the most vulnerable target records for which membership inference can be the most successful. It was observed that the more a model is good at predicting data it had not seen during training, the less outlier points it had for which prediction is way off. These individual data records were selected by estimating the number of neighboring samples that each sample has throughout the a reference dataset from the same latter-mentioned distribution. The authors marked the records with fewer neighbors as more vulnerable as they represent outlier points in the distribution and therefore would more likely impose unique influence on the target model. The launched attacks were able to get good performance such as an 88% attack precision on a model trained with UCI-Cancer dataset [57] but this was achieved for models which had enough vulnerable outlier records in the first place. This meant that well-generalized models were only vulnerable to MIAs when it comes to its outlier records as records that were very similar yielded the most incorrect inference results so it was key to identify the records for which MIAs are most effective. Furthermore, L2 regularization was applied to some target models and it was observed that the weight decay reduced the number of outlier vulnerable target records but did not completely eliminate the privacy risk. Therefore, the authors recommend identifying outlier records in the training set and eliminating those that do not contribute much to the utility of the model as a form of protection against MIAs.

In a similar experiment to Truex et al.'s work [62] a set of data records that proved to have a considerable influence on the decision boundary of a model were selected and had their membership inference compared to that of data points that are very similar (in their features). Results have also shown that the MIAs were able to perform much better on the outlier data records.

Broad Data Applicability. In an attempt to relax the assumptions laid down by the initial work that introduced MIAs, Salem et al [51] deployed the same attacks using only one shadow model trained on data that does not belong to the same distribution as the training data of the target model. In a similar context, the attack in this work only had black-box access to their targets. It was able to achieve similar results in comparison to the results in [55] such as a precision of 95% on the same target model trained on CIFAR-100 [31]. The experiments done by the authors allowed to demonstrate that MIAs are broadly applicable.

Long et al. designed an additional indirect inference attack that does not involve querying the target model directly with the record for whom membership should be inferred but by searching and selecting records whose predictions are influenced by the target record but are uncorrelated to it. The latter form of the attack achieved comparable performance as well, showing that MIAs can still be successful even without having to query with the target record directly. One suggested countermeasure was Model-Stacking but only in the case of non-neural network where Dropout is not applicable.

Data-Driven MIAs. Truex et al.'s work was more concerned with the role that datasets of a target model play in its vulnerability to MIAs. In experiments, the Shadow Models technique was used to construct the attack where shadow models are trained using synthetically generated data points created by leveraging the target model prediction API to control the number of features, data types and amount of classes. The attack model is eventually trained using the generated shadow models. The attack results revealed that when it comes to data, the makeup of the training dataset (in-class standard deviation and amount of classes) has an influence on the effectiveness of MIAs.

Role of loss values. Yeom et al. [67] built an attack with assumption that the loss function of the target model is bounded by some constant. The attack simply predicted membership status of a data point with a probability that is proportional to the target model output's loss when querying that data point. This attack achieved a precision of 69.4% and 87.4% on CNNs trained on CIFAR-10 and CIFAR-100 respectively while only being as successful as a random guess when applied on the model trained with MNIST. In comparison to Shokri et al. [55], lower attack precision values have been registered but this attack is way less expensive computationally and does not require to set the same assumptions.

Additionally, Shokri et al. observed a significant difference in the distributions of the outputs produced by the target model between member and non-member inputs and it was marked as one of the factors that the attack exploits. The difference in distributions was computed by the authors using the entropy of the model's prediction vector.

In a later work by Sablayrolles et al. [49], the authors presented a theoretical proof for the optimal MIA depending only on the loss function of the target model and specifically its ability to determine a decisive threshold for the loss values of member and non-member samples. For this purpose, a Membership Attack Threshold (MALT) was implemented along with different variations where the main idea behind it is that it relies on calculating the loss produced by the output prediction vector of a

target model for a data record and then comparing it to a certain threshold where each variation relies on a different technique to compute the threshold value. The success of the latter attack types meant that the white-box setting does not provide any benefits in comparison to a black-box if the produced loss values are enough. When inspecting the loss values further, data records on the validation sets (which the target model did not use for training) showed relatively higher loss values in comparison to the ones from the training dataset but low loss values were common for both member and non-member samples. As a protective measure, Sablayrolles et al. suggest using strong data augmentation when possible to train a model to reduce the accuracy of MIAs.

The scope of the thesis allows only to deal with query access target models, meaning that like the aforementioned experiments in this section, a query-able model would be the target. Additional work has been also been made in different settings of MIAs where members of a collaborative learning environment represent adversaries that attempt to infer membership status of other target models participating in the same federated learning. Nasr et al. [42] demonstrated that in a federated setting and depending on the information that the attacker is assumed to have and with respect to passive attacks that have access to the aggregate target model parameters or even to each individual participant's parameter updates over multiple epochs, the MIAs were able to produce results that were, in some cases, slightly below that of a stand-alone setting but having collaborative learning for a higher number epochs allowed the attacks to perform much better. More work where MIAs are executed under different settings can only help provide more insights into the aspects that can be exploited in similar privacy related attacks.

4. Comparison of MIA Attack Tools

4.1. Researching Attack Tools

In order to carry out the attack and due to the limited time for this thesis, implementing attacks from scratch according to past work was not a viable option so a research is first conducted in order to find a library that offers dynamic MIA attacks that can be launched on models built in TensorFlow (TF) 2 [60] as TF would be used to carry out the experiments. There were two main candidates for the MIA evaluation framework: Adversarial Robustness Toolkit (ART) [17] TF 2 Privacy Tools [61]. Other publicly available MIA libraries are either abandoned, based on a sole attack technique from one underlying research or are not as user-friendly.

The selection experiment involved testing the functionality of ART and TF Privacy Tools using a reoccurring point in recent works where overfitting was highlighted as a factor that increases the threat of MIAs. Therefore, two CNNs were trained on CIFAR-10 using a simple architecture where the first CNN_1 was trained over only 10 epochs allowing it to achieve a 68% training accuracy and a 65% validation accuracy while the second CNN_2 was trained over 60 epochs to be intentionally overfitted to its training data and thus achieving a 84% training accuracy while keeping a 66% validation accuracy. The expectation here is that the MIAs would yield more success on CNN_2 over CNN_1.

All MIAs in ART that assumed black-box access to their target achieved the same performance for both CNN_2 and CNN_1. One of the attacks would return positive membership status and one would return negative membership status no matter the input while another attack under the name of *Label Only Decision Boundary* came with a parameter-calibration function that yielded a parameter with whom the attack would not launch which was very alarming, no examples of using this attack were provided neither.

TF MIAs were in contrast able to achieve a better performance on CNN_2 over CNN_1 as it was expected. Initial tests with this framework also showed that there was quite a handful options that allow to customize launching the attacks. TF Privacy Tools includes MIAs that are created to allow ML model producers build more private models and to help identify architectural and data processing choices that reduce

the information leakage of a model from its training set. Other publicly available MIA libraries are either abandoned, based on a sole attack technique from one underlying research or is not user-friendly. In this thesis, the MIAs from TF Privacy Tools would be used for benchmarking during experimentation.

4.2. TF Privacy Tools MIAs

Considering a target model T, whose vulnerability to MIAs is the interest of an experiment. This section lays out the procedure of setting up these attacks from providing them with the proper input data, specifying certain configurations, running the attacks and interpreting the attack results.

Attack Input Data The attack input is a collection of labels, predictions and loss values that are compute using the target T.

T is trained over a dataset D_s. The predictions $Prob_{train}$ and $Prob_{test}$ of D_{train} and D_{test} features respectively are computed by feeding them as inputs of T and getting the predictions as vectors of probabilities that represent the chances that each input belongs to each possible class. TF MIAs can use either train and test probabilities, or train and test logits to build the attack input data. Logits are non-normalized predictions where they are typically passed to a normalization function such as softmax in order to generate probabilities in the case where a model is solving a multi-classification problem, which is the problem solved in all models in all experiments in this work. Therefore logits have a direct dependence on probabilities. I chose to use probabilities since the target models I am experimenting with produce probabilities as an output vector as the final output layer uses a softmax activation function. I have done few tests to see if there is any major difference in the attack results if one would use logits over probabilities but there was no significant difference.

Now that the predictions are computed for the available data, Categorical Crossentropy [38] is used to calculate the $Loss_{train}$ and $Loss_{test}$ loss generated by each prediction of each training set member instance in X_{train} and non-member instance in X_{test}.

$$((Prob_{train}, Prob_{test}), (Loss_{train}, Loss_{test}), (Y_{train}, Y_{test}))$$

represent the Attack Input Data and this is all that the MIAs need in order to evaluate the resilience of T against the attacks.

Specifying and Generating Attack Input Data Slices While an attack can run on the entire dataset, it can also run on a specific slice of the dataset by targeting specific instances and assessing their vulnerability to MIAs to try to understand better what are the characteristics of the individual data samples that are more threatened by this type of privacy attack. The following options terms of data slice specifications are available:

- Entire Dataset: MIA is ran over the entire dataset.
- Per Class: MIA is ran over data that belong to the same class, for each class.
- Per Loss Percentiles: MIA is ran over data grouped depending on the interval that the computed loss value of each individual data sample falls in relation to the maximum loss value produced by any data sample. There are 10 intervals: $0-10\%, 10-20\%, ..., 90-100\%$.
- Per Classification Correctness: MIA is ran once over data samples that were correctly classified by T and once over the rest of the samples that were misclassified by T.

The Attack Input Data Slices are generated from the Attack Input Data created in paragraph 5.3 according to the provided slicing specifications.

Picking and Running Attack Type There are a few options regarding what attack type to use. There are two categories for attack types: Threshold Attacks and Trained Attacks. The former includes a regular Threshold Attack that relies only on the provided $(Loss_{train}, Loss_{test})$ values in the Attack Input Data and a Threshold Entropy Attack that converts logits in the input into probability values and then calculates their entropy and uses it to set its decision boundary, neither one requires training. For both attacks, a ROC curve is computed to represent the attacker performance at different threshold values. Since no logits are provided within the Attack Input Data and since the Threshold Entropy Attack computes the entropy of probabilities after normalizing logits and therefore behaving as if it is calculating the loss generated by these logits, it is not used in the experiments in this work. In addition, tests done while running the few first experiments produced results where differences between the Threshold Attack and the Threshold Entropy Attack were very significant.

Trained Attacks on the other hand are different ML models that can be trained to produce a prediction that determines how likely an instance is a member of the training dataset of a target T. This includes Logistic Regression, Multi-Layered Perceptron, Random Forest and K-Nearest neighbors. Each attack model is trained

using both ($Prob_{train}, Prob_{test}$) and ($Loss_{train}, Loss_{test}$). In the case of trained attacks, an option to balance the training and test sets used to train the attack models is available. The latter is achieved by sampling a roughly equal number of instances that belong to the specified data slice from the training and testing datasets of used to train T. When running early experiments, having the balancing option turned on or off made insignificant changes to the attack results with a slight advantage to having the balance enabled. The attack models are trained by fine-tuning a set of parameters using a cross-validated grid-search over a parameter grid to minimize the loss of predicting the membership status of the Attack Input Data Slice training set. After training, the attacker predicts the membership status of the Attack Input Data Slice test set and a ROC [6] curve is computed for these predictions in order to try different threshold values for the decision boundary of the membership status prediction produced by the trained attacker. The four trained attack types have been used in all experiments in this work.

Each attack type is now ran through each slice of the Attack Input Data.

Interpreting Attack Results For each category of attack types, the attack results are in the form of a ROC curve from which following values can be extracted:

- Area Under the Curve (AUC) $\in [0, 1]$: Area Under the Curve that measures the degree of separability of the attacker in regards to differentiating between training member and non-member instances where the bigger the AUC is, the better the attacker is at membership inference.

- Attacker Advantage $\in [0, 1]$: Inspired by Definition 4. in [67]. The Attacker Advantage is the result of $\max_{|TPR-FPR|}$ representing the maximum value of the absolute difference of the True Positive Rate (TPR) and False Positive Rate (FPR) of inferring the membership status of data records. This represents the maximum advantage that the attacker can have over all possible thresholds in the classifier.

For all experiments, only the Attacker Advantage metric is used to evaluate the performance of an attacker since according to results on early experiments, both of the AUC and Attacker Advantage are proportional but the Attacker Advantage allows to consider the optimal situation for the attacker in order to truly assess the actual threat to MIAs.

TF MIAs rely on finding differences in the distribution of probabilities (or logits) and loss values of member and non-member data samples (The Threshold Attack only considers loss values while trained attacks take both as input). Trained attacks do not scale the loss input before feeding it to the attacker model to train it which

might problematic since the loss values can be > 1 while probabilities are only within $[0, 1]$.

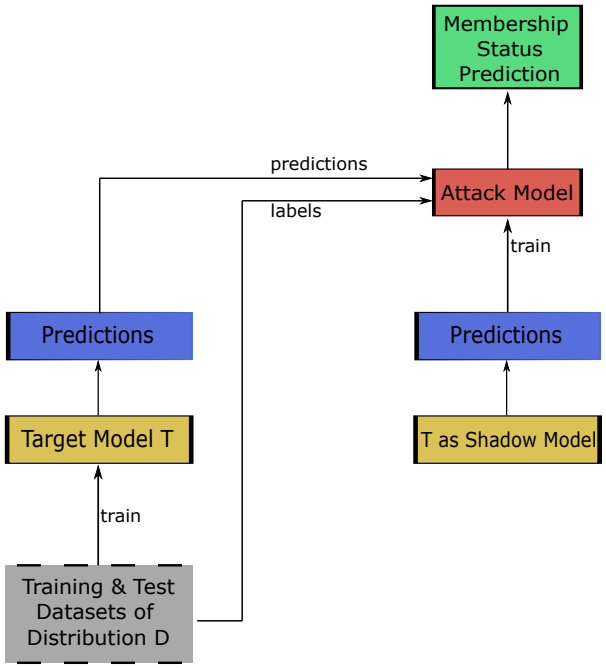


Figure 4.1.: TF MIAs workflow.
The attack involves using the target model T as the only shadow model.

VS. Attacks in Related Work TF Privacy Tools MIAs do not use an explicit shadow model but they use one implicit one and that is the target model itself T as illustrated in Figure 4.1. It also uses the same dataset as the one used to train T since it uses the probabilities produced by T for all samples of the training and validation datasets. One can argue here that according to past work, TF MIAs are deployed in optimal circumstances since the shadow model's goal is to imitate the behavior of T, and the shadow model here is the best possible imitator of T, T itself, trained on the same dataset. One would then expect that these attacks would provide a good estimate of the potential threat that MIAs can have. Now these optimal conditions are only optimal for the situation where there is a classifier target model to attack. In the case of collaborative learning, members of the shared learning have access to additional information during training such as gradient and parameter updates which means other techniques can be implemented to infer membership status of data used to train other instances participating in the same shared learning [42].

5. Experimental Environment

The main goal of the thesis is to study the effects that different architectural choices or training parameters have been on a model's T vulnerability to MIAs. To carry out the experiments, one baseline model has been chosen such that in each experiment, one specific aspect of the training is implemented under different variations while all other training elements and architectural decisions remain the same as in the baseline model. This allows each variation of the baseline model to represent a unique model in the context of this work's experiments. TF MIAs, discussed in chapter 4, are launched on each variation of the baseline model and attack results allow to interpret the role that each studied aspect in the baseline model has in regards to leaking membership information of its training dataset. The experiments are implemented and made available in the official GitLab group for the research group in Privacy Preserving and Secure Machine Learning.[1]

5.1. Baseline Model

The baseline model T is a multi-class classifier Convolution Neural Network (CNN) [10]. T is based on VGG-Net [56] which achieved the best performance in the ImageNet Challenge 2014 competition. VGG-Net models have a modular structure that is easy to understand and implement. In VGG-Net, the model is composed of *VGG Blocks* where each block is a succession of multiple Convolution layers followed by exactly one Max Pooling layer. In the baseline model, the hidden layers are a series of successive VGG blocks composed of one or more Convolution layers, each followed by a non-linear activation and one Max Pooling layer at the end of the block. This architecture relies mainly on increasing the depth (the amount of neurons) of Convolution layers with each VGG block. The final VGG block is followed by a fully Dense block composed of one or more FC layers, each followed by an activation function. The last layer of the Dense block is the output layer that returns the predictions vector and therefore uses Softmax as its non-linear activation function. All kernels in the Convolution layers in VGG-Net have a size of 3×3. Figure 5.1 illustrates the baseline model T. T has 3 VGG blocks composed of 2 successive Convolution

[1] https://git.imp.fu-berlin.de/private_secure_ml/bsc_thesis_mia_mitigation_practical_eval

Name	Value
Training Epochs	100
Batch Size	64
Data Augmentation	None
Dataset	CIFAR-10
Activation Function	ReLU
Weight Initialization	Glorot Uniform
Weight Regularization	None
Batch Normalization	None
Dropout	None
Optimization Algorithm	SGDm with $momentum = 0.01, lr = 0.01$
Loss Function	Categorical Cross Entropy

Table 5.1.: Baseline Model Architecture and Training Parameters.

layers with a kernel size of 3×3 each followed by a Max Pooling layer that uses a kernel size of 2×2. The first VGG block has Convolution layers with a depth of 32, followed by 64 for the second and 128 for the third. The Dense block in the baseline model is composed of a layer that flattens the output of the previous VGG block by unrolling all values into a one-dimensional array. The Flatten layer is followed by a FC layer of 128 neurons. The output layer is a FC layer with 10 neurons for the 10 classes in CIFAR-10 that uses the Softmax activation function and allows the output of each neuron to hold the probability value of the input sample belonging to each class. The rest of architectural choices and training parameters of T are listed in Table 5.1.

T is trained on the CIFAR-10 [31] Dataset D_s which includes 60000 samples of images where 50000 samples are members of D_{train} and 10000 are in D_{test}. Each sample is a RGB image of size 32×32, with a shape of $32 \times 32 \times 3$. The features in this dataset are represented by the three color channels with values in $[0, 255]$ for each pixel in the image. Each sample is pre-segmented, meaning that it contains a single object that represents the class it belongs to and it can belong to only one out of 10 classes. CIFAR-10 is widely used for benchmarking computer vision algorithms in ML [13]. Features X are normalized so that all pixel values are re-scaled to the range $[0, 1]$, this is done by simply dividing the pixel values by the maximum value 255. In all experiments, the training and validation dataset instances are not shuffled meaning that the same samples are used to train all models and the same samples are used to validate its performance.

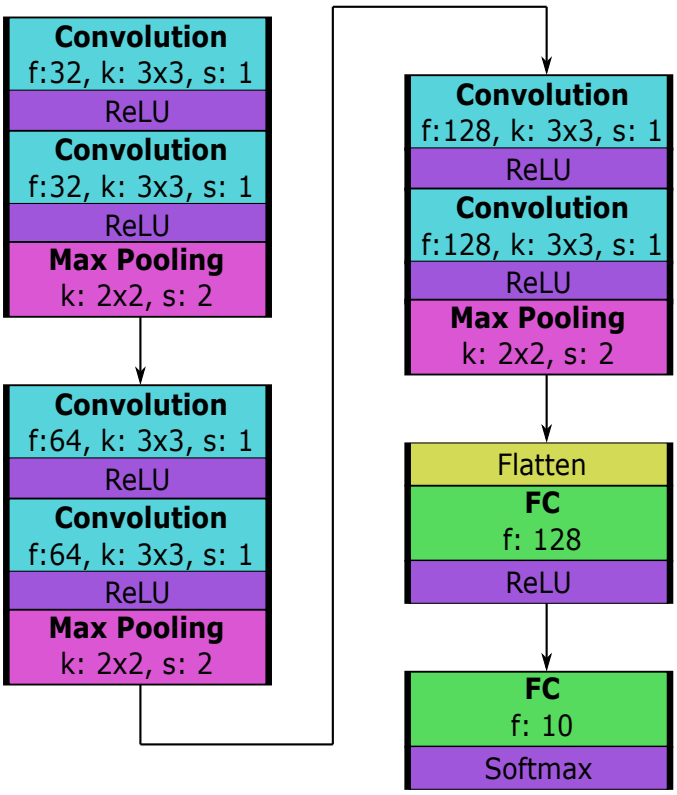


Figure 5.1.: Baseline Model Architecture.
Convolution Layer: f: depth, k: kernel size, s: stride.
Fully Connected (FC) Layer: f: depth.

5.2. Experiments

There are 94 variations of the baseline model T that have been part of the experiments described in the following paragraphs, in addition to eight variations that were using the baseline but manipulating its output and studying the impact on the MIAs performance. The original goal is to find implementations and architectural choices that reduce T's vulnerability to MIAs, while maintaining or preferably improving its performance, especially on unseen data during training.

Training Epochs. The first experiment is used to determine the amount of epochs needed to train the baseline model to reach an acceptable general performance. According to [55], [37], and [67] overfitting is a factor capable of exposing the model to MIAs but it is not a required one. The baseline model is trained over 15 epochs

then TF MIAs are launched on it, then it continues training for another 15 epochs to reach 30 epochs and another round of MIAs are executed and the same process is repeated such that the same model is trained over 50, 100, 150 and 200 epochs where MIAs are launched on each of the aforementioned training "checkpoints".

Batch Size. In this experiment, the baseline model is trained multiple times from scratch, each time with a different batch size $\in \{32, 48, 64, 80, 96, 128, 144, 256, 512\}$.

Data Augmentation. The impact of data augmentation on the performance of MIAs is the point of interest in this experiment. Sablayrolles et al. [49] recommended it as a protective measure against MIAs. In this experiment, 1000 training samples are selected from each of the CIFAR-10 10 classes, resulting into 10000 augmented instances. For each model, only one augmentation technique is applied on the same training samples, these transformations are listed in Table 5.2. The choice of these techniques was inspired by comparative studies [59] [28] where the impact of multiple data augmentation methods on the performance of CNNs is evaluated. The augmented samples are appended to D_{train} such that they occupy the same order in the dataset and batches for every model. Selecting the same data records for augmenting allows to evaluate and compare the effect that each individual transformation has on T's vulnerability to MIAs.

