NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Membership Inference Attacks: Threat Analysis

Vissarion E. Moutafis

Supervisor: **Konstantinos Chatzikokolakis,** Associate Professor

ATHENS

JULY 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διαρροή Δεδομένων στην Μηχανική Μάθηση: Ανάλυση Κινδύνου

Βησσαρίων Μ. Μουτάφης

Επιβλέπων: **Κωνσταντίνος Χατζηκοκολάκης,** Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2022

BSc THESIS

Membership Inference Attacks: Threat Analysis

Vissarion E. Moutafis

S.N.: 1115201800119

SUPERVISOR: **Konstantinos Chatzikokolakis,** Associate Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διαρροή Δεδομένων στην Μηχανική Μάθηση: Ανάλυση Κινδύνου

Βησσαρίων Μ. Μουτάφης
Α.Μ.: 1115201800119

ΕΠΙΒΛΕΠΩΝ: Κωνσταντίνος Χατζηκοκολάκης, Αναπληρωτής Καθηγητής

ABSTRACT

In the new era of data, companies and organizations around the world offer Machine Learning Services as a tool for enhancing people's lives. Recommendation algorithms, search engine's, intra-orgranization usage in medicine, military etc. All of the above are working on top of continuous data streams which are getting larger and more rich on user data, day by day.

Users, unaware how their data are being used, accept terms and conditions, giving away the right of data privacy, participating in various machine learning experiments with the promise of each vendor's data anonymity process. The vendors are reassuring users that their data are safe and completely anonymized, ignoring the fact that the machine learning models, they so much strive to incorporate to their product flow, suffers from subtle vulnerabilities, which can be used to expose and identify users, along with their, otherwise, private data.

These types of attacks are called *Population Inference Attacks* and we are, specifically, going to deepen our knowledge and analyze with detail the so-called *Membership Inference Attack*.

In these attacks, the target uses a machine learning model trained on a secret 'target' dataset. On the other hand the attacker, tries to inference whether some user-victim is a member of this dataset. To display the danger posed by this attack consider the scenario where an attacker knows that the clinical records of a user-victim are part of a disease-related-model's training set, then the attacker can infer if the person has the disease with high certainty, leading to a serious privacy breach.

The goal of this thesis is to further examine, analyze and understand the mechanism, reasoning behind membership inference attacks against machine learning models, as well as the effect and the various ways we could prevent data leakage during training of ML models.

Throughout this thesis, a plethora of plots and boards will be provided to the reader, to enhance his/her understanding of this study via experiments.

SUBJECT AREA: Data Privacy, Machine Learning

KEYWORDS: Machine Learning, Neural Networks, Classification, Differential Privacy, Security, User data, Data Privacy

ΠΕΡΙΛΗΨΗ

Στην σύγχρονη εποχή, οι εταιρίες και οι οργανισμοί σε όλο τον κόσμο, χρησιμοποιούν υπηρεσίες μοντέλων μηχανικής μάθησης, ως ένα εργαλείο για την βελτίωση της ζωής των πελατών τους. Αλγόριθμοι συστάσεων ταινιών, παιχνιδιών και τάσεων, μηχανές αναζήτησης, αλλά και ενδοεταιρικές υπηρεσίες σε φαρμακευτικούς, στρατιωτικούς οργανισμούς, λειτουργούν βασιζόμενοι σε μια συνεχή ροή δεδομένων, η οποία συνεχώς αυξάνεται σε όγκο και πλήθος.

Οι χρήστες αγνοώντας το πως οι πάροχοι χρησιμοποιούν τα δεδομένα τους , συννενούν να παραδώσουν το δικαίωμα της ιδιωτικότητας των δεδομένων τους, βασιζόμενοι στον εκαστοτε οργανισμό για την διατήρηση της ανωνυμίας και της ιδιοτικότητας των ευαίσθητων πληροφοριών τους. Απο την άλλη πλευρά, οι πάροχοι κατευνάζουν τις όποιες ανησυχίες των χρηστών υποσχόμενοι πως χρησιμοποιούν τις τελευταίες τεχνολογίες ιδιοτικοποίησης δεδομένων, αγνοόντας το γεγονός ό,τι τα μοντέλα μηχανικής μάθησης, που τοσο πασχίζουν να βγάλουν στην παραγωγή, διαθέτουν ευπάθιες, οι οποίες διακιβεύουν τα ευαίσθητα, προσωπικά δεδομένα των χρηστών τους.