Dataset. The baseline model is trained over 3 additional datasets in this experiment. T is trained with the Street View House Number (SVHN) [54] dataset which is an image dataset of 32×32 colored images where each image is centered around a single digit, which represents the class or label of that image. There are 73257 data records for training and an additional 26032 instances can be used for validation purposes. The MNIST [15] and Fashion-MNIST [65] datasets include grayscaled images of size 32×32 where MNIST includes images of digits while Fashion-MNIST is a dataset of clothing article images and both consist of 600000 training and 10000 testing samples. All records in the latter two datasets are reshaped from 28×28 to 32×32 and since they are grayscaled by default, they are converted to RGB to introduce three color channels for each sample. The goal here is to expose T to different datasets that hold similarities to CIFAR-10 and see how its exposure to the threat of MIAs varies from one dataset to another.

Activation Functions. This experiment involves around training T using different activation functions. Wang et al. [64] studied the influence that activation functions

Name	Description	Range
Random Translation	Translate the input image horizontally and vertically with a rate r.	$r \in [-0.3, 0.3]$
Random Zoom	Zoom in or out the input image a rate r.	$r \in [-0.5, 0.5]$
Random Rotation	Rotate the input image r degrees.	$r \in [-30°, 30°]$
Random Shear	Shear the image horizontally or vertically with a rate r.	$r \in [-0.3, 0.3]$
Inversion	Invert the colors of the input image.	
Random Brightness	Change the brightness of the input image by a rate r.	$r \in [-0.3, 0.3]$
Random Contrast	Contrast the colors of the input image by a rate r.	$r \in [0.6, 1.6]$
Random Saturation	Saturate the colors of the input image by a rate r.	$r \in [0.6, 1.6]$

Table 5.2.: Data Augmentation techniques in experiment.
Geometric techniques include: Translation, Zoom, Rotation, and Shear.
Photometric techniques include: Inversion, Brightness, Contrast, and Saturation

have on the performance of CNNs where they presented common activation functions and used them to solve a facial expression recognition problem, serving as a source to pick the activation functions that suit the baseline CNN model. By default, layers in T use ReLU [21], the most common in CNN models, as activation function by default. Models in this experiment are trained with leaky ReLU [35], sigmoid [2], and tanh [36] where for each model nodes in both Convolution and FC layers use the same activation function.

Weight Initialization. In this experiment, the baseline model's weights are initialized differently each time and the resulting model's resiliency to MIAs is then evaluated where for each model, all weights in all layers are initialized the same way. Initialization techniques are only applied to weights and not to biases, which are always initialized as zeros. Initialization techniques can be divided into two groups: Constant Initialization where weights are initialized with a constant value such as 1 or 0 and Random Initialization where small random values that are for instance from a uniform or a normal distribution are sampled and assigned as the weight's initial values. Li et al. [34] and Dewa et al. [16] both dealt with CNNs and presented an evaluative study to learn the impact that different weight initialization techniques have on the performance of a model and according to their observations, a variety of initializers, listed in Table 5.3, are selected to train the baseline model with.

Weight Regularization. There are three types of weight regularization techniques used in this experiment. Lasso (L1) Regularization [33] uses a term that encourages the sum of the absolute values of the weights to be small, while Ridge (L2) Regularization [63]'s added penalty term encourages the sum of the squares of the parameters to shrink to zero. One of the main differences is that L1 regularization can lead weights to be equal to zero, which can be useful in the case of feature selection [43] as it eliminates the effect of some features completely, and L2 regularization decays the weight's very close to zero but never renders exactly null. The third type of weight regularization in this experiment is a combination of both L1 and L2 Regularization called L1L2 regularization that involves adding both of types of penalizing terms. During this experiment, the baseline model is trained while its Convolution and FC layers have their weights regularized each time by one of the three mentioned techniques and for each technique, three different values of the regularization parameter λ are used: 0.01 (large), 0.005 (medium) and 0.0025 (small).

Batch Normalization. During this experiment, 6 variations of T where 3 have BN layers before every non-linear activation and the other 3 have all of their BN layers after every non-linear activation. For both groups of variations, 3 values of the $momentumm$ parameter are experimented with $m \in \{0.99, 0.5, and 0.1\}$.

Dropout. The Dropout layer is implemented in the baseline model using three different approaches with different $thresholdt$ values for each approach. The first approach has Dropout layers implemented after each convolution layer while the second one has them implemented after each full VGG block resulting in about half the amount of Dropout layers. Both of these two approaches have four variations each where in each variation a different $t \in \{0.05, 0.1, 0.2, 0.3\}$ is used. The third variation has only Dropout added after only FC layers in T's Dense block with three variations $t \in \{0.5, 0.25, 0.1\}$.

Optimization Algorithms. In this experiment, the baseline model is trained using different optimizers where for each optimizer, different learning rates inspired by recent comparative work [18] are used, these are listed in Table 5.4.

Name	Description
Ones	All weights are initialized as 1.
Zeros	All weights are initialized as 0.
Glorot Normal [25]	Samples from a normal distribution with $\mu = 0, \sigma = \sqrt{\frac{2}{units_{in}+units_{out}}}$ where $units_{in}$ is the amount of input units and $units_{out}$ the output units of the neuron where the weight resides.
Glorot Uniform [25]	Samples from a uniform distribution within $[-y, y]$ with $y = \sqrt{\frac{6}{a+b}}$ where $units_{in}$ is the amount of input units and $units_{out}$ the output units of the neuron where the weight resides.
He Normal [26]	Samples from a normal distribution with $\mu = 0, \sigma = \sqrt{\frac{2}{units_{in}}}$ where $units_{in}$ is the amount of input units of the neuron where the weight resides.
He Uniform [26]	Samples from a uniform distribution within $[-y, y]$ with $y = \sqrt{\frac{6}{units_{in}}}$ where $units_{in}$ is the amount of input units of the neuron where the weight resides.
Lecun Normal [32]	Samples from a normal distribution with $\mu = 0, \sigma = \sqrt{\frac{1}{units_{in}}}$ where $units_{in}$ is the amount of input units of the neuron where the weight resides.
Lecun Uniform [32]	Samples from a uniform distribution within $[-y, y]$ with $y = \sqrt{\frac{1}{units_{in}}}$ where $units_{in}$ is the amount of input units of the neuron where the weight resides.
Random Normal Distribution	Samples from a normal distribution with $\mu = 0, \sigma \in \{0.05, 0.025, 0.01\}$.
Random Uniform Distribution	Samples from a uniform distribution within $[-y, y]$ with $y \in \{0.15, 0.05, 0.075\}$.

Table 5.3.: Weight Initialization options in experiment.
μ represents the mean and σ the standard deviation of a normal distribution.

Name	Description
Stochastic Gradient Descent (SGD)	$lr \in \{0.02, 0.05, 0.1\}$.
Stochastic Gradient Descent with Nesterov (SGD_n)	$lr \in \{0.02, 0.05, 0.1\}$. Applies Nesterov momentum [58].
Stochastic Gradient Descent with plain Momentum (SGD_m)	$lr \in \{0.001, 0.005, 0.01\}$. Applies a plain momentum, with a value of 0.9, that accelerates the gradient descent in the relevant direction dampening oscillations.
Stochastic Gradient Descent with plain Momentum and Nesterov (SGD_{mn})	$lr \in \{0.001, 0.005, 0.01\}$. Applies the same concepts from both SGD_n and SGD_m.
$RMSProp$	$lr \in \{0.0001, 0.0002, 0.0005\}$. Maintains a moving average of the square of gradients and divides the gradient by the root of this average. It uses plain Momentum.
$Adam$	$lr \in \{0.0001, 0.0002, 0.0005\}$. Based on an adaptive estimation [30].
$Adamax$	$lr \in \{0.0001, 0.0002, 0.0005\}$. Variant of $Adam$ [30].
$Nadam$	$lr \in \{0.0001, 0.0002, 0.0005\}$. Variant of $Adam$ that implements Nesterov momentum [19].

Table 5.4.: Weight Initialization options in experiment.
lr: Learning Rate.

Loss Functions. T is a multi-classification model therefore the choice of loss functions that can be used to train such a model is limited. By default, T uses Categorical Crossentropy as its loss function. Model variations in this experiment use KL-Divergence and Poisson loss functions.

Combinations. This experiment is ran after the aforementioned experiments have been ran and results are analyzed. It includes two variations of the baseline model where one implements all architectural choices and techniques that achieved the best general performance according to results of previous experiments and the other implements the choices that led to the best protection against MIAs while also keeping a general performance that is at least as good as the baseline model's. The two variations in this experiment are trained for the amount of training epochs that achieved best performance or protection against MIAs and then trained further since they need more epochs to converge as they implement various regularization

techniques handled in other experiments.

Top K Probabilities. Limiting the output of a prediction vector has been suggested by [55] as a mitigation strategy. This experiment considers the baseline T and runs MIAs on it while providing the attacks only with a limited amount K of probabilities in the prediction vector each time with $K \in \{10, 5, 3, 2, 1\}$. $K = 10$ represents the default case where all probabilities are supplied to the adversary since there are only 10 classes and $K = 1$ means that only the top probability which is also the final prediction of the model is given to the attacks. The MIAs receive the top K probabilities along with the indexes of the classes that each probability value belongs to. The goal is to evaluate the efficiency of this mitigation measure.

Manually Generated Probabilities. The final experiment in this work does not deal with actual trained models as the probability values that are given to the MIAs to build and execute their attacks are manually generated. There are 4 variations in this experiment where for each variation the prediction vectors all predict the correct class (they have the highest probability on the true class) while the probabilities assigned to other classes is divided equally between them. Generated probabilities for instances in D_{train} are set as 0.91 for the true class and 0.01 for the rest of the classes while instances in D_{test} are set as $0.91 - l$ for the true class and $(0.09 + l)/9$ for the rest of the classes where $l \in \{0, 0.05, 0.1, 0.2\}$. The goal of this experiment is to supply the MIAs with probabilities that produce specific distributions when one computes the loss values of the predictions between member and non-member instances. For $l = 0$, the probabilities and their loss values have the same distribution while for $l > 0$, the distributions start to differ in their mean but still keep the same equal standard deviations.

5.3. Metrics

When analyzing experiments, the focus point is looking at the effects of the architectural choices on specific metrics in regards to the ML model and study the impact of changes in these metrics on the vulnerability of the models to MIAs.

Training Accuracy (Train Acc.) The amount of correct classifications divided by the total amount of classifications for instances in D_{train}

Validation Accuracy (Test Acc.) The amount of correct classifications divided by the total amount of classifications for instances in D_{test}

Train-Validation Accuracy Difference (Train-Test Diff.) The difference between the training and validation accuracy values. The training accuracy is usually higher than the validation one, rendering this difference always positive, at least in the case of models trained in these experiments.

Train-to-Validation Accuracy Ratio (Train-Test Ratio) The training accuracy divided by the validation accuracy. Along with the train-to-validation accuracy difference, this metric aims to measure how well a model performs on data it has seen during training in comparison to other data.

In-Class Standard Deviation (STD) Truex et al. [62] highlighted the role that the distribution of features in the training set has on the efficiency of MIAs. To investigate this when analyzing the results of the experiments, the standard deviation of features class-wise are computed their correlations with the attack results for each class is evaluated.

Train-Test Loss Distribution Difference (Loss Dist. Diff.) The difference between the distributions of loss values between training and testing data samples is captured via the Kruskal-Wallis-Test, a non-parametric method for testing whether samples originate from the same distribution or not. The test can be used to compare two or more independent samples of equal or different sample sizes. This test's output consists of a P-Value in $[0, 1]$ and a Statistic. When the P-Value is small enough (usually < 0.05) then one can assume that the differences in the distributions is not just due to random sampling and one can conclude that the input distributions come from different populations. In the latter case, one can use the Statistic value to evaluate the differences where the higher that value is, the larger the discrepancies between the two distributions are.

Spearmans Correlation. Interpreting the performance of MIAs is done by inspecting the Attack Advantage value (defined in paragraph 4.2) that each attack achieves. In order to measure the role that the previously mentioned metrics have in terms of maximizing or minimizing the threat of MIAs, the correlation between those metrics and the Attack Advantage values is measured using Spearman's Rank Correlation coefficient r with $r \in [-1, 1]$. It represents a non-parametric measure of rank correlation that computes the statistical dependence between two variables V_1 and V_2

without assuming a linear relationship. The sign of r indicates the direction of correlation between the two variables and the absolute value $|r|$ represents the strength of the correlation. For $r > 0$, V_1 tends to increase when V_2 increases while for $r < 0$, V_1 tends to decrease when V_2 increases and for $r = 0$ then one can conclude that there is no tendency for V_1 to increase or decrease when V_2 moves. As for the strength of the correlation, one can define Strong (S) for $|r| >= 0.7$, Moderate (M) for $|r| \in [0.5, 0.7)$, Weak (W) for $|r| \in [0.3, 0.5)$ and None (N) for $|r| < 0.3$.

The mentioned metrics would not be tracked for the variations of the models in experiment as a whole as the same metrics applied in the context of instances belonging to the same class or according to the classification correctness would have their correlation to the attack performance evaluated. As highlighted in , TF MIAs allow to launch attacks on specific splits of the dataset and studying the performance of MIAs in different slices of the available data is a major point of interest in this thesis.

6. Results

6.1. Notable Architectural Choices and Training Parameters

In all experiments, if a variation of the baseline model T produced a model that is unable to reach a validation accuracy of around 65% then it is discarded from the experiment as the goal is to study the performance of MIAs in a setting where the target model has a good enough general performance. Table 6.1 displays a summary of the experiments regarding training implementations for T and presents the results with an emphasis on two aspects:

- Best Performance: The variation that allowed T to achieve the highest validation accuracy among the variations in the same experiment. The best-performance variant is compared to the general performance of the baseline model.

- Best Protection: The variation that allowed T to achieve the lowest attack advantage value among the five available MIA types discussed in paragraph 4.2 among the variations in the same experiment. The general performance of the variant with the highest protection is then also compared to that of the baseline.

Experiment	Best Performance (vs. Baseline)	Best Protection (vs. Baseline)
Training Epochs	100 (0.0%)	150 (-1.0%)
Batch Size	64 (0.0%)	144 (+1.5%)
Data Augmentation	Saturation (+0.1%), and Translation (0.0%)	Translation (0.0%)
Dataset	MNIST (+27.7%)	MNIST (+27.7%)
Activation Function	tanh (+3.0%)	ReLU (0.0%)
Weight Initialization	He Normal (+0.6%)	He Normal (+0.6%)
Weight Regularization	None (0.0%)	L2 with large $\lambda = 0.01$ (-1.2%)
Batch Normalization	Before activations with large $momentum = 0.99$ (+9.9%)	After activations with small $momentum = 0.99$ (+5.7%)
Dropout	Per VGG-Block with large $threshold = 0.3$ (+10.1%)	Per Convolution layer with large $threshold = 0.3$ (+5.9%)
Optimization Algorithm	RMSProp with $lr = 0.0005$ (+6.6%)	SGDmn with $momentum = 0.99, lr = 0.01$ (-0.7%)
Loss Function	Categorical Cross Entropy (0.0%)	Categorical Cross Entropy (0.0%)

Table 6.1.: Best Performing and Best Protective Architectural Choices and Training Parameters.
vs. Baseline: For each architectural choice or training parameter that represents the best performance or best protection against MIAs, the difference in general performance (measured using the validation accuracy) between the baseline model and the variation that implements the given architectural choice is highlighted between parenthesis. For example, the variation trained over 150 epochs provides the best protection among all other variations in the training epochs experiment and it has a general performance that is 1% lower than that of the baseline and hence -1.0%. This is used to highlight the changes in utility of the variation models.

6.2. Individual Experiment Results

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
Epochs: 15	91.6%	70.8%	1.294	0.208	1299.503
Epochs: 30	95.6%	71.4%	1.339	0.242	1789.965
Epochs: 50	95.6%	70.4%	1.358	0.252	2176.95
Epochs: 100	93.4%	71.7%	1.303	0.217	2220.754
Epochs: 150	90.6%	70.7%	1.281	0.199	1553.021
Epochs: 200	84.7%	64.2%	1.319	0.205	1082.692

Table 6.2.: Training Epochs experiment variation models performance metrics.

Figure 6.1.: MIAs performance on Training Epochs experiment variation models.

Training Epochs. Increasing the training epochs of the baseline model from 15 to 100 increases its exposure to MIAs while keeping its general performance relatively unchanged in the ranges of 71.7% for 100 epochs and 70.4% for 50. This increase in MIAs threat is also met by an increase in the loss distribution difference. Training for 150 and 200 epochs degraded the model's performance both on training member and non-member samples and also led to decreasing the difference in the generated loss of those member and non-member records. Overall, training for 200 epochs yields the least exposure to MIAs at the cost of losing performance as it can only reach a 64.2% for validation accuracy while training for 150 epochs kept the validation accuracy barely above 70% and allowed for the next best protection against MIAs.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
Batch Size: 32	69.1%	61.4%	1.125	0.077	253.319
Batch Size: 48	84.0%	65.2%	1.288	0.188	840.164
Batch Size: 64	93.4%	71.7%	1.303	0.217	2220.754
Batch Size: 80	97.3%	72.5%	1.342	0.248	3693.785
Batch Size: 96	98.4%	71.9%	1.369	0.265	4534.32
Batch Size: 128	99.3%	72.4%	1.372	0.269	5293.78
Batch Size: 144	99.4%	73.2%	1.358	0.262	6060.222
Batch Size: 256	100.0%	71.8%	1.393	0.282	9022.628
Batch Size: 512	100.0%	69.2%	1.445	0.308	8033.676

Table 6.3.: Batch Size experiment variation models performance metrics.

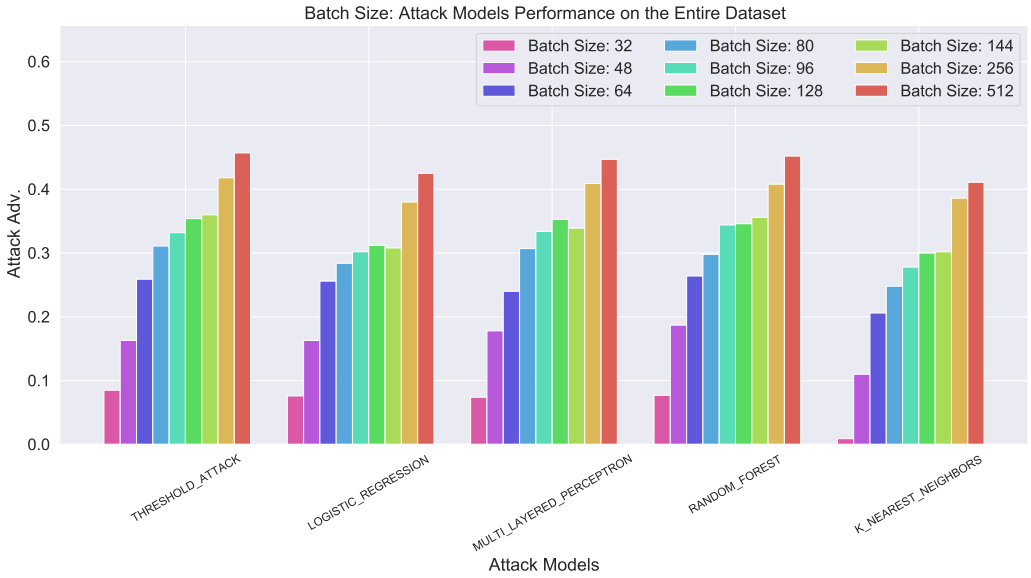


Figure 6.2.: MIAs performance on Batch Size experiment variation models.

Batch Size. In this experiment, results showed that the higher the batch size value, the higher the train-test accuracy ratio, train-test difference and loss distribution difference. MIAs reached more success when launched on variations with the highest batch sizes values as all attack types were able to reach an attack advantage of more than 0.35 for batch size > 128. Batch size 64 had the best resilience to the attacks in comparison to other variations with a validation accuracy $> 70\%$ since a lower batch size meant less exposure to MIAs but a validation accuracy of $< 65\%$.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
None	93.4%	71.7%	1.303	0.217	2220.754
Translation	91.6%	71.7%	1.278	0.199	1524.19
Zoom	91.6%	70.2%	1.305	0.214	1628.607
Rotation	91.9%	70.9%	1.296	0.21	1625.931
Shear	93.1%	69.8%	1.334	0.233	1711.826
Invert	91.4%	71.0%	1.287	0.204	1746.081
Brightness	93.0%	69.6%	1.336	0.234	1895.144
Saturation	92.9%	71.8%	1.294	0.211	1994.477
Contrast	92.1%	69.2%	1.331	0.229	1867.187

Table 6.4.: Data Augmentation experiment variation models performance metrics.

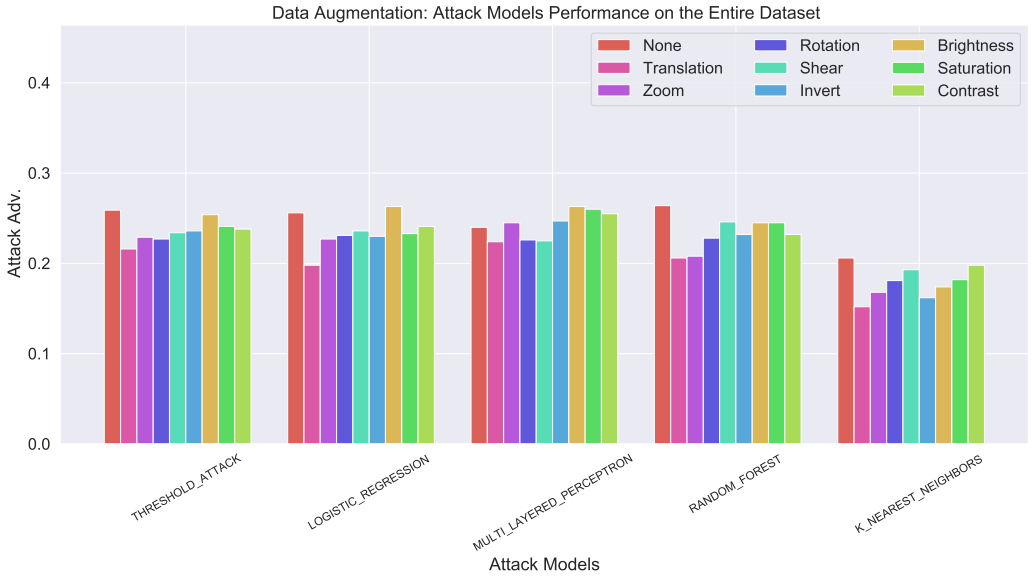


Figure 6.3.: MIAs performance on Data Augmentation experiment variation models.

Data Augmentation. The impact of the augmentation techniques on the performance of the baseline model resulted into lowering its validation accuracy into a value in the range $[69.2\%, 71.7\%]$ and its training accuracy to the range $[91.4\%, 93.4\%]$. All variations in this experiment allowed to decrease the baseline's exposure to MIAs, with varying degrees of success, at the cost of a drop in its performance. Applying random translation or random zoom has allowed the model to consistently achieve the lowest attack advantage values among all attack types. Photometric data transformations provided the least protection against MIAs but they were still more resilient than the baseline.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
CIFAR-10	93.4%	71.7%	1.303	0.217	2220.754
Fashion	99.5%	91.5%	1.087	0.08	0.434
MNIST	100.0%	99.4%	1.006	0.006	0.51
SVHN	98.7%	91.8%	1.075	0.069	0.484

Table 6.5.: Datasets experiment variation models performance metrics.

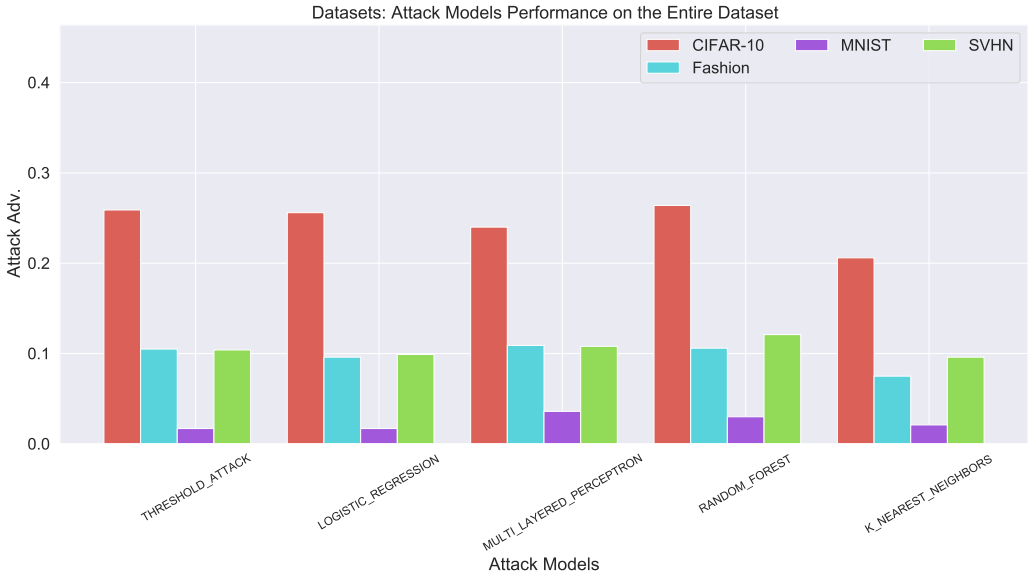


Figure 6.4.: MIAs performance on Datasets experiment variation models.

Datasets. MNIST was the "easiest" dataset that T was able to learn as it reached a 100% training accuracy and a 99.4% validation accuracy with no single MIA being able to reach an attack advantage of over 0.05 when launched on that variation. Both SVHN and Fashion-MNIST allowed T to get to a 91% accuracy with MIAs being more successful in their attacks on these two datasets than they were against MNIST. CIFAR-10, the default baseline dataset, was the "hardest" to learn for T and the most susceptible to MIAs by a significant difference to the rest of the variations in the dataset experiment.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
ReLU	93.4%	71.7%	1.303	0.217	2220.754
Leky ReLU	93.7%	71.5%	1.31	0.222	2667.876
tanh	100.0%	74.7%	1.339	0.253	4869.428

Table 6.6.: Activation Functions experiment variation models performance metrics.

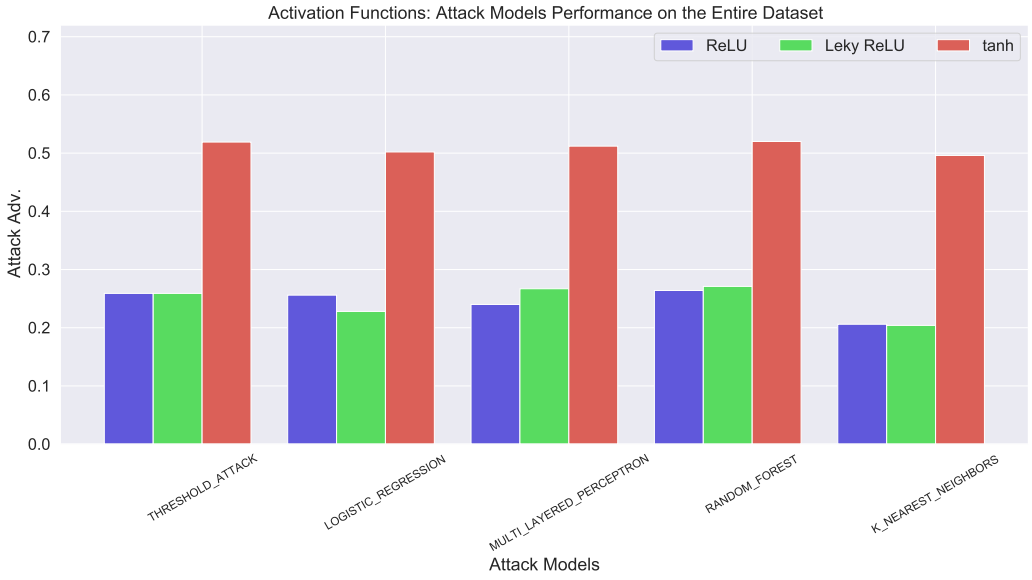


Figure 6.5.: MIAs performance on Activation Functions experiment variation models.

Activation Functions. When trained with activation functions other than ReLU, leaky ReLU or tanh, T's performance suffers significantly and is unable to meet the minimum general performance requirement. Using ReLU or leaky ReLU got very similar results such as 93.4% and 93.7% training accuracy, and 71.7% and 71.5% validation accuracy respectively with very close results in regards to the attack advantage achieved by every attack type while training with tanh allowed the training accuracy to reach 100% increasing the test accuracy to 74.7% and at the same time, exposing the model to more than twice the MIAs threat in comparison to training with ReLU or leaky ReLU.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
Glorot Normal	93.6%	69.9%	1.339	0.237	2429.058
Glorot Uniform	93.4%	71.7%	1.303	0.217	2220.754
He Normal	92.0%	72.3%	1.272	0.197	1538.813
He Uniform	92.2%	70.4%	1.31	0.218	1669.119
Lecun Uormal	94.1%	71.0%	1.325	0.231	1883.075
Lecun Uniform	94.1%	71.6%	1.314	0.225	2312.826
Normal, $\sigma = 0.05$	91.7%	68.8%	1.333	0.229	2316.041
Normal, $\sigma = 0.025$	94.0%	67.3%	1.397	0.267	2954.445
Uniform, $y = 0.15$	88.6%	67.1%	1.32	0.215	1604.321
Uniform, $y = 0.075$	91.7%	68.3%	1.343	0.234	2172.483
Uniform, $y = 0.05$	93.9%	68.6%	1.369	0.253	2599.92

Table 6.7.: Weight Initialization experiment variation models performance metrics.
Uniform Dist. in range $[-y, y]$, Normal distribution with standard deviation σ and mean $\mu = 0$

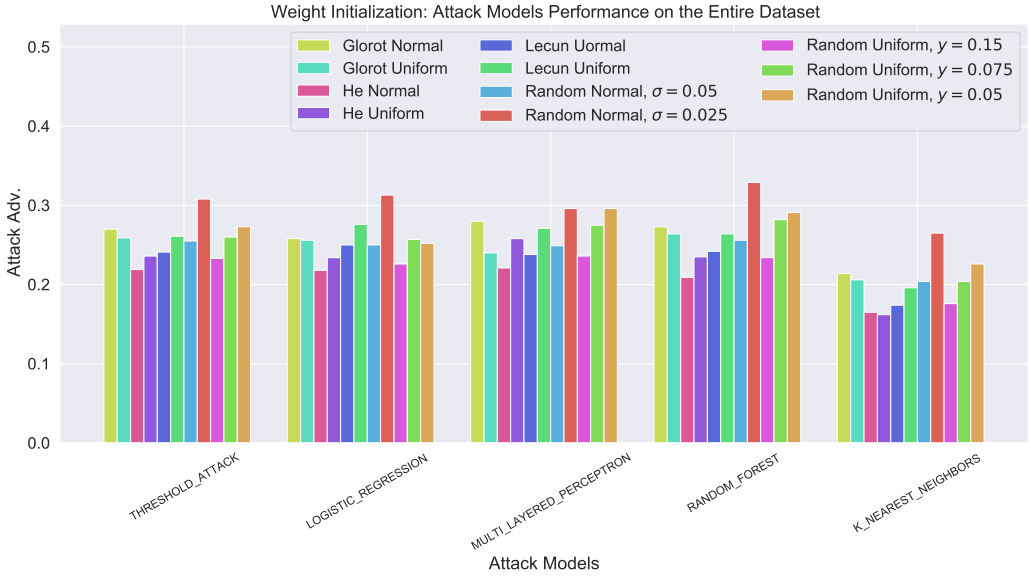


Figure 6.6.: MIAs performance on Weight Initialization experiment variation models.
Uniform Dist. in range $[-y, y]$, Normal distribution with standard deviation σ and mean $\mu = 0$

Weight Initialization. Training with constant initialization for the weights to values of zero or one or using the random initialization by sampling from a normal distribution with $\mu = 0, \sigma = 0.01$ (which produced initial weight values very close to zero) yielded models that were stuck during training and therefore discarded. Initializing the weights using He Normal allowed for an increase of the validation

accuracy to reach 72.3% while contributing to decrease the training accuracy to 92% in comparison to the baseline model, that initializes its weights with Glorot Normal, training accuracy of 93.4%. Using He Normal also allows T to get the lowest attack advantage values registered on all attack types with He Uniform and Random Uniform (with $y = 0.15$) being the next two most resilient variations to MIAs.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
None	93.4%	71.7%	1.303	0.217	2220.754
L1, $\lambda = 0.01$	93.7%	71.6%	1.309	0.221	2334.106
L1, $\lambda = 0.005$	93.5%	69.2%	1.351	0.243	2042.989
L1, $\lambda = 0.0025$	93.9%	70.3%	1.336	0.236	2003.992
L2, $\lambda = 0.01$	93.7%	70.5%	1.329	0.232	1813.368
L2, $\lambda = 0.005$	93.7%	70.9%	1.322	0.228	2177.053
L2, $\lambda = 0.0025$	93.0%	71.2%	1.306	0.218	2266.556
L1L2, $\lambda = 0.01$	94.4%	68.9%	1.37	0.255	1763.014
L1L2, $\lambda = 0.005$	92.8%	70.7%	1.313	0.221	1882.08
L1L2, $\lambda = 0.0025$	94.2%	69.8%	1.35	0.244	2344.172

Table 6.8.: Weight Regularization experiment variation models performance metrics.

Figure 6.7.: MIAs performance on Weight Regularization experiment variation models

Weight Regularization. The impact of the weight regularization techniques on the baseline's training accuracy was very negligible but all implemented techniques contributed to decreasing T's validation accuracy to as much as a 2.8% and 2.5% decrease when using L1L2 with $\lambda = 0.01$ and L1 with $\lambda = 0.005$ respectively. Using L2 with $\lambda = 0.01$ achieved both best general performance and least exposure to MIAs threat among all deployed attack types while using L1L2 with the same $\lambda = 0.01$ offered comparable protection against MIAs but resulted into a higher decrease in the validation accuracy as it wasn't to break through 68.9%.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
None	93.4%	71.7%	1.303	0.217	2220.754
Before, $m = 0.99$	100.0%	81.6%	1.225	0.184	2533.32
Before, $m = 0.5$	100.0%	80.1%	1.248	0.199	2404.657
Before, $m = 0.1$	100.0%	78.0%	1.282	0.22	2087.143
After, $m = 0.99$	100.0%	80.9%	1.236	0.191	2523.526
After, $m = 0.5$	100.0%	79.4%	1.259	0.206	2487.458
After, $m = 0.1$	100.0%	77.4%	1.292	0.226	2273.411

Table 6.9.: Batch Normalization experiment variation models performance metrics.
After: variation where batch normalization layers implemented after applying the activation function, Before: variation where batch normalization layers implemented before applying the activation function, m: batch normalization *momentum*.

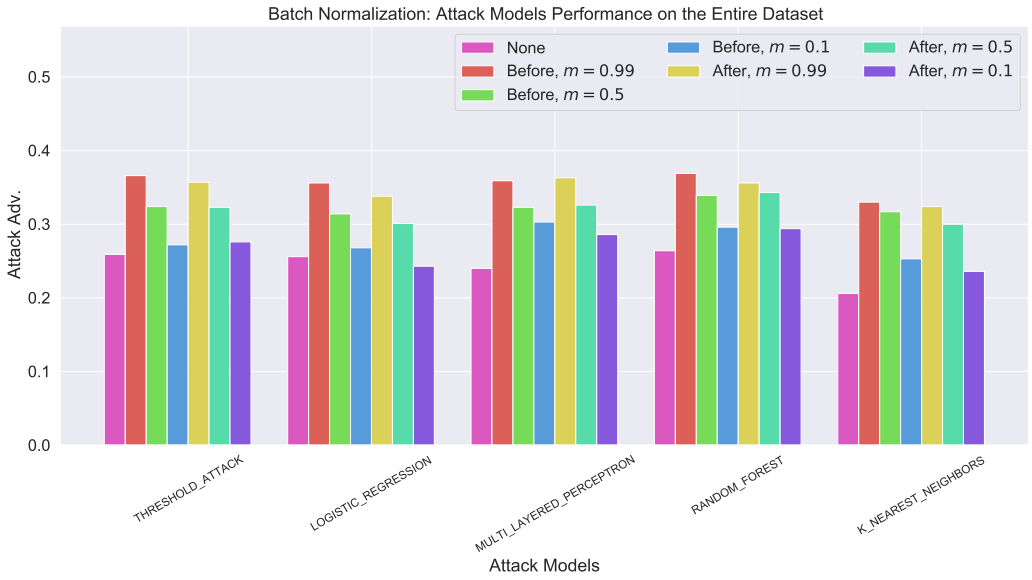


Figure 6.8.: MIAs performance on Batch Normalization experiment variation models.
After: After Activations, Before: Before Activations, m: batch normalization *momentum*.

Batch Normalization. Implementing batch normalization layers with any of the three *momentum* values allowed T to reach 100% training accuracy while using a higher *momentum* value allows the model to reach a higher validation accuracy for batch normalization layers implemented both before or after the activations. As for attack performances, the higher the *momentum* value, the more exposure the model has to MIAs with the baseline model, that does not make use of any batch normalization layer, achieving the lowest attack advantage consistently on all attack types.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
None	93.4%	71.7%	1.303	0.217	2220.754
Layer-wise, $t = 0.3$	84.3%	71.7%	1.176	0.126	773.643
Layer-wise, $t = 0.2$	91.2%	72.1%	1.265	0.191	1625.568
Layer-wise, $t = 0.1$	93.4%	73.4%	1.272	0.2	2151.144
Layer-wise, $t = 0.05$	79.5%	77.6%	1.024	0.019	266.712
VGG-Only, $t = 0.3$	86.8%	78.1%	1.111	0.087	767.038
VGG-Only, $t = 0.2$	92.0%	77.1%	1.193	0.149	1543.35
VGG-Only, $t = 0.1$	93.6%	74.5%	1.256	0.191	2089.237
VGG-Only, $t = 0.05$	86.3%	81.8%	1.055	0.045	515.599
Dense-Only, $t = 0.5$	90.5%	81.2%	1.115	0.093	1035.585
Dense-Only, $t = 0.25$	94.3%	78.2%	1.206	0.161	1715.058
Dense-Only, $t = 0.1$	94.7%	76.1%	1.244	0.186	1984.212

Table 6.10.: Dropout experiment variation models performance metrics.
t: dropout *threshold*.

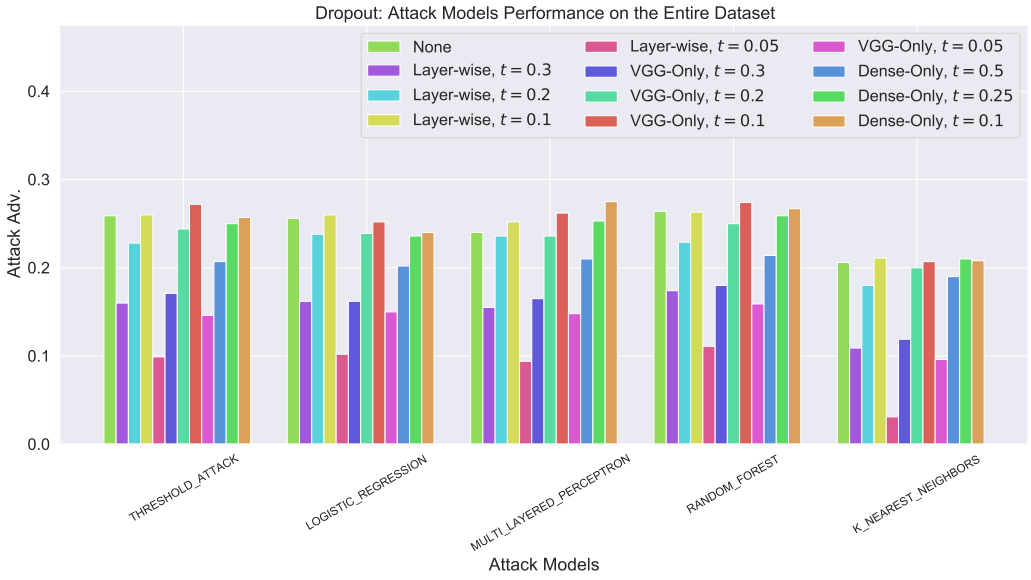


Figure 6.9.: MIAs performance on Dropout experiment variation models.
t: dropout *threshold*.

Dropout. All variations were able to meet or surpass the baseline's 71.7% validation accuracy while the training accuracy was comparable to that of the baseline for low *threshold* values but significantly lower for high *threshold* values. Across all attack types, large *threshold* values allowed for the highest protection against MIAs followed by variations that used medium dropout *threshold* values while the lowest values yielded models that were as exposed to MIAs as the baseline model.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
SGD, $lr = 0.1$	93.4%	68.4%	1.365	0.25	2220.754
SGDn, $lr = 0.1$	93.2%	68.2%	1.367	0.25	2965.983
SGDm, $lr = 0.01$	93.4%	71.7%	1.303	0.217	2220.754
SGDmn, $lr = 0.01$	93.9%	71.0%	1.323	0.229	1693.642
RMSprop, $lr = 0.0005$	98.9%	78.3%	1.263	0.206	6603.181
Adam, $lr = 0.0005$	99.3%	77.2%	1.286	0.221	3753.805
Nadam, $lr = 0.0005$	99.1%	77.3%	1.282	0.218	3377.019
Adamax, $lr = 0.0001$	94.3%	69.0%	1.367	0.253	826.035

Table 6.11.: Optimization Algorithms experiment variation models performance metrics.
lr: learning rate.

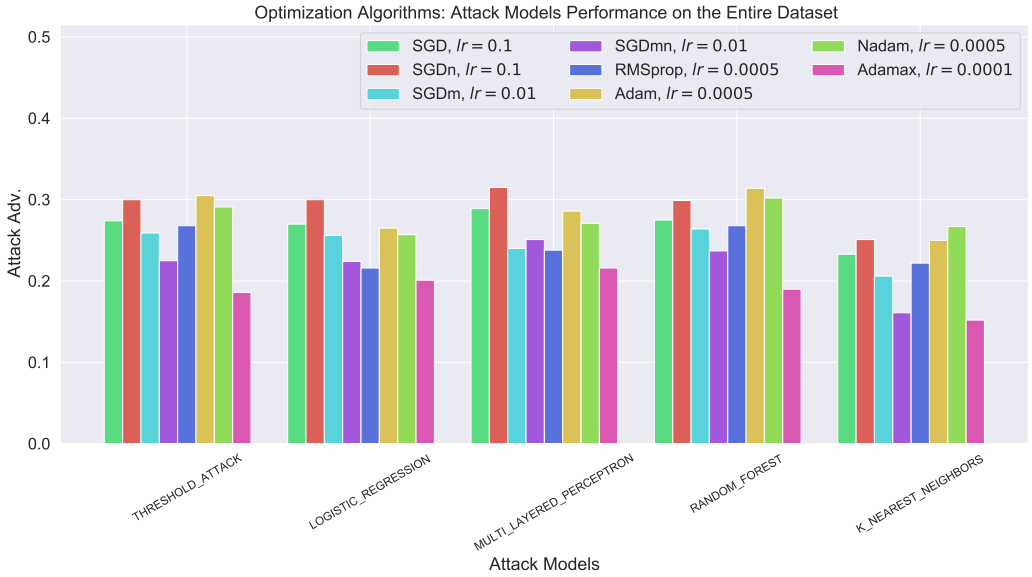


Figure 6.10.: MIAs performance on Optimization Algorithms experiment variation models.
lr: learning rate.