Επιθέσεις, τετοιου τύπου ονομάζονται *Population Inference attacks* και συγκεκριμένα εμείς θα ασχοληθούμε με τα *Membership Inference Attacks*.

Σε αυτές τις επιθέσεις, το μοντέλο θύμα, εκπαιδεύεται πάνω σε ένα κρυφό συνολο δεδομένων, που αποτελεί τον στόχο. Ο κακόβουλος χρήστης από τη δικιά του πλευρά, προσπαθεί να συμπαιράνει αν κάποιοι χρήστες-στόχοι ανήκουν μέσα στο παραπάνω σύνολο εκπαίδευσης. Ως προς της επίδειξη του κινδύνου μια τέτοιας επίθεσης σκεφτήκε ενα σενάριο όπου, ο επιτιθέμενος γνωρίζει πως τα κλινικά δεδομένα ενός χρήστη, χρησιμοποιήθηκαν για την εκπαίδευση ενός μοντέλου προβλεψεων σχετικών με μία ασθένεια. Ο επιτιθέμενος, γνωρίζει πλέον αν το θύμα έχει την ασθένεια ή όχι, εφόσον το μοντέλο του έχει εκθέσει τις προσωπικές πληροφορίες του θύματος.

Η εργασίας αυτή έχει ως στόχο ο αναγνώστης να εξετάσει, αναλύσει και κατανοήσει τον τρόπο λειτουργίας των Membership Inference Attacks, καθώς και τις επιπτώσεις τους και τις διάφορετικές άμυνες που μπορεί κανείς να λάβει για να αποφευχθεί η διαρροή δεδομένων κατά την εκπαίδευση μοντέλων μηχανικής μάθησης.

Για την καλύτερη κατανόηση και ενίσχυση των επιχειρημάτων που ακολουθούν, παρέχουμε διαγράμματα και πίνακες ως οπτίκά βοηθήματα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Προστασία και Ιδιωτικότητα Δεδομένων, Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μηχανική Μάθηση, Νευρωνικά Δίκτυα, Ταξινόμηση, Διαφορική Ιδιωτικότητα, Ασφάλεια, Δεδομένα Χρηστών, Προστασία Δεδομένων

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1. INTRODUCTION

1.1 Data Leakage in ML Models

In a world where machine learning, data science and automated decision making is the leading power of most organizations, user data are of the greatest importance. Unfortunately, that means that private user data are on the scope for many malicious individuals/organizations, aiming to exploit this knowledge for covering their own agenda. One of the ways privacy could be breached, is by exploiting inherit vulnerabilities found in the structure and nature of many ML models, deployed in public.

ML Models, by definition, try to fit training data, so that they can distinguish patterns on them and 'learn' how to make elaborated predictions on new input queries. This training routine, if not done with caution, has one particular disadvantage. The training samples, should not be skewed in any way and should be acquired randomly and uniformly across the population we want to describe, otherwise the statistical models over-fit on them and making non generalizable predictions, based solely on the patterns of the provided training set. This phenomenon is called over-fitting and it causes models to under-perform when making predictions on unseen input data. This situation is the reason behind ML models responding with abnormal confidence to some queries, while failing to hit random choice accuracy on others[15]. Over-fitting is the simplest vulnerability an adversary would exploit to infer private user information.

In this thesis, we are considering only classification algorithms, meaning that the model receives an input vector of features of any kind and it returns the predicted labeling for input object, along with the prediction confidence scores. These models seems to be the perfect candidate for our attack since we can use statistical inference methods to predict a victim's membership status.