Optimization Algorithms. For each type of optimizers, different learning rates are used and the variations with the learning rates that yield the highest validation accuracy values are then compared to one another. Variations of the baseline model trained with RMSProp $lr = 0.0005$, SGDmn $lr = 0.01$ and Adamax $lr = 0.0001$ were able to offer better protection against the MIAs in comparison to the baseline that uses SGDm with $lr = 0.01$ with the variation that used RMSProp being the only one able to achieve a higher validation accuracy of 78.3% in comparison to the baseline.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
CCE	93.4%	71.7%	1.303	0.217	2220.754
KL Divergence	92.8%	70.2%	1.322	0.226	1898.248
Poisson	100.0%	67.9%	1.473	0.321	6893.189

Table 6.12.: Loss Functions experiment variation models performance metrics.
CCE: Categorical CrossEntropy

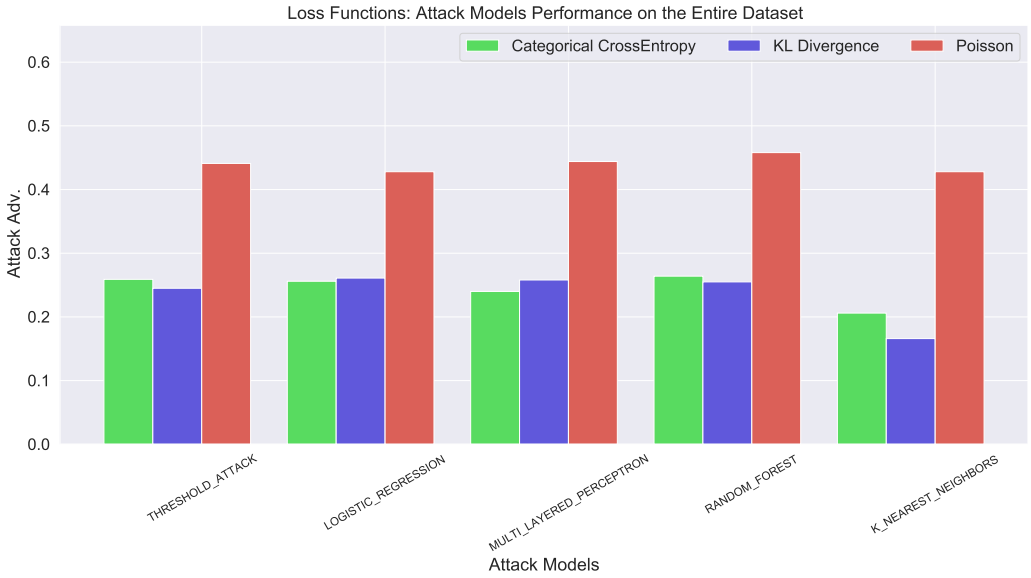


Figure 6.11.: MIAs performance on Loss Functions experiment variation models

Loss Functions. Baseline model trained with Categorical Crossentropy and the variation that used KL-Divergence provided similar performance in terms of both training and validation with comparable exposure to MIAs while the variation trained with the Poisson loss was able to achieve a 100% training accuracy but failed to get its general performance past 67.9% while also being the most vulnerable to MIAs.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
Best Performance, $e = 100$	89.1%	82.2%	1.084	0.069	783.993
Best Performance, $e = 200$	92.8%	82.6%	1.123	0.102	1108.003
Best Protection, $e = 150$	94.4%	77.4%	1.22	0.17	344.316
Best Protection, $e = 200$	95.1%	79.2%	1.201	0.159	385.351

Table 6.13.: Combinations experiment variation models performance metrics. e represents the training epochs amount.

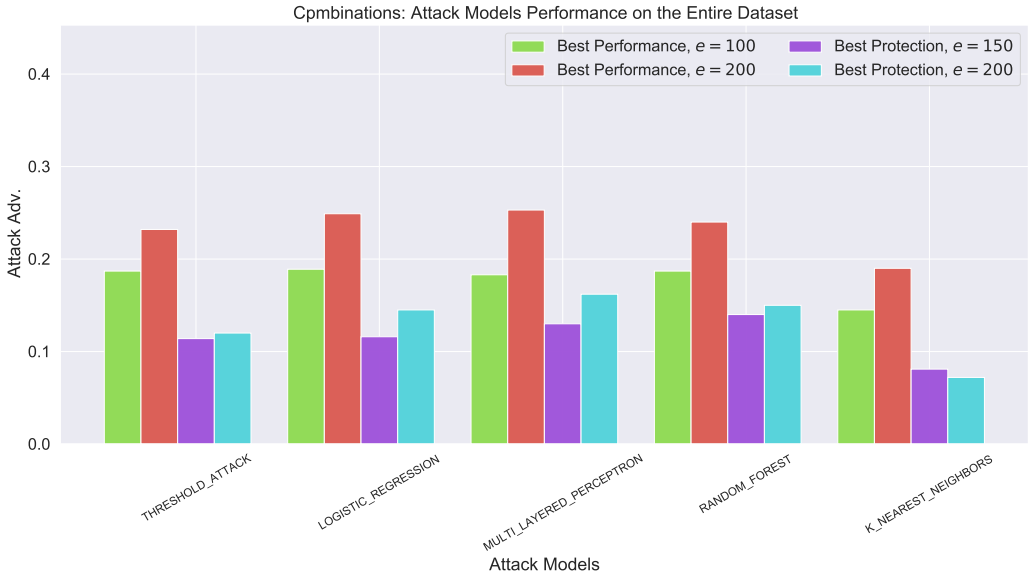


Figure 6.12.: MIAs performance on Combinations experiment variation models e represents the training epochs amount.

Combinations. The results of the past experiment and specifically the choices that yielded the highest performance and protection, listed in Table 6.1, were used to build two variations of the baseline model where one has the best performing traits and the other implements the most protective traits. Combination variations were all able to achieve better security against MIAs than the bare baseline model with the best protective measures achieving the least exposure to the attacks.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
$k = 10$	93.4%	71.7%	1.303	0.217	2220.754
$k = 5$	94.6%	71.7%	1.319	0.229	2220.755
$k = 3$	94.6%	71.7%	1.319	0.229	2220.754
$k = 2$	94.6%	71.7%	1.319	0.229	2220.754
$k = 1$	94.6%	71.7%	1.319	0.229	5204.624

Table 6.14.: Top K Probabilities experiment variation models performance metrics.

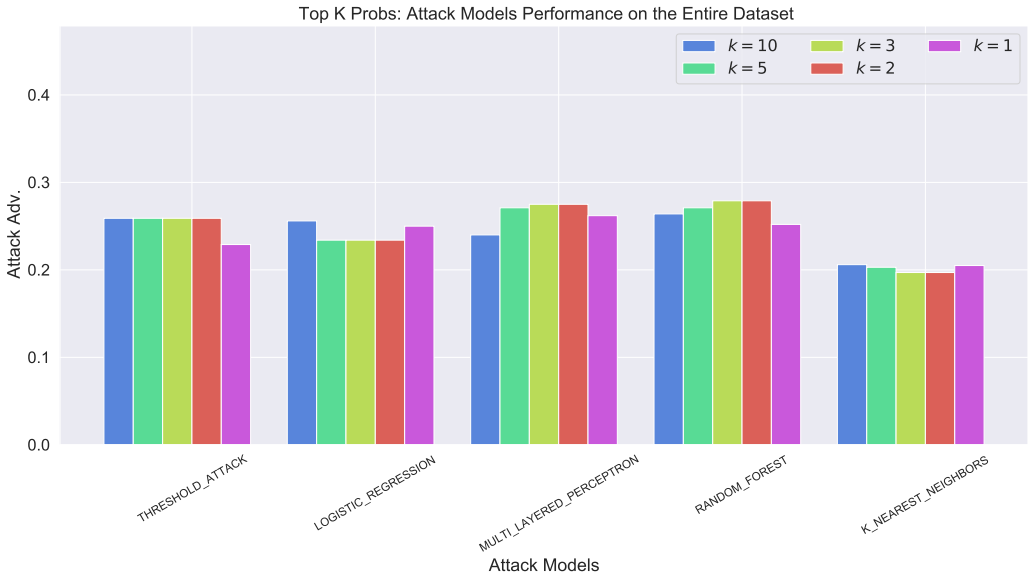


Figure 6.13.: MIAs performance on Top K Probabilities experiment variation models

Top K Probabilities. When the MIAs are not able to receive the probabilities to all possible classes but rather only the highest probabilities and their correspondent classes then results have shown that MIAs are still capable to extract about the same membership status information from its target model with the attacks being more successful when they are only allowed only the top five, three or two probabilities in the output prediction vector.

Target Model	Train Acc.	Test Acc.	Acc. Ratio	Acc. Diff	Loss Dist. Diff.
$l = 0$	100.0%	100.0%	1.0	0.0	0
$l = 0.05$	100.0%	100.0%	1.0	0.0	59999.0
$l = 0.1$	100.0%	100.0%	1.0	0.0	59999.0
$l = 0.2$	100.0%	100.0%	1.0	0.0	59680.703

Table 6.15.: Manually Generated Probabilities experiment variation models performance metrics.

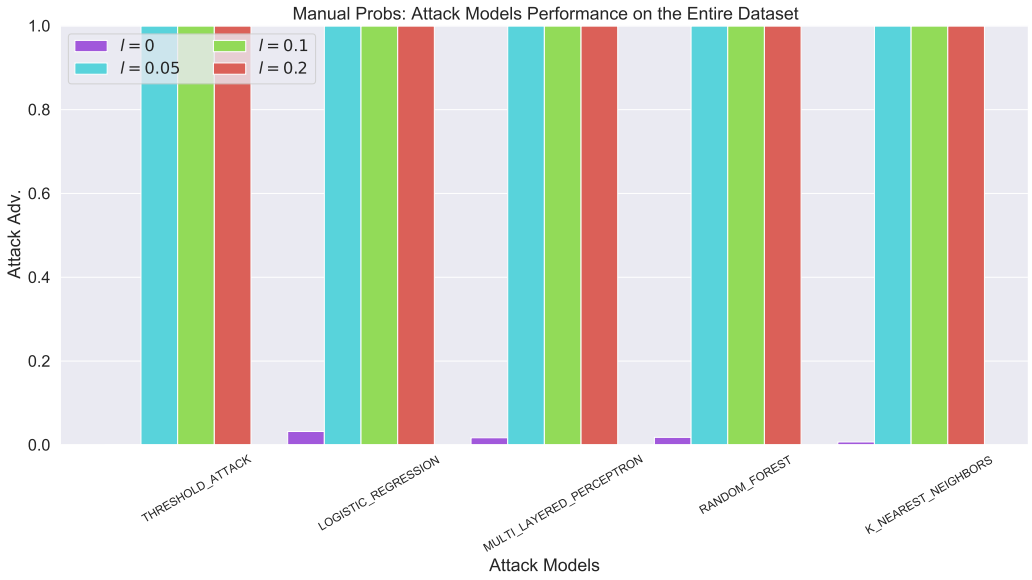


Figure 6.14.: MIAs performance on Top K Probabilities experiment variation models

Manually Generated Probabilities. The final experiment in this thesis involves generating manual probabilities for the MIAs as described in details in paragraph 5.2. For the variation where there was no difference between the member and non-member instance probability vectors, no attack type was able to extract meaningful membership information as they all achieved near-zero attack advantage. For the rest of the variations, all attack models achieved the maximum 1.0 attack advantage when deployed on all member and non-member instances.

After highlighting the aspects that can make the baseline model perform better on unseen data or get the lowest exposure to MIAs, it is of the interest of this work to look at the variations in these experiments and study the relationships that the tracked metrics might have with the vulnerability of the models to MIAs on different data slices.

6.3. Consistent Tendencies

Entire Dataset. When inspecting the performance of MIAs on the entire dataset, which would be CIFAR-10 in all experiments except the one discussed in paragraph 5.2 where different datasets have been used, one consistent correlation that has been observed is between the attack advantage and the difference in distributions of loss values between member and non-member instances. This correlation was positive strong and in most cases > 0.9. A correlation between the difference of training and validation accuracy and the attack advantage was also observed but it was moderate strong and not observable in variations that handled weight regularization, batch normalization and optimizers. Figure 6.15 and Appendix A illustrate these correlations among variation models from all experiments. This implies that the bigger the gap between the training and validation accuracy of a model and the more the model's prediction of member and non-member instances generate loss values that vary consistently between the two groups of samples, the higher is the model's exposure to MIAs.

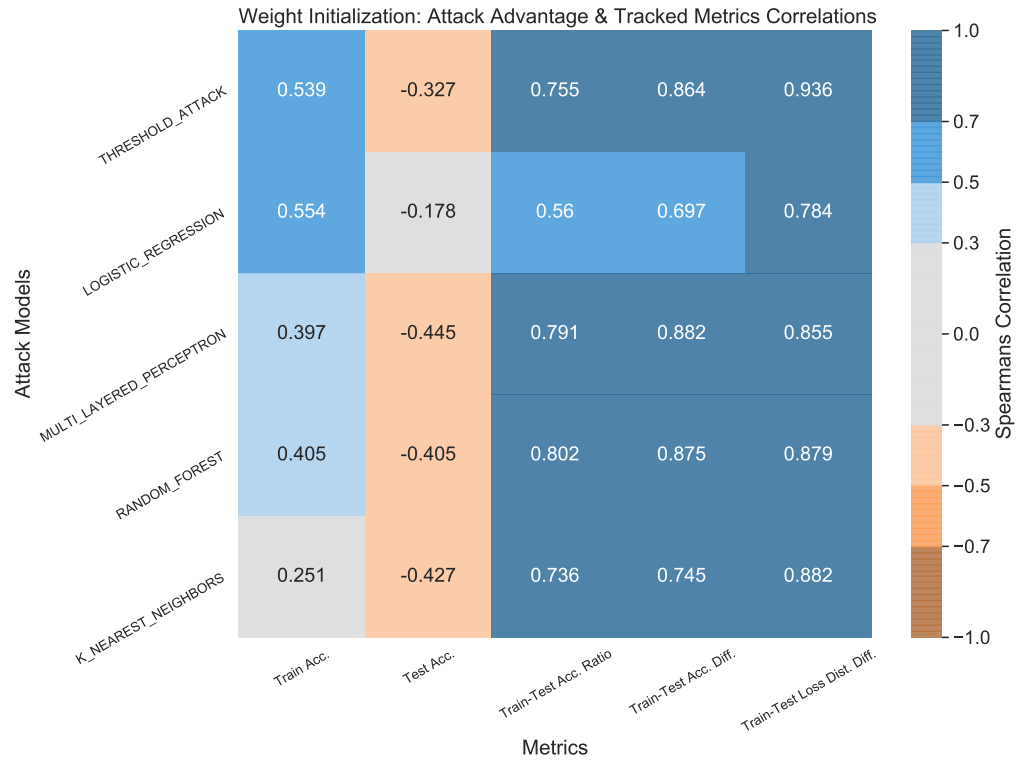


Figure 6.15.: MIAs Attack Advantage and Variations Tracked Metrics Correlations Heat-map, Data Slice: Entire Dataset.
Experiment: Weight Initialization.

Per Class. The performance of MIAs when launched with data that belong to the same class showed a few strong correlations among all variations of T. The first observed tendency was that of the validation accuracy per class which was negative strong correlated to the attack advantage registered per class. Figure 6.16 and Appendix B show that a data point from a class y is more susceptible to MIAs if the model can not generalize well to solve for unseen data of class y. At the same time, a strong positive correlation between the in-class attack advantage and both the in-class train-to-test accuracy ratio and the train-test accuracy difference is observed as illustrated in Figure 6.17 and Appendix C. The latter observation means that the wider the gap between the performance of the model on samples of the a class that it has seen during training and samples that it had not encountered in learning, the more the samples of that class are exposed to MIAs. Another strong positive correlation is formulated between the attack advantage and in-class loss difference between member and non-member samples of the same class. Figure 6.18 and Appendix D illustrate that the better MIAs are at differentiating between the produced loss between training and non-training samples of the same class, the more successful they are at inferring membership status.

Per Classification-Correctness. MIAs are also deployed on two additional data slices consisting of first, all correctly classified samples by the model, and secondly the rest of instances that were misclassified. Among all variations of T in the experiments, Figure 6.19, and Figure 6.20 and Appendix E show that MIAs were able to perform much better when they are inferring membership status of incorrectly classified data in comparison to correctly classified samples with a ratio of > 1 for all variations with a higher ratios in most cases. In all variations, it has been also observed that the difference in produced loss distributions between member and non-member misclassified elements is higher than the distribution difference between member and non-member but incorrectly classified data with ratios comparable to that of the attack advantage.

Per Loss-Percentile. When launching the attacks at data sliced and grouped according to the loss that each individual sample produces via its prediction, the MIAs perform similarly on data samples that have their loss in percentile intervals from $0-10\%$ to $80-90\%$ where a clear trend of increasing or decreasing is lacking. However, the performance of the attacks consistently witnesses a sharp increase in terms of the attack advantage when they are executed on data samples that produce the highest loss, meaning in the $90-100\%$ percentile interval. This latter observation is illustrated in Figure 6.21 and Appendix F via heatmaps of the MIAs performance on data sliced according to the loss value it generates.

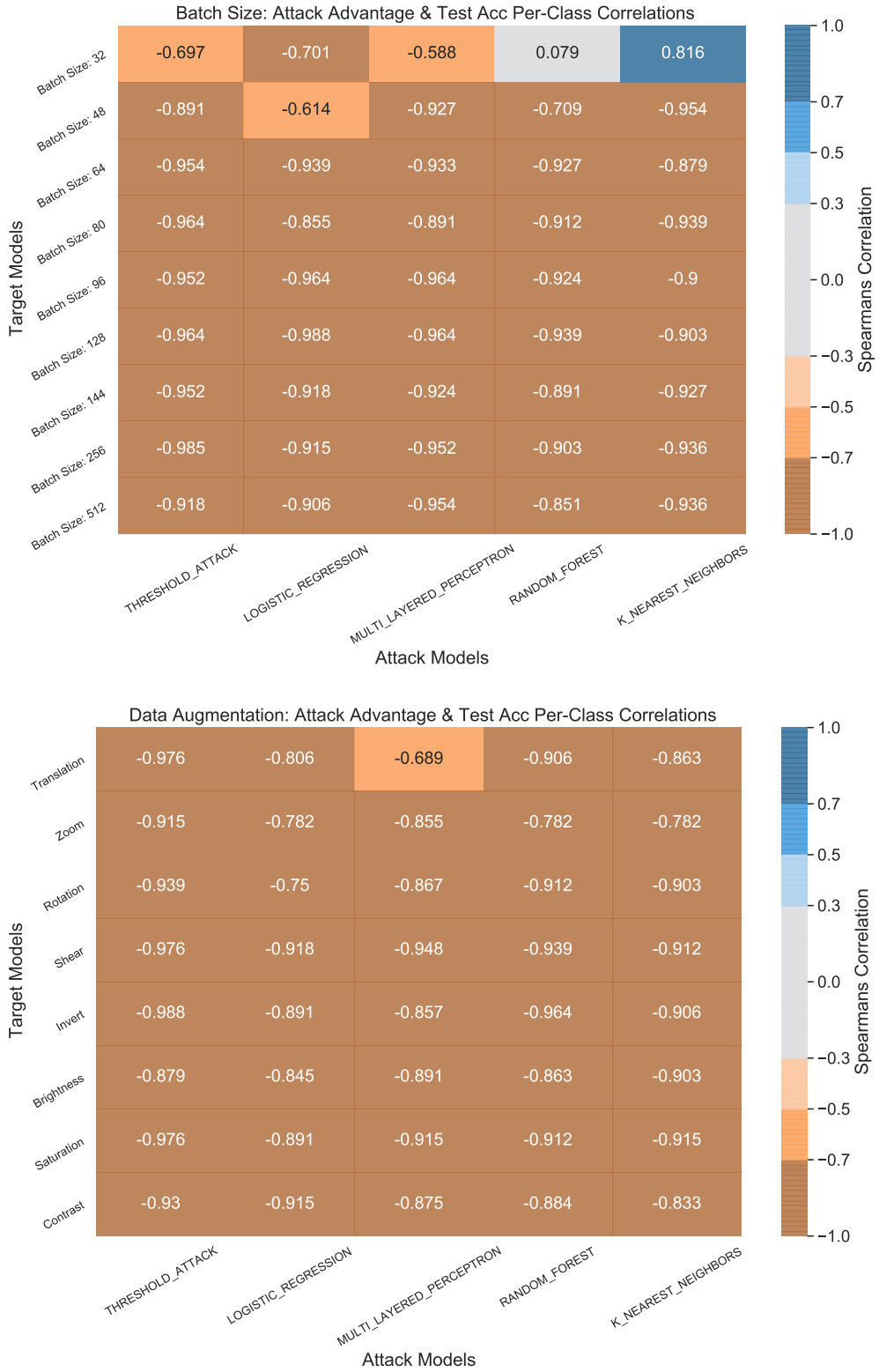


Figure 6.16.: MIAs Attack Advantage and Validation Accuracy Correlations Heatmap, Data Slice: Per Class.
Experiment: Batch Size, and Data Augmentation.