On a different notion, the risk of breaching privacy has motivated respective defences, which, as we will see, are not always effective against more proactive and creative adversaries. So, why is over-fitting and data leakage directly connected? How do we detect such subtle vulnerabilities on our models? How could we be sure that our models are completely bulletproof and is that even possible when discussing ML algorithms? These are some of the questions, we are trying to answer in this Thesis.

1.2 Membership Inference Attacks

Given a target ML model and its training data-set, an adversary could use a Membership Inference Attack, in order to infer the membership status of users, respective to the target data set, resulting to exposure of sensitive private user data.

A membership inference attack could be described with the following example. Let's say that Alice, is a patient of hospital A and she is diagnosed with cancer. At the same time, hospital A has agreed to provide scrubbed patient data to a data analysis team, for the training purposes of a model that predicts the probability of a patient to have cancer, based on a medical feature set. Now let us say that Bob, one of the analysts, wants to infer if Alice is in this training set, so that he knows for sure if she has cancer or not. So Bob uses a membership inference classifier, such as the one we describe later and attacks the model. If the attack is successful, then sensitive medical information about Alice's health

are publicly disclosed.

Attacking Different Services Membership Inference Attacks have multiple stages and each one of them could be modified to fit the respective attack context. Throughout this thesis, we explore the different points of view of the attack and experiment on different models and data-sets, while observing each model's behaviour, on a given data-set. At the same time, we try to elaborate on the observed behaviour and comprehend the reasons behind the attack's success or failure. Proceeding, we provide the reader with a high level overview of the two main attack scenarios we encounter, with respect to the target model's framework interface.

Confidence Vector Membership Inference Attacks In this version of the Membership Inference Attack, the attacker has access to an output confidence vector, the ML service returns, after a query. The confidence vector is a k-dimensional vector of probabilities, representing the probability of the input object belonging to class $c \in [0, k]$. In our example, the data analysis team's model would have 2 labels, which would interpret the answer to whether the queried patient has cancer or not. The adversary would use those output vectors to infer membership of the victim user[10].

Nevertheless many ML classification frameworks provide either a slice of the confidence vector (i.e. top 3 classes) or a sole label along with its confidence probability score(top 1 class), making the membership inference's objective harder to acquire.

Label Only Membership Inference Attacks The Label-Only MIA aims to solve the problem above by incorporating different techniques, throughout the different attack phases, overcoming the edge case of top 1 class provided by the target model[2]. Referring back to our example, this means that the target model would provide only a label (0 or 1) and a confidence score, relative to that label.

This alteration, uses data augmentations and treats the attacking stage a little differently, in order to bypass the lack of information on classes' confidence probabilities. However, this alternative is also, directly connected to the over-fitting degree of the target model, along with the extreme confidence of predictions on queries alike the target's training data.

Note that the basic vulnerability that our attack exploits is the fact that the target model is over confident on predictions from its training set and not over-fitting solely, while there is a direct connection between the two[15]. Later on we discover that, in most attack cases, the data-set characteristics and the implicit effects of model behaviour on predictions play a significant role to the overall attack success.

1.3 Thesis' Goal

The main goal of this thesis is to understand the basic mechanisms of Membership Inference Attacks and elaborate on the fact that ML Models have inherit vulnerabilities, with data inter-relations and varying model behaviour on different data-sets, being at the very center of them.

The following chapters aim to thoroughly explain the attack flow and the mechanics of every stage in a MIA and elaborate on the reason the attack works, firstly on a theoretical

basis and, secondly, according to experimental results, plots, boards, etc. Afterwards, we focus on some slight alterations of the main attack, in order to broaden our understanding of it.

Finally, we also provide the reader with a high level overview of proposed defences and some experimental results from our research based on the theory behind those defences. In the same spirit, we also discuss of how MIAs would overcome some of the most famous defences and we search for the roots of this particular fact.

2. BACKGROUND

2.1 Machine Learning

Machine Learning (ML) is one of the most advanced and continuously growing fields of our time. It focus expands on creating prediction models and using statistical algorithms for predicting the behavior of an object, given a subset of the object's features. Essentially in ML we call this set of objects a dataset

$$D = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N} \sim p,\ x_i \in \mathsf{R}^D,\ N, D \in \mathsf{N}^+$$

where $\boldsymbol{x}_i$'s are representing the objects' feature vectors, the independent variables and $\boldsymbol{y}_i$ are representing the true output values, the dependent variables, that the ML model tries to predict.

To apprehend this task, we use a so-called learning algorithm that is nothing more that the iterative solution of an optimization problem. The objective function of this problem is called the loss function and usually refers to the 'distance' between our predicted values $\hat{y}_i$ and the true values y_i. Let $f(x, w)$, be our statistical predictor, where x is the input value and w is our algorithms weights, which consist the trainable terms of our function. Also, let $L(f(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y})$, be our loss function. The objective function of this optimization problem is usually in the form of

$$\min_{\boldsymbol{w}} L(f(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y})$$

By solving this equations, we usually derive either a closed form solution or an iterative weight update rule like the following

$$\boldsymbol{w} = \boldsymbol{w} - \eta \cdot \nabla L(f(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y}).$$

This final equation wraps up on of the most famous update rule based algorithms, gradient descent.

Classification is a supervised learning technique that is used when the problem lies to assigning distinct classes or "labels" to an entity, based on a feature set. Given an input vector $\boldsymbol{x}$, the ML model outputs a k-dimensional probability vector $\boldsymbol{y}$, where k is the number of possible classes. Following we define some key terminology,

- **Decision Boundary** is the region of problem space, where the classifier predictions are ambiguous. The reader can think of this area as a line in a 2-D vector space defined by the equation $y = \boldsymbol{w}^T \cdot \phi(\boldsymbol{x}),\ x, w \in \mathsf{R}^D$, where $\boldsymbol{w}$ are the tuned weights and $\phi(\boldsymbol{x})$ is a transformation of the input space.
- **Binary Classifier** is a classifier that uses one single decision boundary to predict the binary class of an input datapoint (0, 1).
- **One vs All Classifier** is a group of binary classifiers, discriminating their respective class from all others used to solve the multi-class classification's objective.
- **Linear Classifier** is a classifier that uses a linear function to create decision boundaries. Most famous example of this classifiers are the logistic regression classifiers.

- **Non Linear Classifier** is a classifier that uses a non linear decision boundary, usually used when input distribution follows a non-linear behaviour. Classic classifiers with non-linear decision boundaries are neural network classifiers and knn classifiers.

2.2 Neural Networks

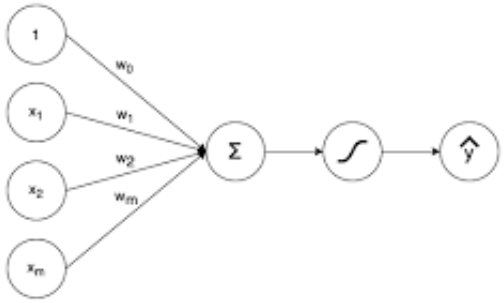

Figure 2.1: Artificial Perceptron

Neural Networks are a subset of ML algorithms and organization interest has being growing large for the last decade, although their creation goes back enough.

The core of a neural network is called a perceptron, like the one in Figure 2.1. The perceptron is consisted of the following parts

- **Input weights**: The weights that resemble to each one of the input features, plus one bias weight. These are also called **parameters** of the perceptron.
- **Aggregation function**: Is usually a sum function that sums up the information the perceptron has acquired from input.
- **Non Linearity**: A non linear function like sigmoid, ReLU, ELU, etc, which enables the perceptron to detect non linear patterns in features without the use of any input transformation.

In the end, we can interpret the described behaviour mathematically through the following equation

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W}),$$

where $\boldsymbol{X}$ and $\boldsymbol{W}$ are the matrix equivalent of $\boldsymbol{x}$ and $\boldsymbol{w}$

This basic paradigm can be expanded to perceptrons with multiple exit nodes and even multiple perceptrons connected together, creating layers and resulting to what we call a neural network. For example two perceptron with all inputs and outputs connected to each other is called a **dense** layer, like the one below.