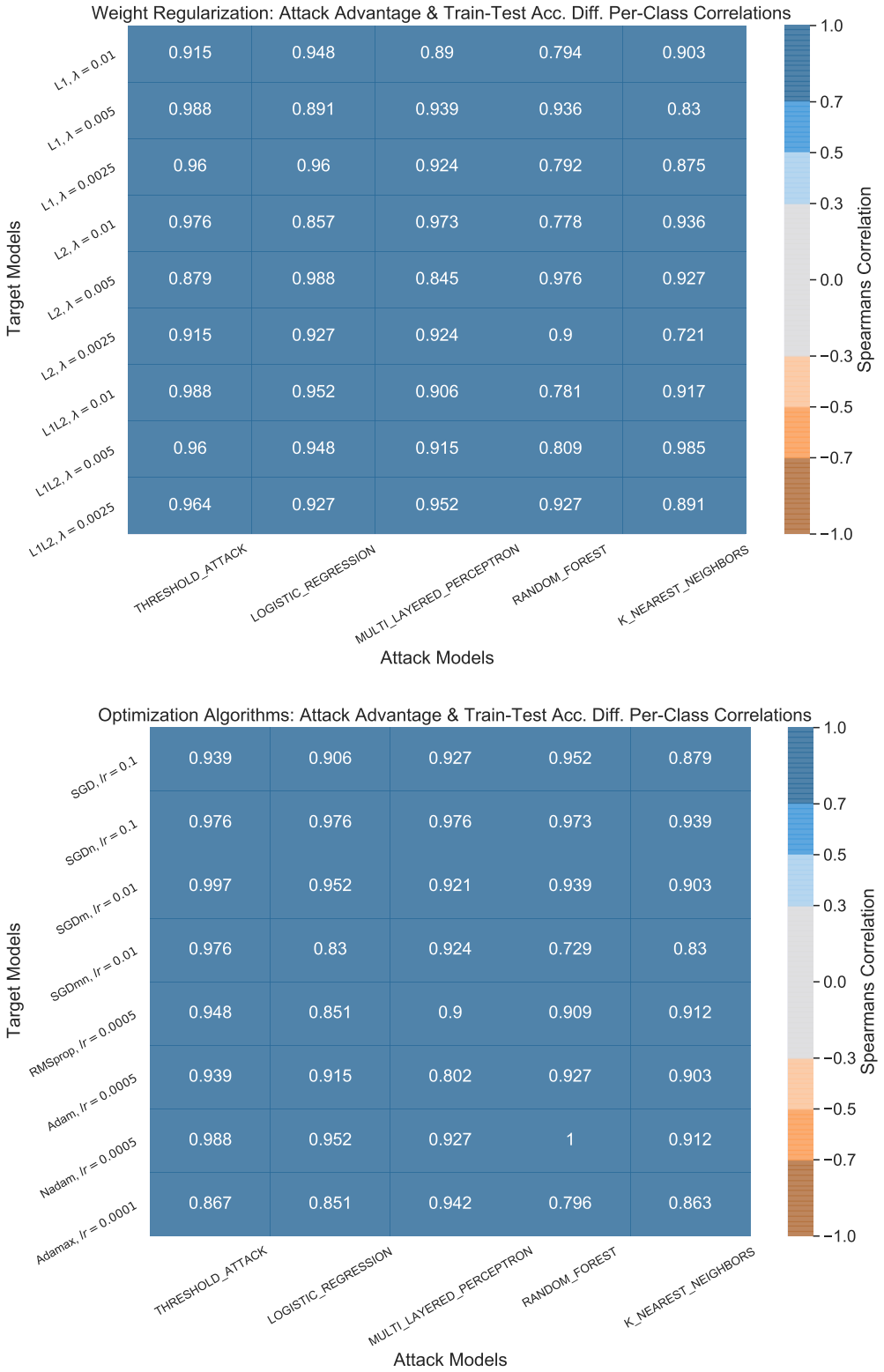


Figure 6.17.: MIAs Attack Advantage and Train-Test Diff. Correlations Heat-map, Data Slice: Per Class.
Experiments: Weight Regularization, and Optimization Algorithms.

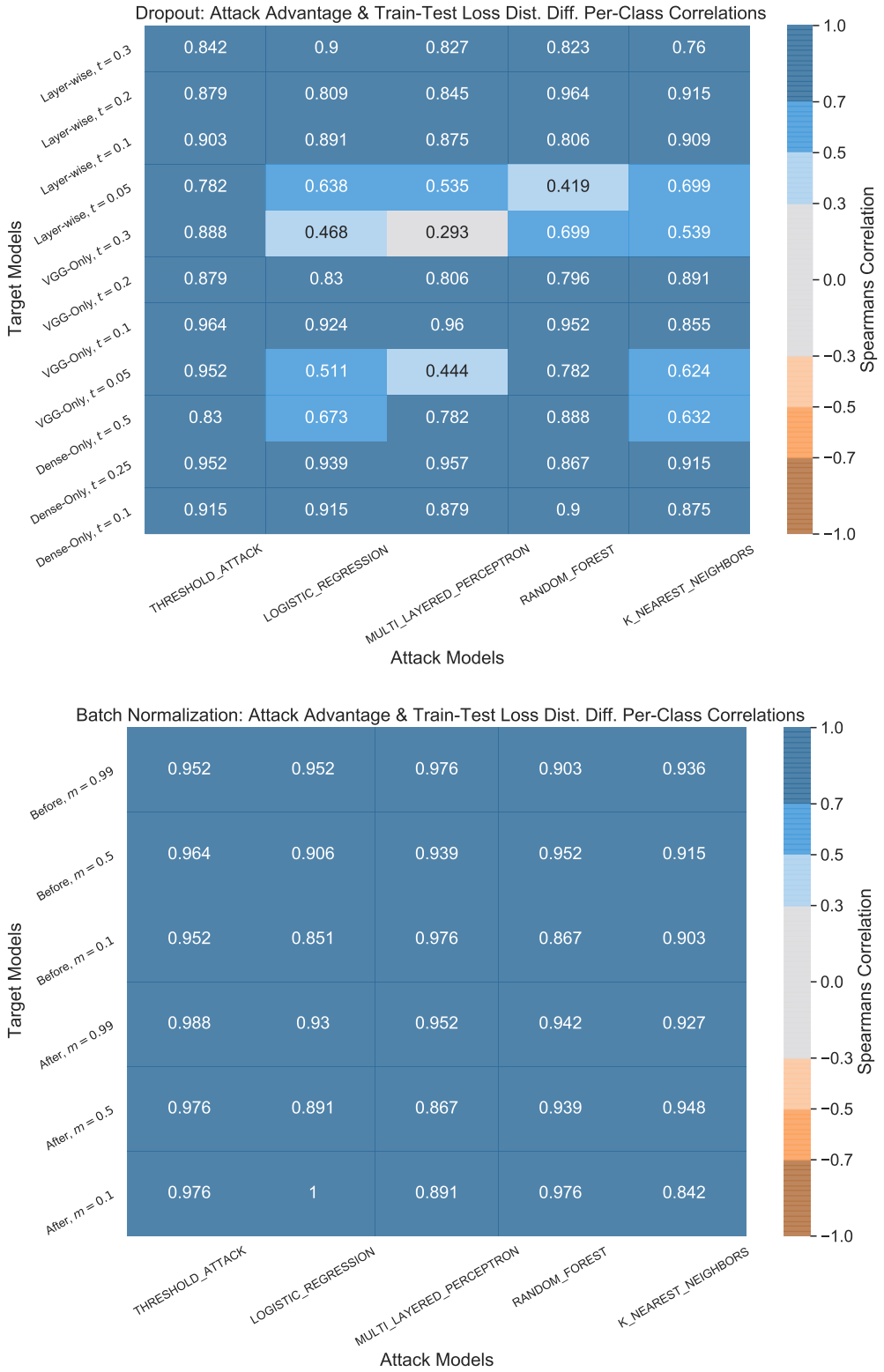


Figure 6.18.: MIAs Attack Advantage and Train-Test Diff. Correlations Heat-map, Data Slice: Per Class.
Experiments: Dropout, and Batch Normalization.

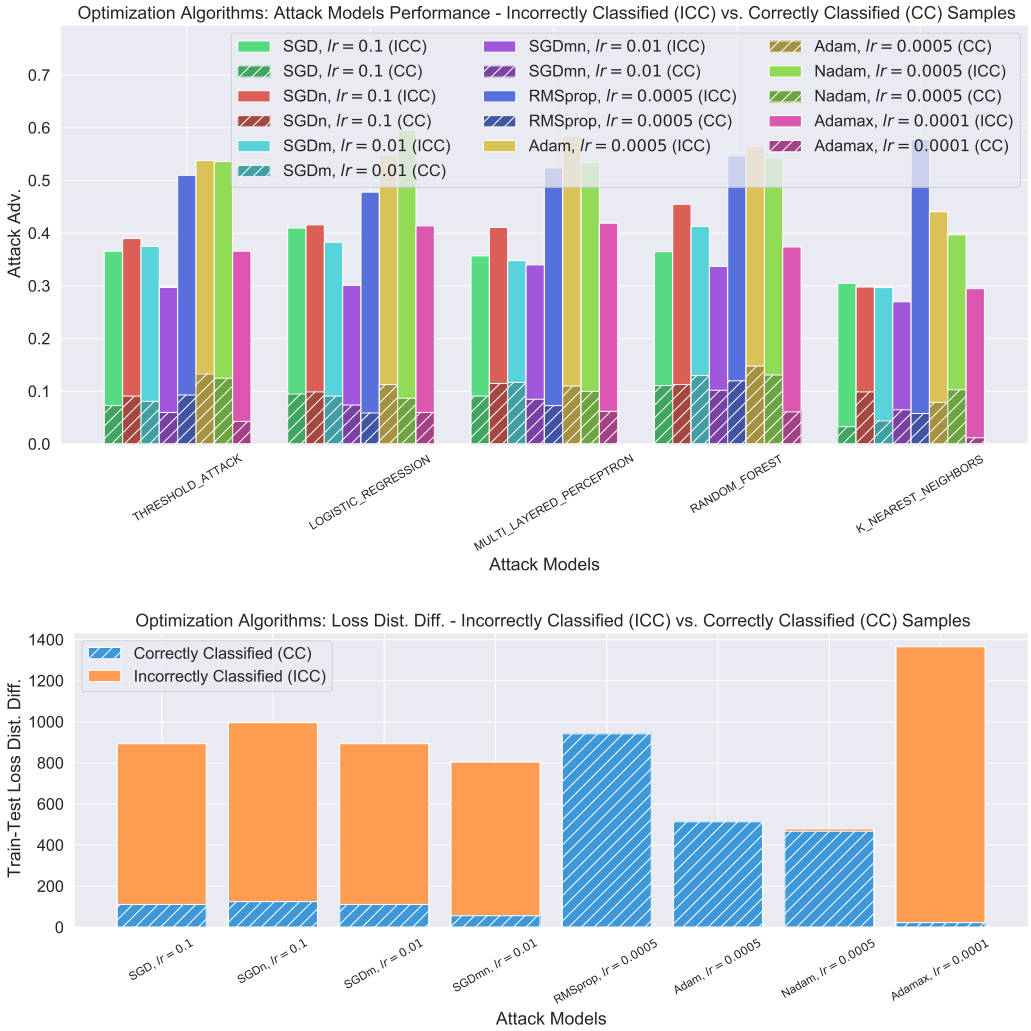


Figure 6.19.: Upper Figure: MIAs performance variation models, Data Slice: Per Classification Correctness.
Lower Figure: Loss Distribution Difference: Incorrectly Classified vs. Correctly Classified Samples.
Experiment: Optimization Algorithms.

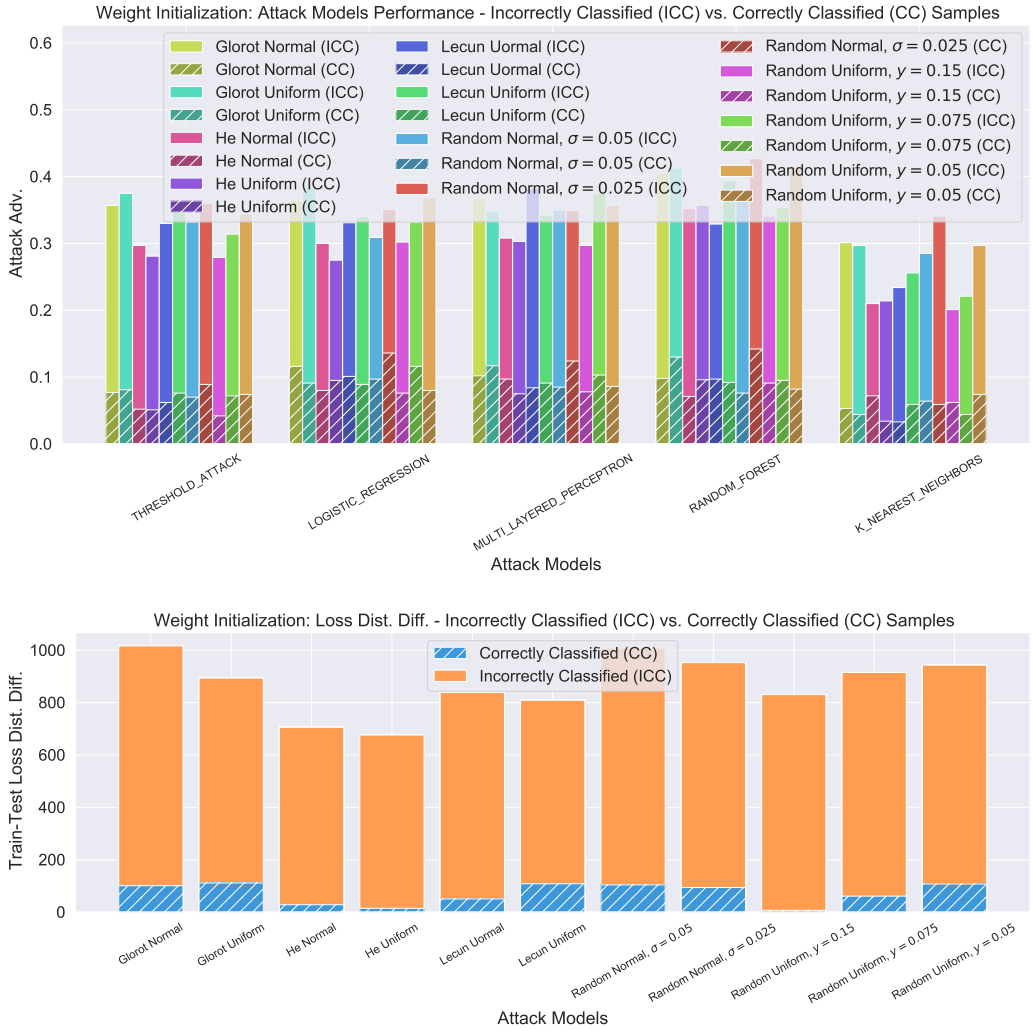


Figure 6.20.: Upper Figure: MIAs performance variation models, Data Slice: Per Classification Correctness.
Lower Figure: Loss Distribution Difference: Incorrectly Classified vs. Correctly Classified Samples.
Experiment: Weight Initialization.
Uniform Dist. in range $[-y, y]$, Normal distribution with standard deviation σ and mean $\mu = 0$.

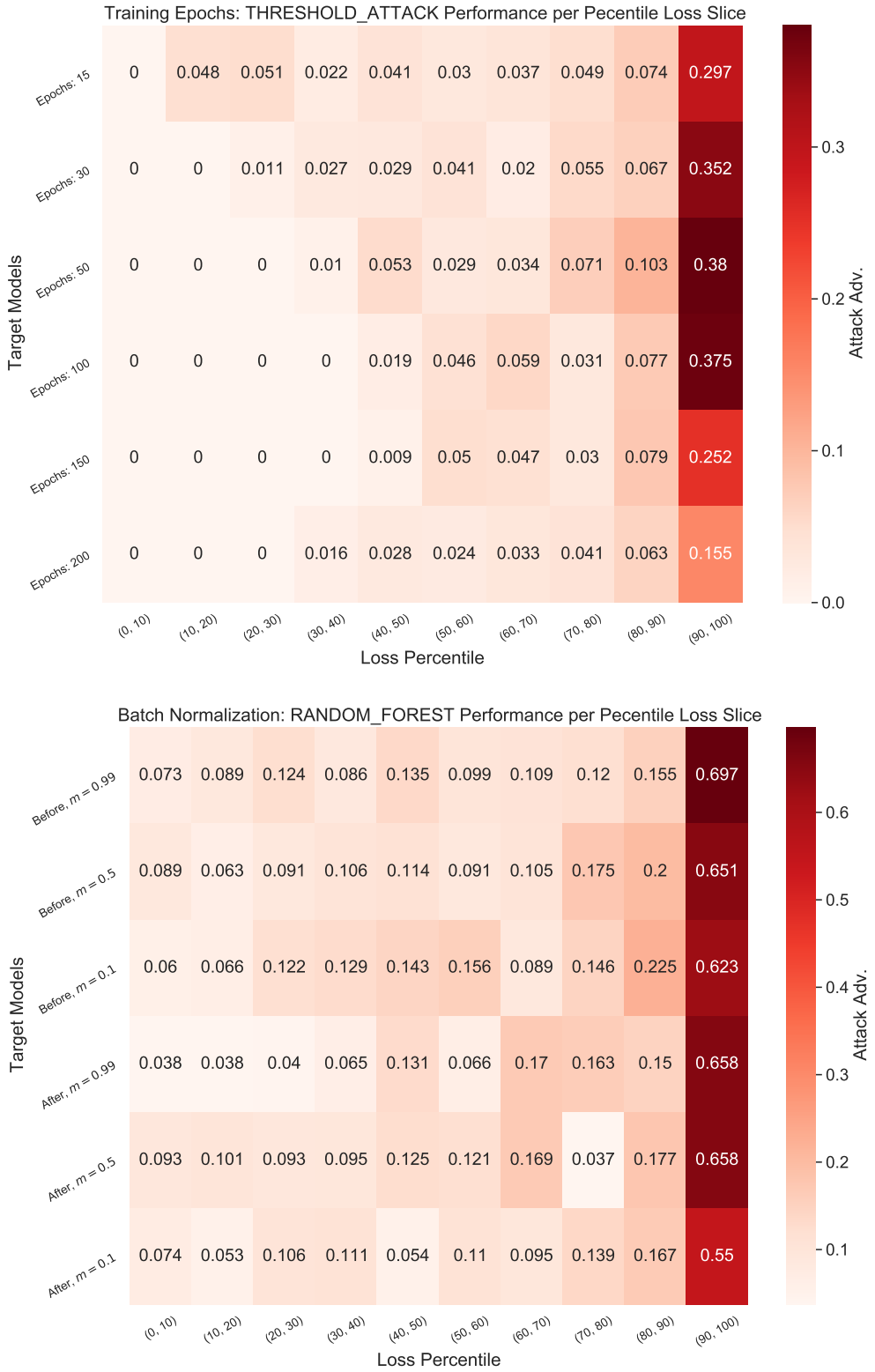


Figure 6.21.: MIAs performance on experiment variation models, Data Slice: Loss Percentiles.
Experiments: Training Epochs, and Batch Normalization.

7. Discussion

Before discussing the results of the experiments further, one should mention the limitations of the executed experiments to highlight the true weight of the registered results. The scope of this work is limited to studying the effects of architecture and training choices of a single NN architecture and that is the CNN so there are more NN and non-NN architectures that one can explore and experiment with. In addition, much more combinations of training implementations for the baseline model are theoretically interesting to experiment with but the results produced by the 94 variations of the baseline in this work showed the same consistent results and tendencies but it would definitely be more interesting to carry on similar experiments with different training choices and even on entirely different ML architectures. Therefore, the discussion and interpretations in this section are on the basis provided by results in the experiments described in chapter 5.

When looking at the performance of the MIAs launched on all models in every experiment, it was clear that the architectural choices, and the training hyper-parameters and implementations had an impact on the MIA vulnerability but the way to measure that impact was through the effects that these different aspects had on a variety of the tracked metrics, showing consistent correlations over different data slices.

Role of Influential Data Points. Previous research done by Truex et al. emphasized the important role that a limited set of data records can play in regards to a model's vulnerability in MIAs. According to the past work, these points had a significant influence on the decision boundary of the model and were the most exposed to the threat of the privacy attacks. According to the experiment results, one can enumerate aspects that make a data record (x, y) more influential making it more susceptible to MIAs. A data record (x, y)'s training membership status in a model T is easier to infer if:

- The model is unable to generalize well for data samples of class y (meaning that the validation accuracy for samples of class y is particularly low).
- T predicts the class of x incorrectly. This is because MIAs are much better at inferring membership for misclassified data than for correctly classified samples.

- T's prediction vector of x produces a relatively high loss value in comparison to the majority of losses produced by other data points.

Role of Overfitting and Impact of Regularization Techniques. The train-test accuracy difference and the train-to-test accuracy ratio both represent metrics that attempt to measure how overfit the model is to its training samples where the higher these two metrics are, the poorer the model is at making predictions for data it has not encountered during training. The correlations observed in those metrics with the attack advantage, especially when it comes to inspecting the attack's performance class-wise but also, with slightly less consistency, in regards to the performance on the entire dataset, shows that overfitting does play an important role in determining a model's risk to MIAs. The poorer the model at generalizing for data past its the records it sees in training, the more susceptible it is to MIAs. Variations of the baseline model T in the experiments handled the implementation of regularization methods, known to combat overfitting in NNs. Weight decay techniques did not contribute to reduce overfitting and none of them was able to drop T's exposure to MIAs significantly. The batch normalization layer, in all variations of its implementation, allowed for the previously mentioned two overfitting-tracking metrics to decrease in comparison to the baseline however not using any batch normalization layer and keeping the baseline as it is provides higher protection so decreasing overfitting did not stop the MIAs from performing better than expected, hinting to the existence of other aspects that the MIAs exploit in order to carry successful attacks. Shokri et al's [55] have suggested dropout as a way to combat overfitting and help against the privacy attacks. Results in the dropout layer experiment showed that this measure was the most efficient for the baseline model to reduce its vulnerability. Implementing dropout with high *threshold* values allowed to significantly decrease the threat from MIAs while implementing it with *threshold* values on the lower end either allowed the MIAs to perform slightly better or saw its performance unchanged. According to the experiments, dropout is the regularization technique that shielded T from the MIAs the most efficiently.

Role of Loss. According to the results of the experiments, MIAs are at their peak performance when they infer the membership status of a data point whose prediction through the target model T produces loss that is relatively higher than losses produced by other data records. In addition, the strong positive correlations observed between the attack advantage and the loss distribution difference of member and non-member records on the entire dataset and even class-wise hint at the important role that the prediction loss plays in determining the success of MIAs. When looking at loss values produced by non-member records, one can observe that the upper bound surpasses the upper bound of losses produced by member records and this is especially remarkable for incorrectly classified non-member samples but low loss

values were quite common for both member and non-member samples, especially for the correctly classified. As the criteria for the most influential points is described above, it is important to know what piece of information do the MIAs exploit to maximize their success. The top k probabilities experiment showed that MIAs are still able to attack with comparable degrees of success even if they are provided with just the top probability and its corresponding class. If T's prediction of a data record (x, y) is incorrect then it suffers from a higher loss value than its correctly classified counterparts and this is what the MIAs seem to exploit as they are able to infer membership status with one probability just as good as they do with ten. In addition, the threshold attack, which is the only non-trained attack in the experiments, solely relies on the computed loss values from the predictions in order to come up with its decision boundary. Among all variations, the threshold attack is one of the most consistent top performers, showing that no training is needed in order to launch an efficient MIA. This can also only stress the significant role that the loss values play when it comes to MIAs. The experiment that allowed to generate manual prediction vectors and feed them into MIAs saw the MIAs performing at a maximum 1.0 attack advantage when there is a clear separation between losses produced by member and non-member elements but when the manual predictions were intentionally generated to allow no distinction between the loss distributions of both data slices, near-zero attack advantage values were the best that all attack types could do.