Following we need to train the neural network to identify patterns in the input feature space. As we have already stated, neural networks can learn non linear patterns with the use of

a non linearity transformation. In order to adjust the NN's weights accordingly so that we converge to a better solution we use the empirical loss

$$\boldsymbol{J}(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} L(f(x^{(i)}; \boldsymbol{W}), y^{(i)}),$$

where $x^{(i)}$ is the i-th input vector, $y^{(i)}$ is the i-th target value and n is the size of the dataset.

In our case, we care more about solving classification problems with NNs, so we use the cross entropy loss function

$$\boldsymbol{J}(\boldsymbol{W}) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log(f(x^{(i)}; \boldsymbol{W})) + (1 - y^{(i)}) \log((1 - f(x^{(i)}; \boldsymbol{W}))),$$

and it's multi-class variation.

Goal of training is to solve the following minimization problem

$$\boldsymbol{W}^* = argmin_{\boldsymbol{W}} J(\boldsymbol{W}).$$

This task is easily apprehended via the gradient descent algorithm, which uses the derivative of the loss function and attempts convergence to the global minima of $J(\cdot)$.

First part of training is the so called, forward pass that estimates the output predicted value and the loss quantity that represents the pass through the whole NN. After that we proceed to the phase of backpropagation, where the derivatives of the loss function at each level is propagated through the whole neural net and the weights are being updated.

Convolutional Neural Networks or CNNs are another great part of the Neural Networks field and are defined by 2 basic layers

- **Convolutional Filters** which aim to extract features from the data and might even reduce the feature dimensions
- **Pooling Layers** which aim to downsample the feature maps after a convolution layer and are used for generalization reasons

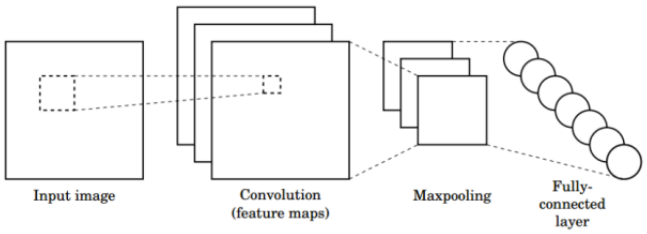

Figure 2.2: Convolutional Neural Network

Usually CNNs are used with image-like inputs but they can also be used with time series and any other input, from which features could be extracted. At the very end of a convolution and min/max pooling layer sequence, there is often a series of densely connected layers that learn the features and generate the predicted values, or as prefered in our case, predicted classes.

3. CONFIDENCE VECTOR MEMBERSHIP INFERENCE ATTACK

In this section, we formalize the membership inference attacks against ML models, stating the following question: *Given an instance x and black box access to a classification model F_t trained on a data set D_t, can an adversary infer whether $x \in D_t$, with high confidence?*. To answer this question, Shokri et al [10] turned Machine Learning against its self and exploited inherit and fundamental vulnerabilities of ML Models and leak meta-data about the training data set, achieving highly confident inference models, that could leak the membership features of victims, breaching their privacy.

The definition above points out that MIAs are more interested in the performance and behaviour of F_t on data inside and outside of D_t, than on the actual data content itself[5] [10][13]. This fact is realized in other studies on ML data leaks, where Yeom et al[15] infers conclusions about the data used to train an ML model, using threshold tests on model performance and loss metrics. Following, we provide the reader with the assumptions of the original confidence vector based membership inference attack[10] and a general high level outline of this chapter.

Before we continue any further, we note that one of the basic privacy-breaching vulnerabilities, that this attack exploits, is that the target model is in some way overfitted on its training dataset and because of that it may leak important information about the data, through the way it behaves on input data, based on their existence in its training (target) data set. Note that over-fitting is not a necessary factor for a model to manifest MIA-related vulnerabilities, but this is the most common case in most vulnerable models.