8. Conclusion

In this thesis, MIA-relevant findings are discussed according to the most recent work where attack techniques, mitigation strategies and especially observations regarding what makes a MIA more successful are highlighted. The role of the architectural choices and training parameters of a NN was studied by launching MIAs on unique variations of a baseline CNN model and then evaluating the attack results. These MIAs, provided by TF Privacy Tools, are arguably running in optimal conditions, at least according to comparisons to the techniques implemented in related work.

According to the experiment results and observed correlations, architectural choices that generally decrease overfitting as in decreasing the gap of performance between on the training and validation datasets can benefit from more protection than MIAs. In terms of the baseline model used in the experiments, Dropout with a high *threshold* value was one architectural choice that allowed for the highest decrease in the train-test performance gap and the highest protection against MIAs. However, a more robust metric to follow would be the difference in the loss distributions between the training and validation datasets as a higher gap was consistently strongly correlated with more exposure to MIAs. The MIAs rely mostly on finding a decision threshold that separates the loss values produced by the training member samples' predictions and the loss produced by non-members. This makes the outlier points, the ones that are the most susceptible to MIAs, the same ones whose predictions produce the highest loss values. This interpretation is supported by the work of Sablayrolles et al. who presented a theoretical proof for the optimal MIA that only takes loss values as input for its attack. This also opens the path for more interesting questions for future works where one could investigate ML algorithms or architectures whose predictions allow for a clear distinction between the distribution of produced loss values between member and non-member elements. At the same time, if the optimal MIA truly relies on setting its decision boundary according to loss values then how much of a threat do MIAs really pose?

Appendices

A. Entire Dataset

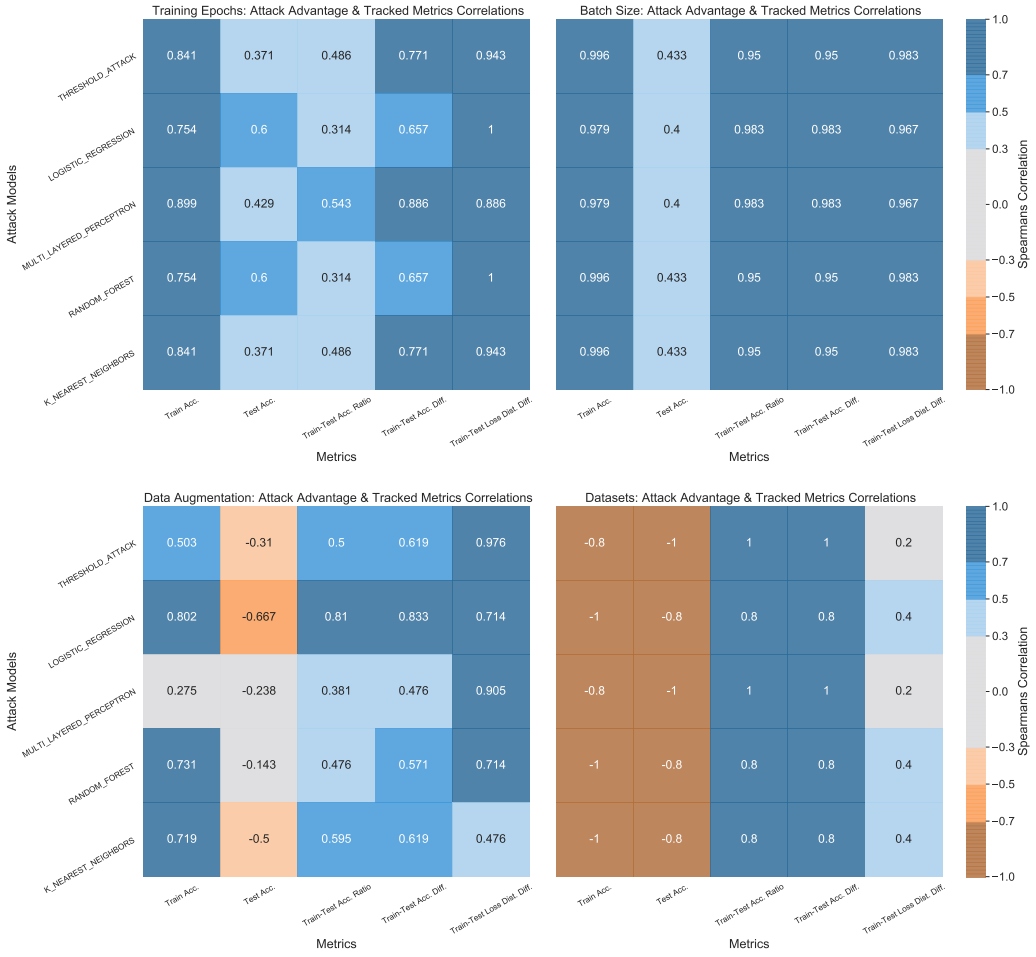


Figure A.1.: MIAs Attack Advantage and Variations Tracked Metrics Correlations Heat-map, Data Slice: Entire Dataset.
Experiments: Training Epochs, Batch Size, Data Augmentation, and Datasets.

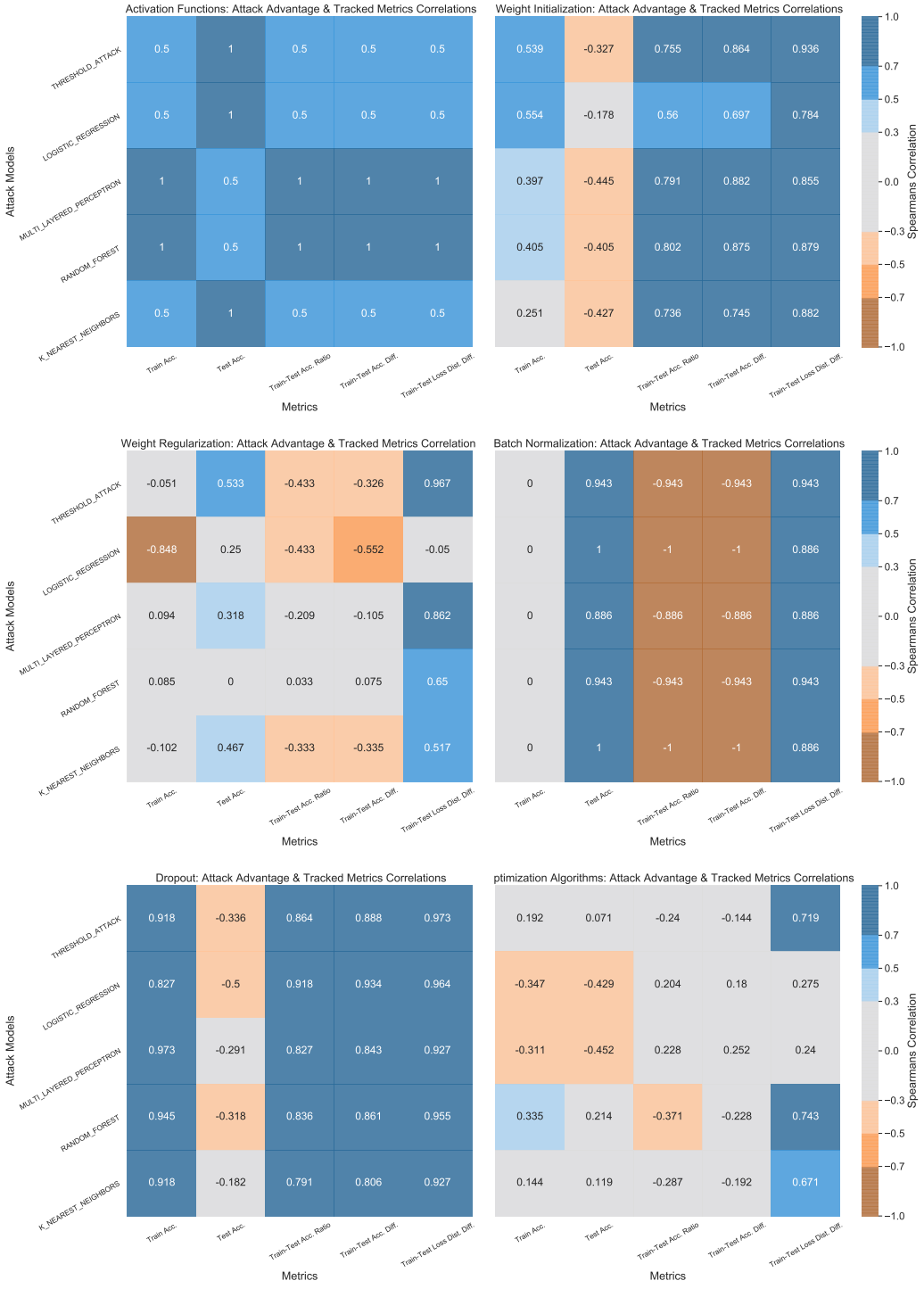


Figure A.2.: MIAs Attack Advantage and Validation Accuracy Correlations Heatmap, Data Slice: Per Class.
Experiments: Activation Functions, Weight Initialization, Weight Regularization, Batch Normalization, Dropout and Optimization Algorithms.

B. Per Class: Validation Acc.

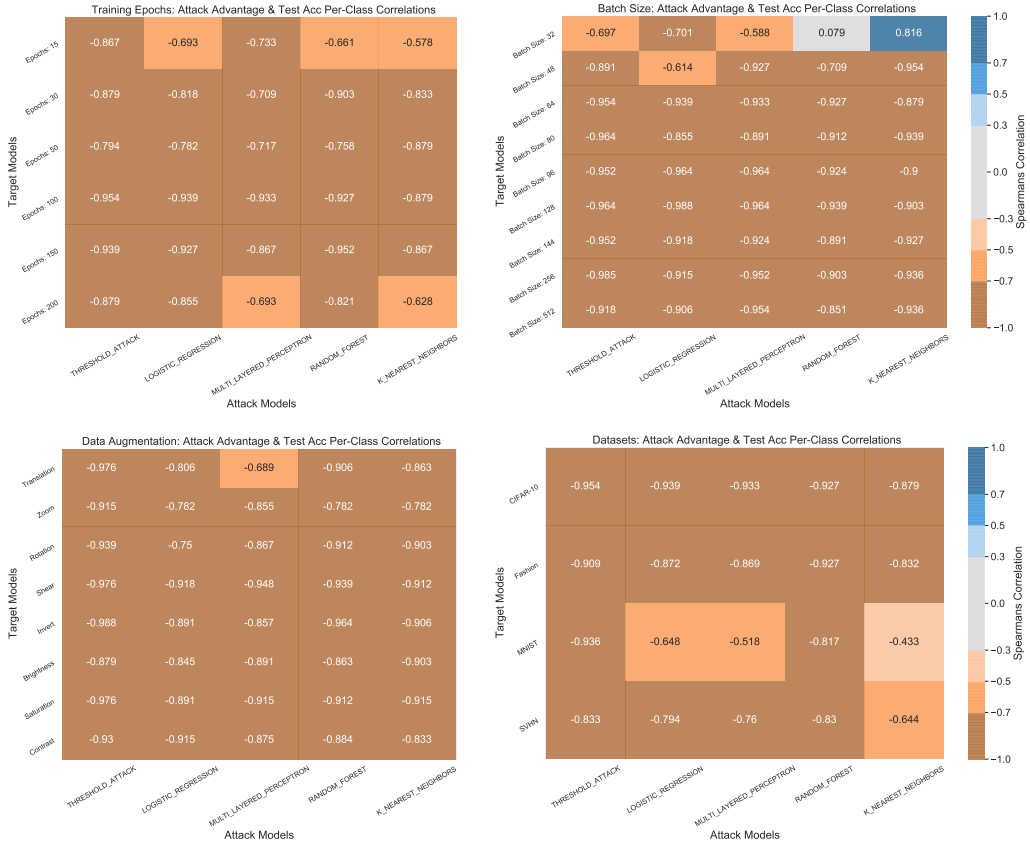


Figure B.1.: MIAs Attack Advantage and Variations Tracked Metrics Correlations Heat-map, Data Slice: Entire Dataset.
Experiments: Training Epochs, Batch Size, Data Augmentation, and Datasets.

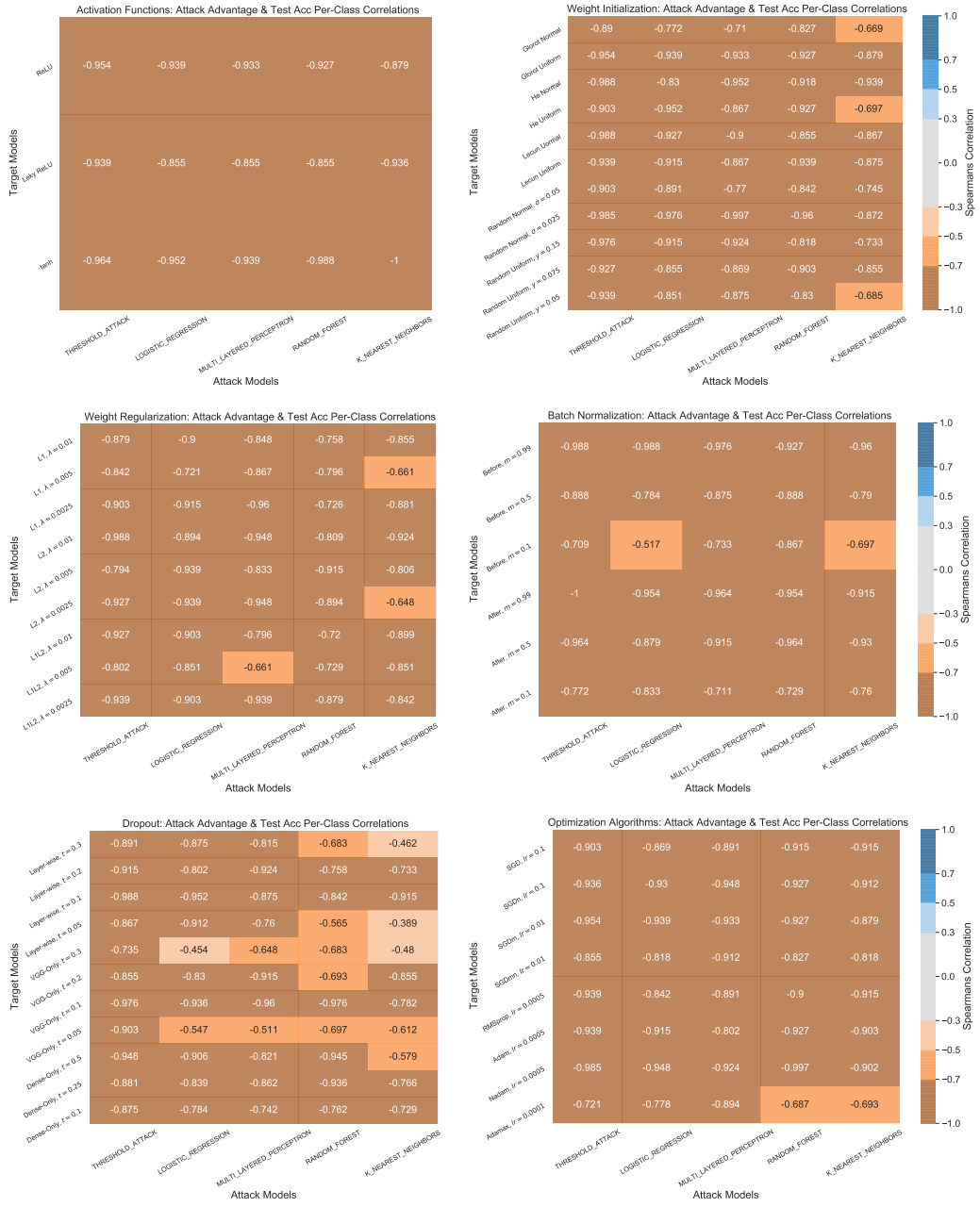


Figure B.2.: MIAs Attack Advantage and Validation Accuracy Correlations Heatmap, Data Slice: Per Class.
Experiments: Activation Functions, Weight Initialization, Weight Regularization, Batch Normalization, Dropout and Optimization Algorithms.

C. Per Class: Train-Test Acc. Diff.

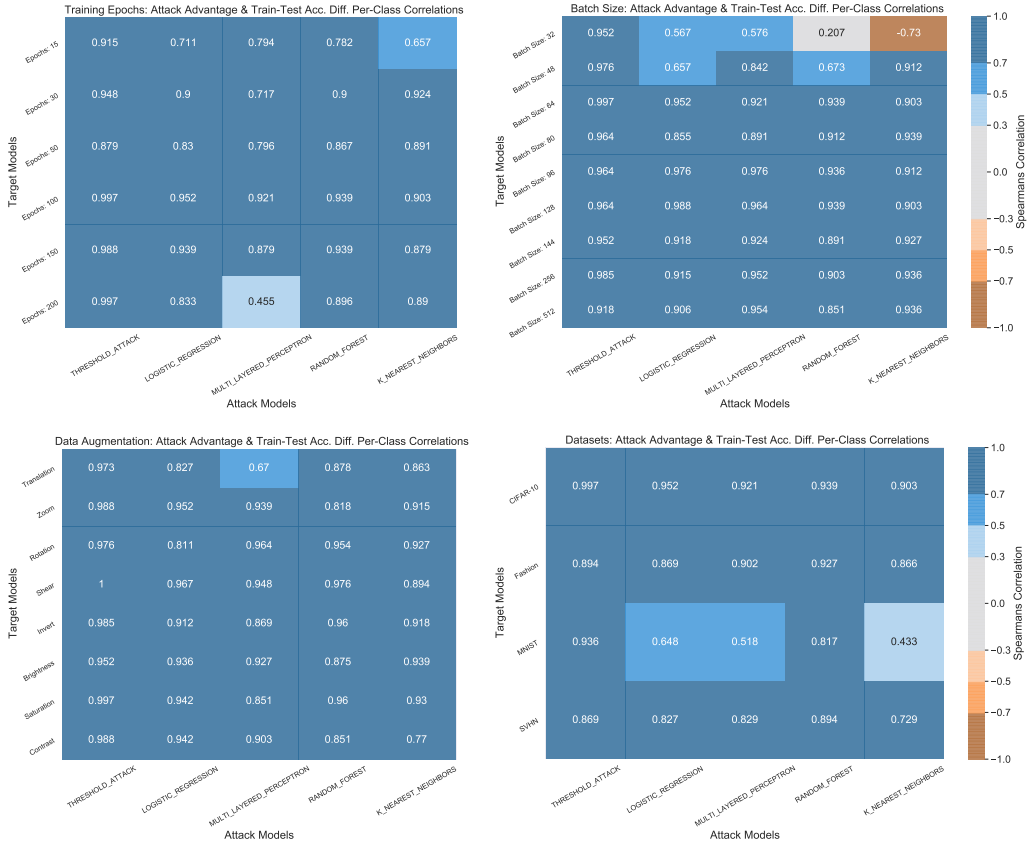


Figure C.1.: MIAs Attack Advantage and Train-Test Diff. Correlations Heat-map, Data Slice: Per Class.
Experiments: Training Epochs, Batch Size, Data Augmentation, and Datasets.

Figure C.2.: MIAs Attack Advantage and Train-Test Diff. Correlations Heat-map, Data Slice: Per Class.
Experiments: Activation Functions, Weight Initialization, Weight Regularization, Batch Normalization, Dropout and Optimization Algorithms.

D. Per Class: Train-Test Loss Dist. Diff.

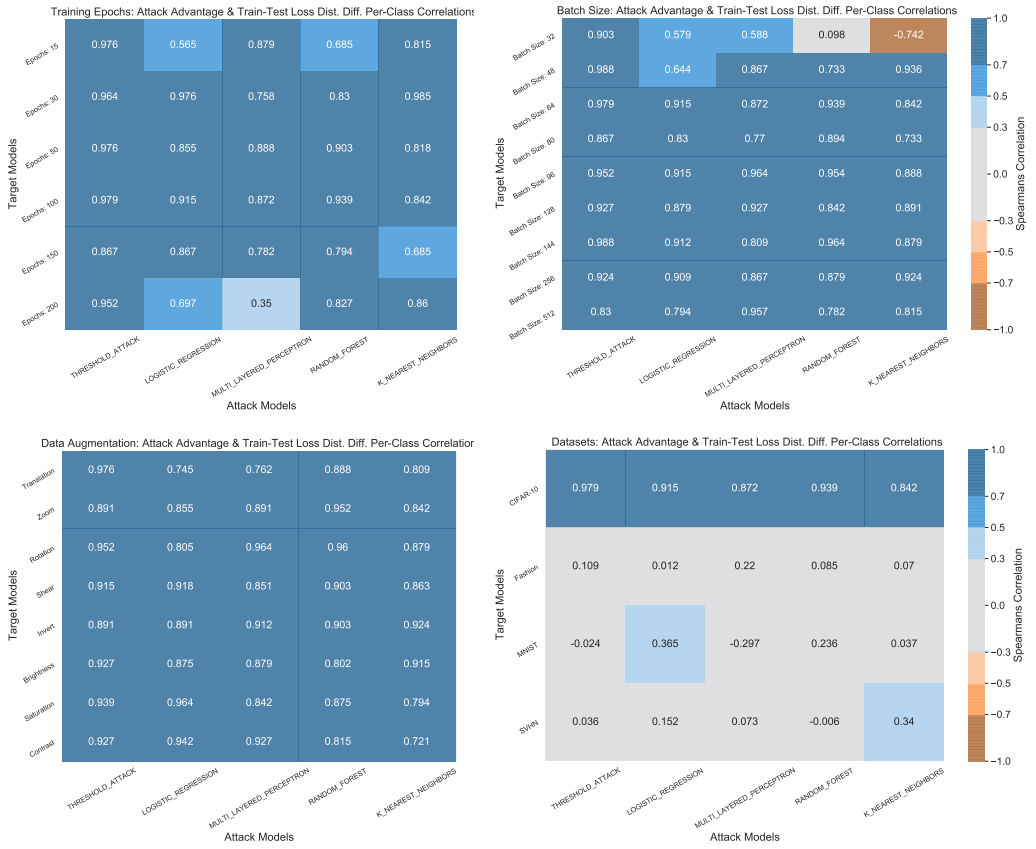


Figure D.1.: MIAs Attack Advantage and Train-Test Diff. Correlations Heat-map, Data Slice: Per Class.
Experiments: Training Epochs, Batch Size, Data Augmentation, and Datasets.

Figure D.2.: MIAs Attack Advantage and Train-Test Diff. Correlations Heat-map, Data Slice: Per Class.
Experiments: Activation Functions, Weight Initialization, Weight Regularization, Batch Normalization, Dropout and Optimization Algorithms.

E. Per Classification Correctness

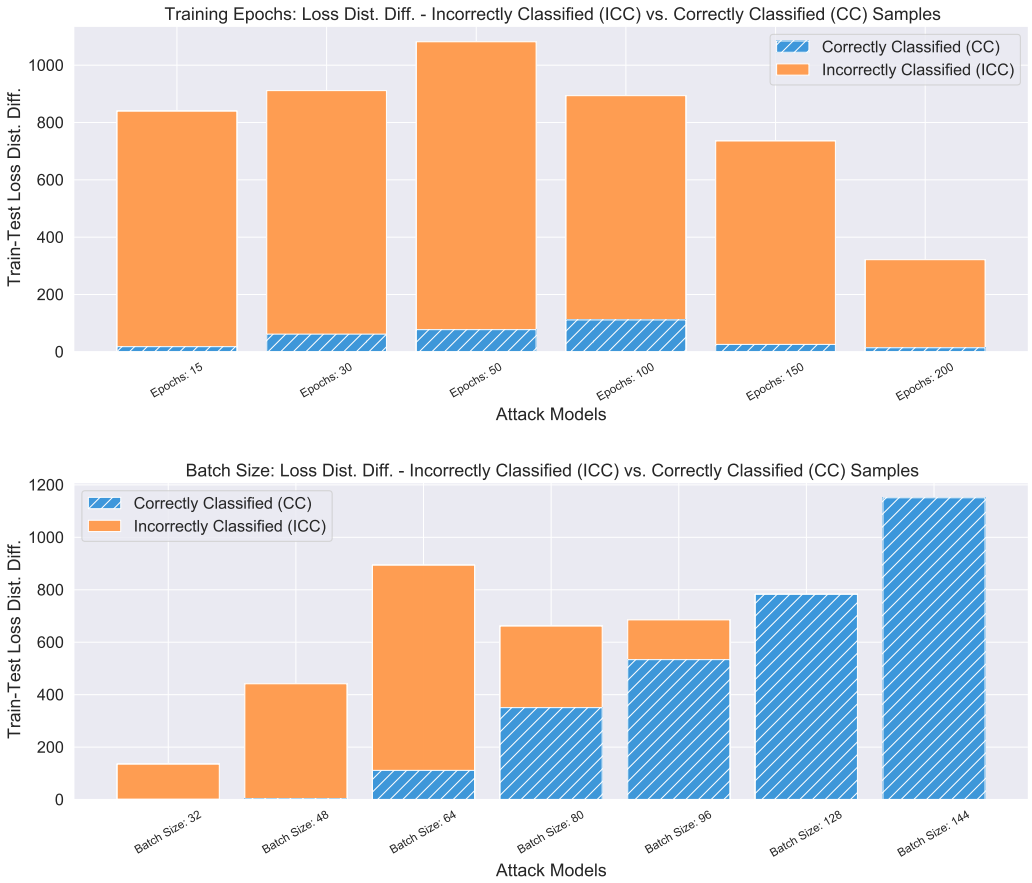


Figure E.1.: Loss Distribution Difference: Incorrectly Classified vs. Correctly Classified Samples.
Experiments: Training Epochs, and Batch Size.

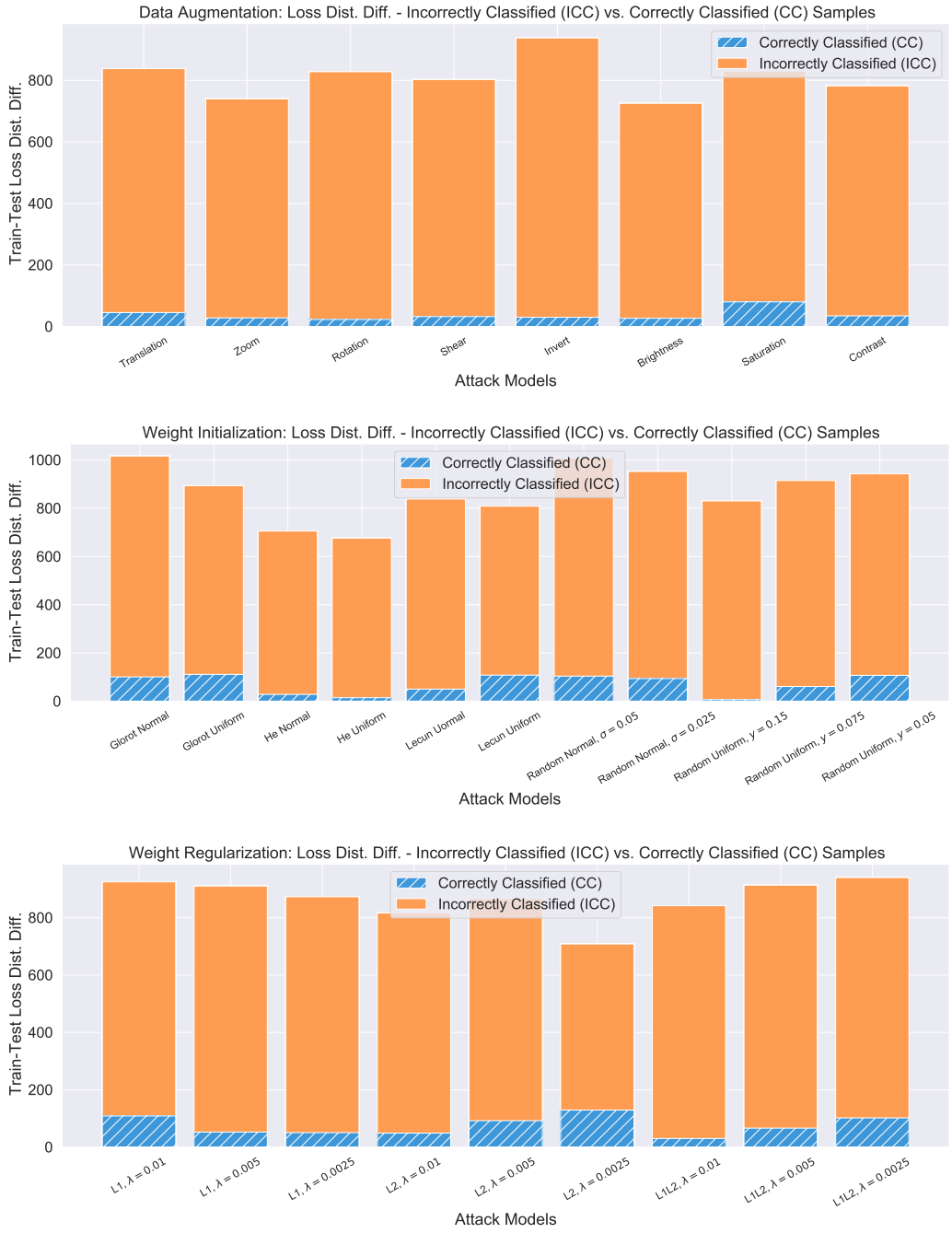


Figure E.2.: Loss Distribution Difference: Incorrectly Classified vs. Correctly Classified Samples.
Experiments: Data Augmentation, Weight Initialization, and Weight Regularization.

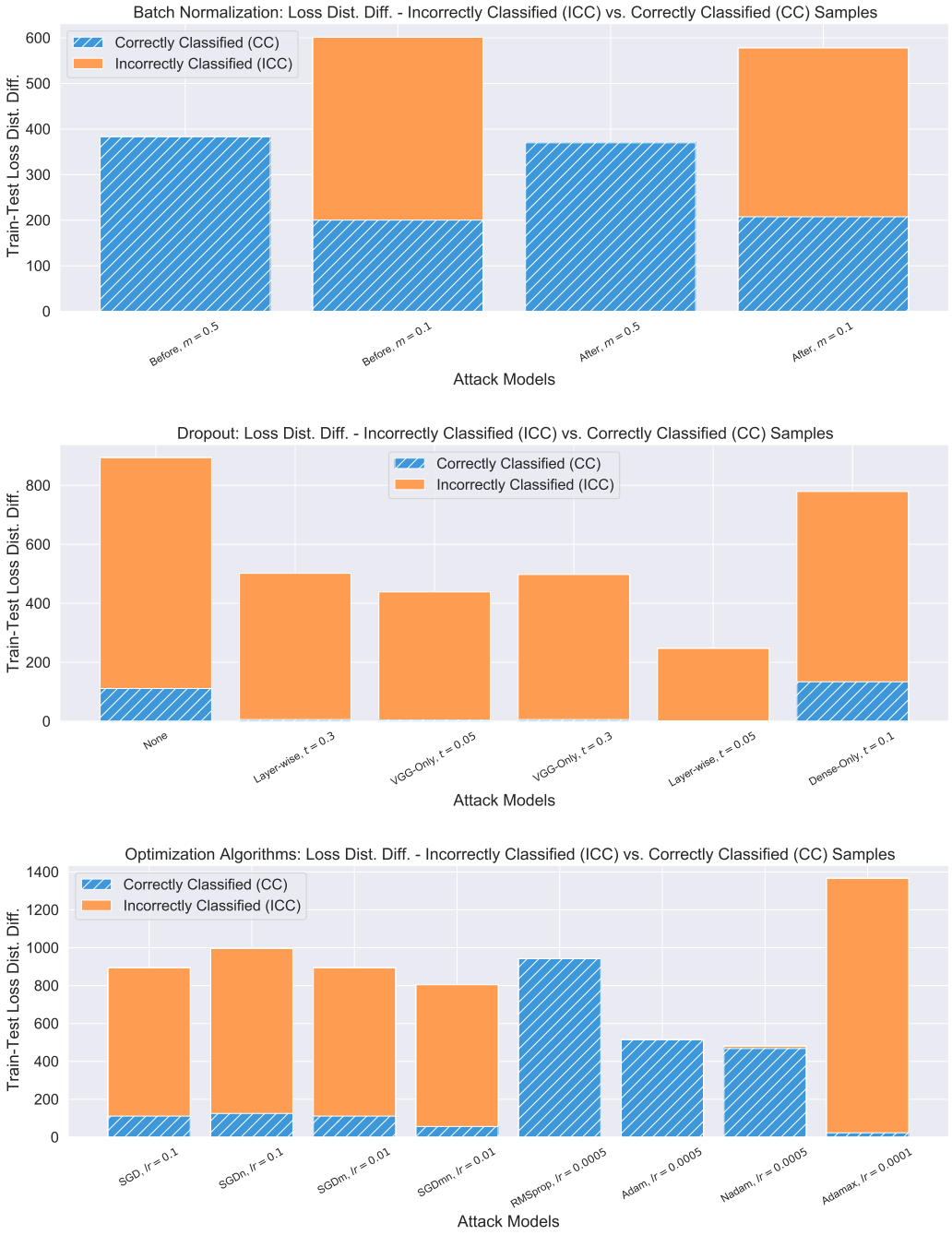


Figure E.3.: Loss Distribution Difference: Incorrectly Classified vs. Correctly Classified Samples.
Experiments: Batch Normalization), Dropout, and Optimization Algorithms.

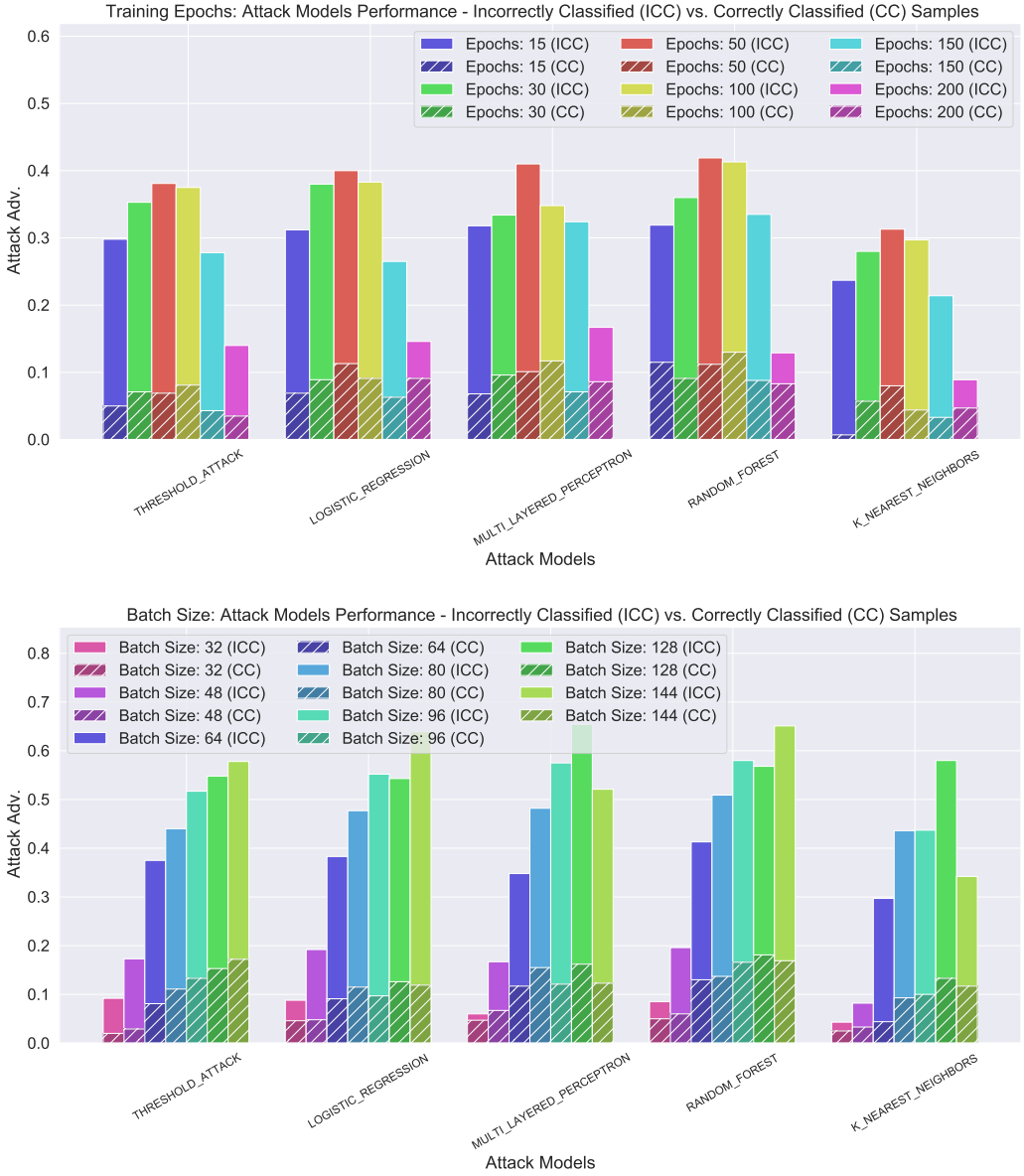


Figure E.4.: MIAs performance on experiment variation models, Data Slice: Per Class.
Experiments: Training Epochs, and Batch Size.

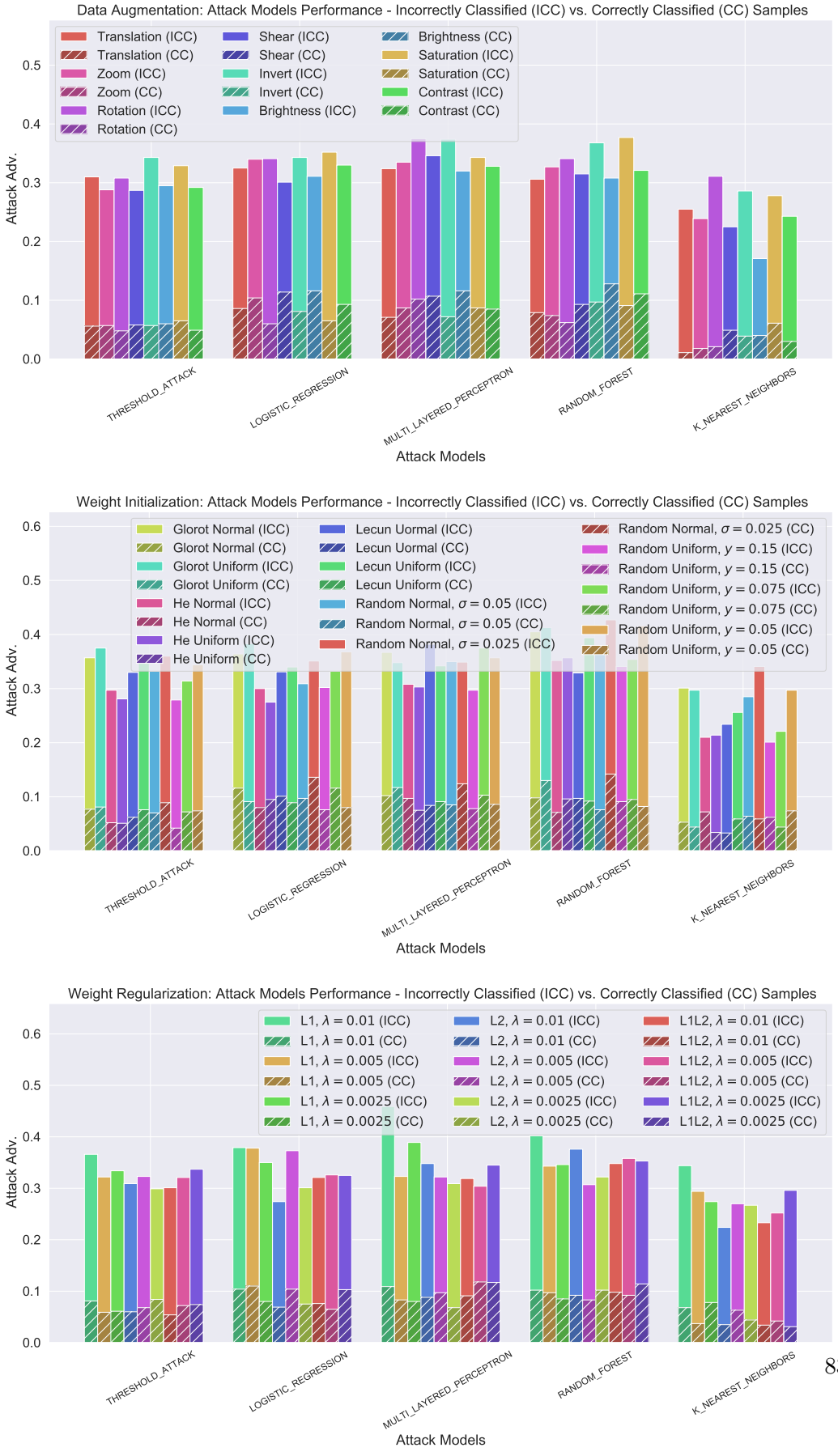


Figure E.5.: MIAs performance on experiment variation models, Data Slice: Per Class.
Experiments: Data Augmentation, Weight Initialization, and Weight Regularization.

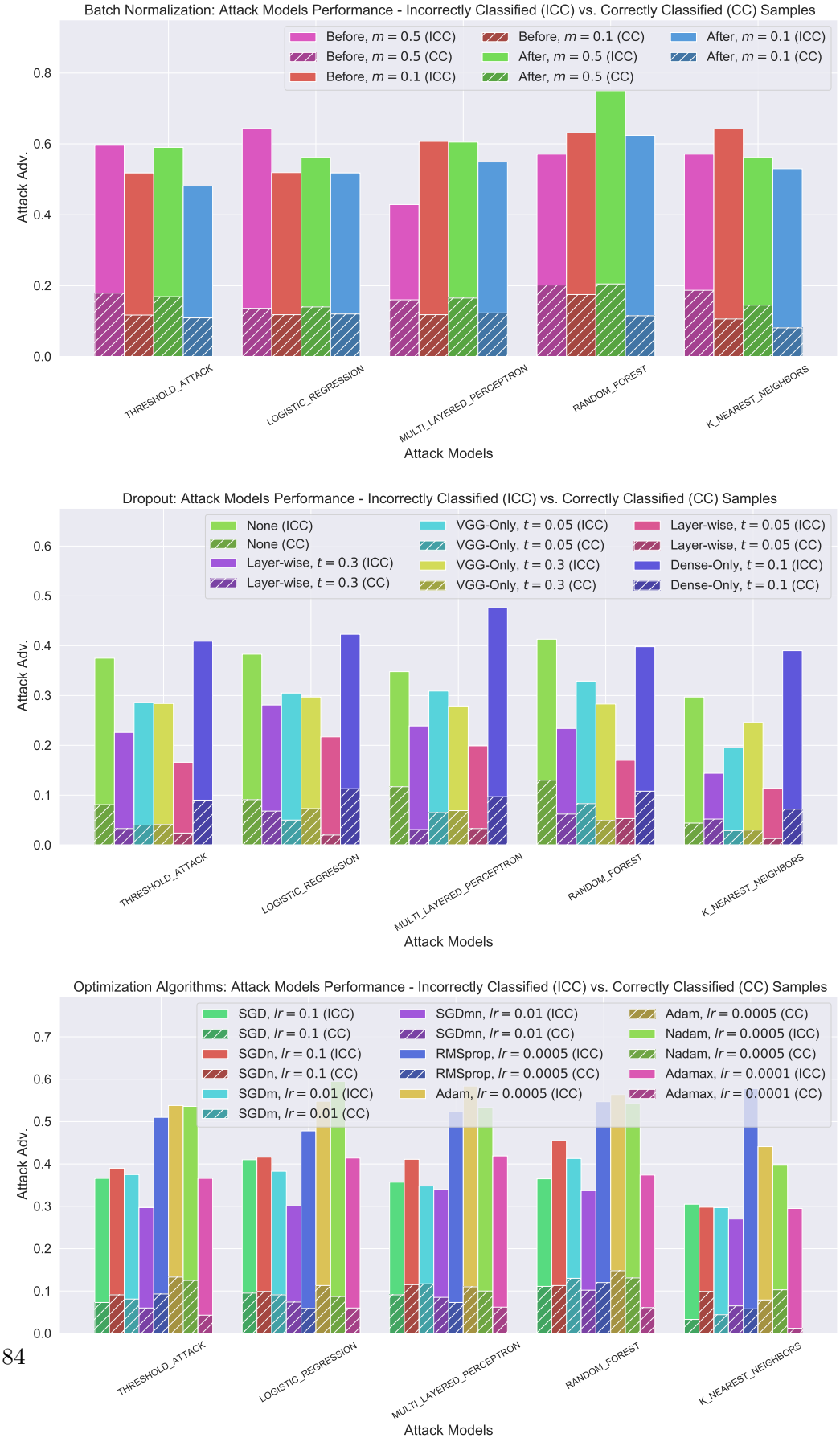


Figure E.6.: MIAs performance on experiment variation models, Data Slice: Per Class.
Experiments: Batch Normalization, Dropout, and Optimization Algorithms.

F. Per Loss Percentile

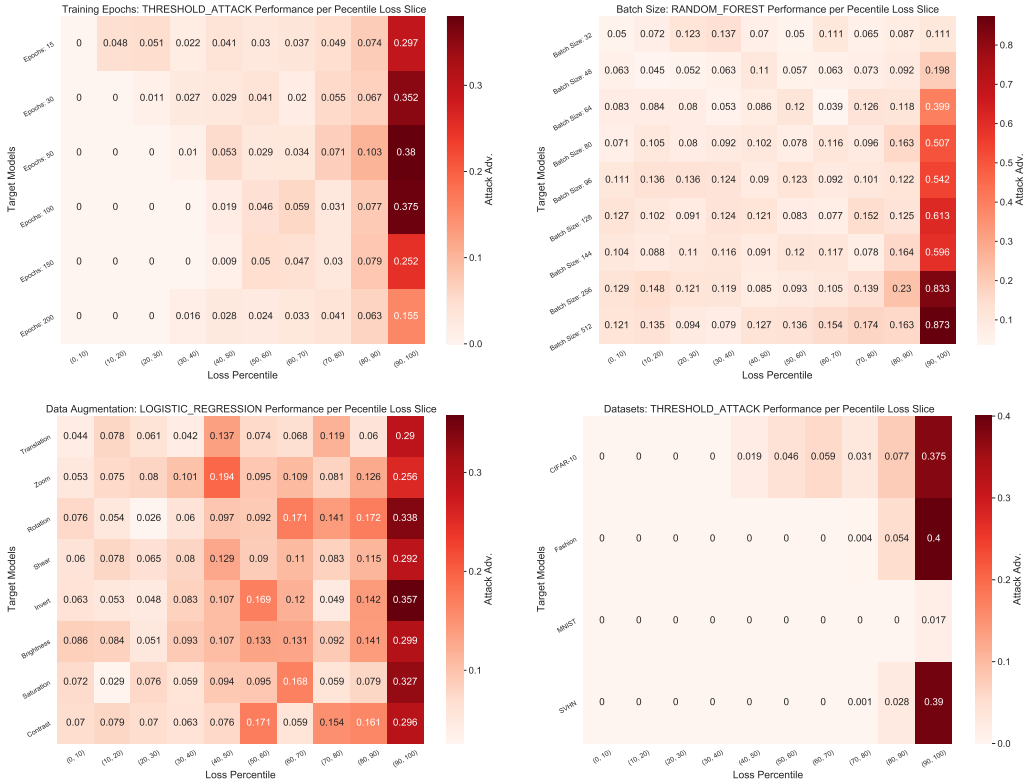


Figure F.1.: MIAs performance on experiment variation models, Data Slice: Loss Percentiles.
Experiments: Training Epochs, Batch Size, Data Augmentation, and Datasets.

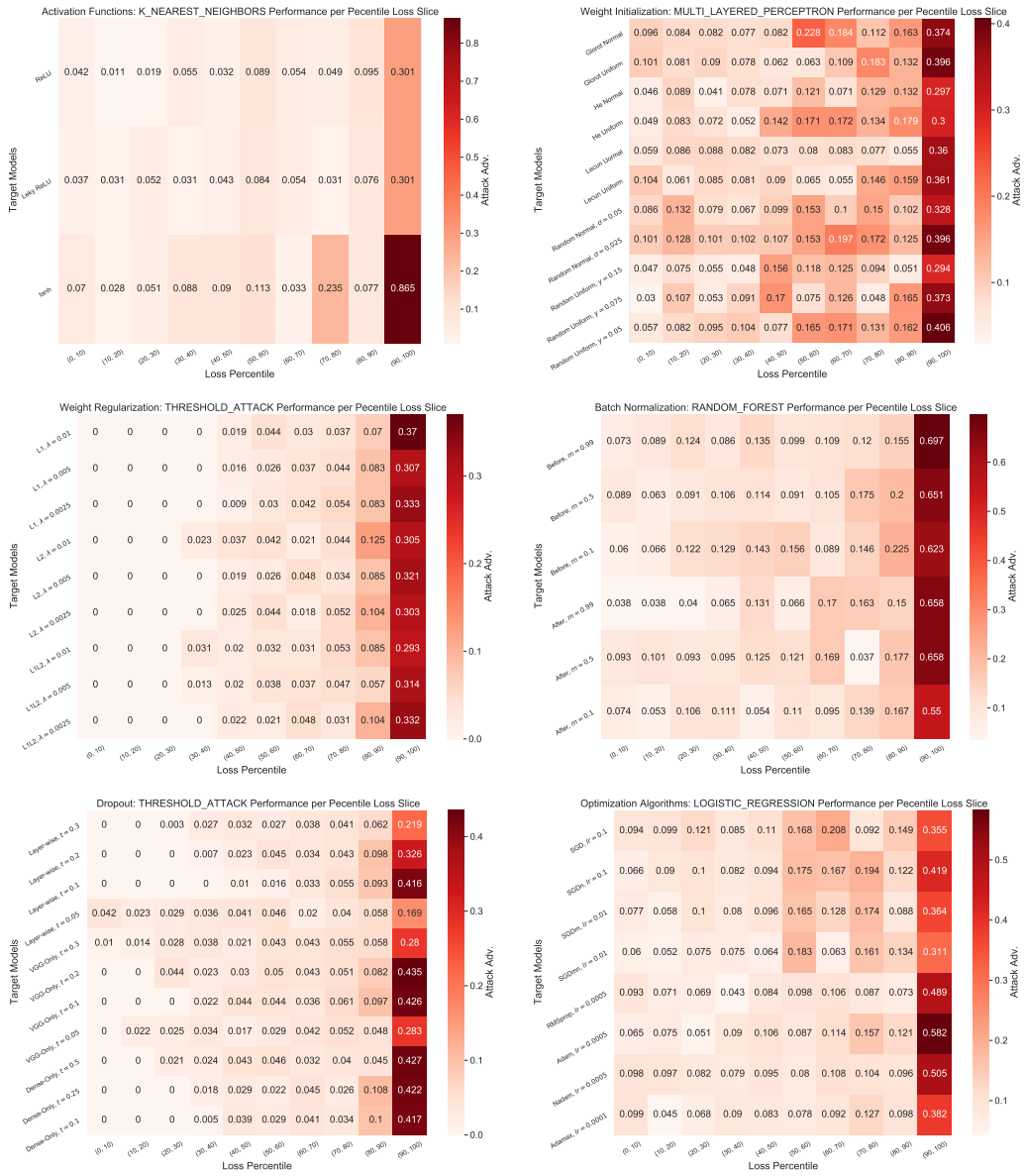


Figure F.2.: MIAs performance on experiment variation models, Data Slice: Loss Percentiles.
Experiments: Activation Functions, Weight Initialization, Weight Regularization, Batch Normalization, Dropout and Optimization Algorithms.

Bibliography

[1] Giuseppe Ateniese, Giovanni Felici, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, and Domenico Vitali. "Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers". In: *arXiv:1306.4447 [cs, stat]* (June 2013). arXiv: 1306.4447 [cs, stat].

[2] Vivek Bawa and vinay kumar. "Linearized Sigmoidal Activation: A Novel Activation Function with Tractable Non-Linear Characteristics to Boost Representation Capability". In: *Expert Systems with Applications* 120 (Nov. 2018). DOI: 10.1016/j.eswa.2018.11.042.

[3] Yoshua Bengio. "Practical Recommendations for Gradient-Based Training of Deep Architectures". In: *arXiv:1206.5533 [cs]* (Sept. 2012). arXiv: 1206.5533 [cs].

[4] *Big Data and Machine Learning in Health Care — Clinical Decision Support — JAMA — JAMA Network*. https://jamanetwork.com/journals/jama/article-abstract/2675024.

[5] Ekaba Bisong. "Google Cloud Machine Learning Engine (Cloud MLE)". In: Sept. 2019, pp. 545–579. ISBN: 978-1-4842-4469-2. DOI: 10.1007/978-1-4842-4470-8_41.

[6] Andrew P. Bradley. "The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms". In: *Pattern Recognition* 30.7 (July 1997), pp. 1145–1159. ISSN: 0031-3203. DOI: 10.1016/S0031-3203(96)00142-2.

[7] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. "On Empirical Comparisons of Optimizers for Deep Learning". In: *arXiv:1910.05446 [cs, stat]* (June 2020). arXiv: 1910.05446 [cs, stat].

[8] Jin-A Choi and Kiho Lim. "Identifying Machine Learning Techniques for Classification of Target Advertising". en. In: *ICT Express* 6.3 (Sept. 2020), pp. 175–180. ISSN: 2405-9595. DOI: 10.1016/j.icte.2020.04.012.

[9] *CHS Data Sets and Reports*. https://www.dshs.texas.gov/chs/data-reports.shtm.

[10] L. O. Chua and T. Roska. "The CNN Paradigm". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 40.3 (Mar. 1993), pp. 147–156. ISSN: 1558-1268. DOI: 10.1109/81.222795.

[11] *Convolutional Neural Network Architecture: Forging Pathways to the Future*. en-US. https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/.

[12] Marshall Copeland, Julian Soh, Anthony Puca, Mike Manning, and David Gollob. *Microsoft Azure: Planning, Deploying, and Managing Your Data Center in the Cloud*. en. Apress, 2015. ISBN: 978-1-4842-1043-7. DOI: 10.1007/978-1-4842-1043-7.

[13] Doan Cong Danh. *CIFAR10: 94% Of Accuracy By 50 Epochs With End-to-End Training*. en. https://blog.fpt-software.com/cifar10-94-of-accuracy-by-50-epochs-with-end-to-end-training.

[14] Emiliano De Cristofaro. "An Overview of Privacy in Machine Learning". In: *arXiv:2005.08679 [cs, stat]* (May 2020). arXiv: 2005.08679 [cs, stat].

[15] L. Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012), pp. 141–142. ISSN: 1558-0792. DOI: 10.1109/MSP.2012.2211477.

[16] Chandra Kusuma Dewa and Afiahayati. "Suitable CNN Weight Initialization and Activation Function for Javanese Vowels Classification". en. In: *Procedia Computer Science*. INNS Conference on Big Data and Deep Learning 144 (Jan. 2018), pp. 124–132. ISSN: 1877-0509. DOI: 10.1016/j.procs.2018.10.512.

[17] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. "Advertorch v0.1: An Adversarial Robustness Toolbox Based on PyTorch". In: *arXiv:1902.07623 [cs, stat]* (Feb. 2019). arXiv: 1902.07623 [cs, stat].

[18] Eustace Dogo, Oluwatobi Afolabi, Nnamdi Nwulu, Bhekisipho Twala, and Clinton Aigbavboa. *A Comparative Analysis of Gradient Descent-Based Optimization Algorithms on Convolutional Neural Networks*. Dec. 2018. DOI: 10.1109/CTEMS.2018.8769211.

[19] Timothy Dozat. "Incorporating Nesterov Momentum into Adam". en. In: (Feb. 2016).

[20] Cynthia Dwork. "Differential Privacy". en. In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 1–12. ISBN: 978-3-540-35908-1. DOI: 10.1007/11787006_1.

[21] Konstantin Eckle and Johannes Schmidt-Hieber. "A Comparison of Deep Networks with ReLU Activation Function and Linear Spline-Type Methods". In: *arXiv:1804.02253 [cs, stat]* (Sept. 2018). arXiv: 1804.02253 [cs, stat].

[22] *Elastic Machine Learning Algorithms in Amazon SageMaker — Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. https://dl.acm.org/doi/abs/10.1145/3318464.3386126.

[23] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1322–1333. ISBN: 978-1-4503-3832-5. DOI: 10.1145/2810103.2813677.

[24] *Full Article: Machine Learning in Automatic Speech Recognition: A Survey.*

[25] Xavier Glorot and Yoshua Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks". en. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, Mar. 2010, pp. 249–256.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1026–1034.

[27] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. "Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors". In: *arXiv:1207.0580 [cs]* (July 2012). arXiv: 1207.0580 [cs].

[28] Benlin Hu, Cheng Lei, Dong Wang, Shu Zhang, and Zhenyu Chen. "A Preliminary Study on Data Augmentation of Deep Learning for Image Classification". In: *arXiv:1906.11887 [cs, eess]* (June 2019). arXiv: 1906.11887 [cs, eess].

[29] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv:1502.03167 [cs]* (Mar. 2015). arXiv: 1502.03167 [cs].

[30] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980 [cs].

[31] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: *University of Toronto* (May 2012).

[32] Yann Lecun, Leon Bottou, Genevieve Orr, and Klaus-Robert Müller. "Efficient BackProp". In: (Aug. 2000).

[33] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Ng. *Efficient L1 Regularized Logistic Regression*. Vol. 21. Jan. 2006.

[34] Huimin Li, Marina Krček, and Guilherme Perin. "A Comparison of Weight Initializers in Deep Learning-Based Side-Channel Analysis". en. In: *Applied Cryptography and Network Security Workshops*. Ed. by Jianying Zhou, Mauro Conti, Chuadhry Mujeeb Ahmed, Man Ho Au, Lejla Batina, Zhou Li, Jingqiang Lin, Eleonora Losiouk, Bo Luo, Suryadipta Majumdar, Weizhi Meng, Martín Ochoa, Stjepan Picek, Georgios Portokalidis, Cong Wang, and Kehuan Zhang. Vol. 12418. Cham: Springer International Publishing, 2020, pp. 126–143. ISBN: 978-3-030-61637-3 978-3-030-61638-0. DOI: 10.1007/978-3-030-61638-0_8.

[35] Yisi Liu, Xiaojun Wang, Lei Wang, and Dongliang Liu. "A Modified Leaky ReLU Scheme (MLRS) for Topology Optimization with Multiple Materials". en. In: *Applied Mathematics and Computation* 352 (July 2019), pp. 188–204. ISSN: 0096-3003. DOI: 10.1016/j.amc.2019.01.038.

[36] Harshit Lohani, Dhanalakshmi Samiappan, and Hemalatha Venkatesan. "Performance Analysis of Extreme Learning Machine Variants with Varying Intermediate Nodes and Different Activation Functions: Proceeding of CISC 2017". In: Jan. 2019, pp. 613–623. ISBN: 9789811306167. DOI: 10.1007/978-981-13-0617-4_59.

[37] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. "Understanding Membership Inferences on Well-Generalized Learning Models". en. In: *arXiv:1802.04889 [cs, stat]* (Feb. 2018). arXiv: 1802.04889 [cs, stat].

[38] Shie Mannor, Dori Peleg, and Reuven Rubinstein. "The Cross Entropy Method for Classification". In: *Proceedings of the 22nd International Conference on Machine Learning*. ICML '05. New York, NY, USA: Association for Computing Machinery, Aug. 2005, pp. 561–568. ISBN: 978-1-59593-180-1. DOI: 10.1145/1102351.1102422.

[39] Dominic Masters and Carlo Luschi. "Revisiting Small Batch Training for Deep Neural Networks". In: *arXiv:1804.07612 [cs, stat]* (Apr. 2018). arXiv: 1804.07612 [cs, stat].

[40] *Membership Privacy — Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. https://dl.acm.org/doi/10.1145/2508859.2516686.

[41] *Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures — Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. https://dl.acm.org/doi/10.1145/2810103.2813677.

[42] Milad Nasr, Reza Shokri, and Amir Houmansadr. "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-Box Inference Attacks against Centralized and Federated Learning". In: *2019 IEEE Symposium on Security and Privacy (SP)* (May 2019), pp. 739–753. DOI: 10.1109/SP.2019.00065. arXiv: 1812.00910.

[43] Andrew Ng. "Feature Selection, L 1 vs. L 2 Regularization, and Rotational Invariance". In: *Proceedings of the Twenty-First International Conference on Machine Learning* (Sept. 2004). DOI: 10.1145/1015330.1015435.

[44] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning". In: *arXiv:1811.03378 [cs]* (Nov. 2018). arXiv: 1811.03378 [cs].

[45] Sungheon Park and Nojun Kwak. "Analysis on the Dropout Effect in Convolutional Neural Networks". en. In: *Computer Vision – ACCV 2016*. Ed. by Shang-Hong Lai, Vincent Lepetit, Ko Nishino, and Yoichi Sato. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 189–204. ISBN: 978-3-319-54184-6. DOI: 10.1007/978-3-319-54184-6_12.

[46] *Personal Privacy vs Population Privacy — Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://dl.acm.org/doi/10.1145/2020408.2020598.

[47] *Property Inference Attacks on Fully Connected Neural Networks Using Permutation Invariant Representations — Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*.

[48] Sebastian Ruder. "An Overview of Gradient Descent Optimization Algorithms". In: *arXiv:1609.04747 [cs]* (June 2017). arXiv: 1609.04747 [cs].

[49] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Herve Jegou. "White-Box vs Black-Box: Bayes Optimal Strategies for Membership Inference". en. In: *International Conference on Machine Learning*. PMLR, May 2019, pp. 5558–5567.

[50] C. Okan Sakar, S. Olcay Polat, Mete Katircioglu, and Yomi Kastro. "Real-Time Prediction of Online Shoppers' Purchasing Intention Using Multilayer Perceptron and LSTM Recurrent Neural Networks". en. In: *Neural Computing and Applications* 31.10 (Oct. 2019), pp. 6893–6908. ISSN: 1433-3058. DOI: 10.1007/s00521-018-3523-0.

[51] 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: *arXiv:1806.01246 [cs]* (Dec. 2018). arXiv: 1806.01246 [cs].

[52] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks". In: *arXiv:1312.6120 [cond-mat, q-bio, stat]* (Feb. 2014). arXiv: 1312.6120 [cond-mat, q-bio, stat].

[53] Juergen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 08936080. DOI: 10.1016/j.neunet.2014.09.003. arXiv: 1404.7828.

[54] P. Sermanet, S. Chintala, and Y. LeCun. "Convolutional Neural Networks Applied to House Numbers Digit Classification". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. Nov. 2012, pp. 3288–3291.

[55] 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)*. May 2017, pp. 3–18. DOI: 10.1109/SP.2017.41.

[56] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556 [cs].

[57] Nick Street, William Wolberg, and O Mangasarian. "Nuclear Feature Extraction For Breast Tumor Diagnosis". In: *Proc. Soc. Photo-Opt. Inst. Eng.* 1993 (Jan. 1999). DOI: 10.1117/12.148698.

[58] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. "On the Importance of Initialization and Momentum in Deep Learning". en. In: *International Conference on Machine Learning*. PMLR, May 2013, pp. 1139–1147.

[59] Luke Taylor and Geoff Nitschke. "Improving Deep Learning Using Generic Data Augmentation". In: *arXiv:1708.06020 [cs, stat]* (Aug. 2017). arXiv: 1708.06020 [cs, stat].

[60] *TensorFlow White Papers*. en. https://www.tensorflow.org/about/bib.

[61] *Tensorflow/Privacy*. tensorflow. Apr. 2021.

[62] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei. "Demystifying Membership Inference Attacks in Machine Learning as a Service". In: *IEEE Transactions on Services Computing* (2019), pp. 1–1. ISSN: 1939-1374. DOI: 10.1109/TSC.2019.2897554.

[63] Twan van Laarhoven. "L2 Regularization versus Batch and Weight Normalization". In: *arXiv:1706.05350 [cs, stat]* (June 2017). arXiv: 1706.05350 [cs, stat].

[64] Yingying Wang, Yibin Li, Yong Song, and Xuewen Rong. "The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition". en. In: *Applied Sciences* 10.5 (Jan. 2020), p. 1897. DOI: 10.3390/app10051897.

[65] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *arXiv:1708.07747 [cs, stat]* (Sept. 2017). arXiv: 1708.07747 [cs, stat].

[66] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. "Federated Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13.3 (Dec. 2019), pp. 1–207. ISSN: 1939-4608. DOI: 10.2200/S00960ED2V01Y201910AIM043.

[67] 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)*. July 2018, pp. 268–282. DOI: 10.1109/CSF.2018.00027.

[68] Xue Ying. "An Overview of Overfitting and Its Solutions". In: *Journal of Physics: Conference Series* 1168 (Feb. 2019), p. 022022. DOI: 10.1088/1742-6596/1168/2/022022.

[69] Hongwei Yong, Jianqiang Huang, Deyu Meng, Xiansheng Hua, and Lei Zhang. "Momentum Batch Normalization for Deep Learning with Small Batch Size". In: Oct. 2020, pp. 224–240. ISBN: 978-3-030-58609-6. DOI: 10.1007/978-3-030-58610-2_14.

[70] Jiaoping Zhang, Hsiang Sing Naik, Teshale Assefa, Soumik Sarkar, R. V. Chowda Reddy, Arti Singh, Baskar Ganapathysubramanian, and Asheesh K. Singh. "Computer Vision and Machine Learning for Robust Phenotyping in Genome-Wide Studies". en. In: *Scientific Reports* 7.1 (Mar. 2017), p. 44048. ISSN: 2045-2322. DOI: 10.1038/srep44048.