Following the definition of MIA attacks along with acknowledging the fact that most popular public ML models today are used by the MLaaS (Machine Learning as a Service) paradigm, we can state the basic adversary assumptions. Firstly, given a target model F_t and its training set D_t, we assume that the adversary is able to generate data samples, from a similar distribution as the training data. This data set is called Attacker's Dataset, D_a and it is completely disjoint to the target's training set, meaning that $D_t \cap D_a = \emptyset$. Following, we should consider that the attacker has some kind of black box access[1] to a trainable model, which is of the same type as the target model. This is quite common in MLaaS, since there are different services provided by the organization, that increase both the utility of those models and the exploitation surface. Nevertheless, having black box access to those models, prevents exposition of hyper-parameter tuning details and other model meta-data to any casual user. So, in that spirit, we should now proceed to a high level overview of a membership inference attack.

At the first stage of a MIA attack, the adversary accumulates as much data similar to the training data set, as he can. This can be accomplished via various ways, such as Statistic Based Generation[13], Active Learning Generation[13], API Querying[10][13], Region Based Generation[13] and others. The next step is consisted of training a bundle of **shadow models**. Although we explore the shadow paradigm in greater detail later on, we note that shadow models are ordinary ML models, used for imitating the behavior of the target model. This step is called Shadow Training.At the end of this pipeline, we train an attack model. By querying the shadow models, we build a new data set, the Attack Model's Data set, that we use to train our attack model. The latter is used to infer membership of a data point, provided only with an output vector, acquired after a query on the target model (Figure 3).

[1] Usage of the mechanism as a components, with no knowledge of internal settings

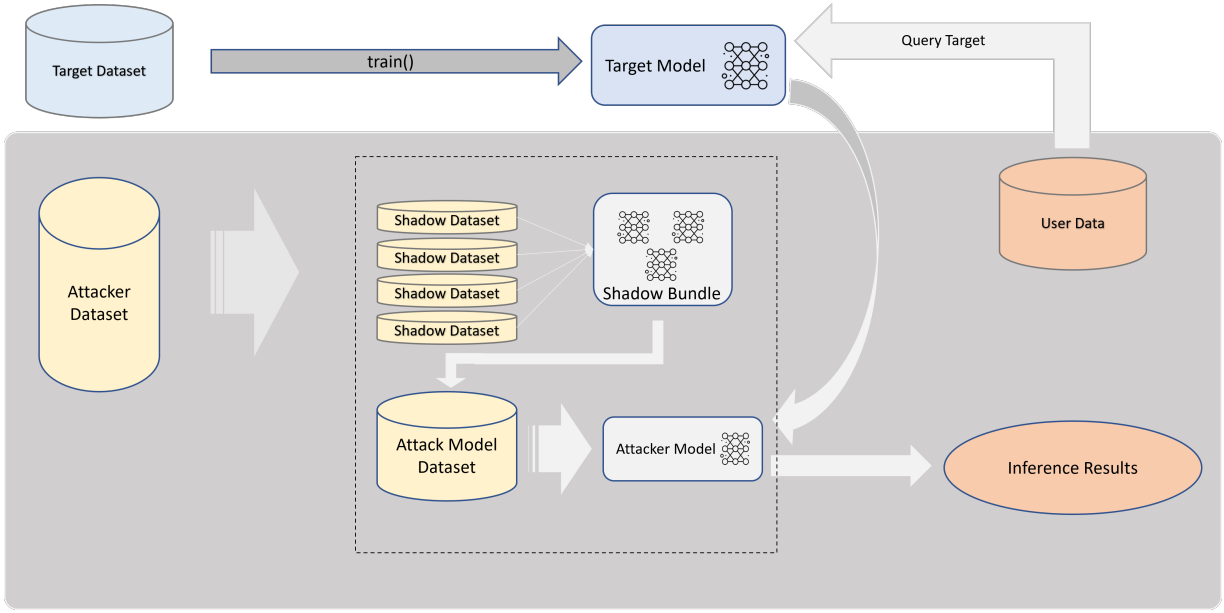

Figure 3.1: Overview of a Membership Inference Attack

In our experimental attacks we incorporated models from Shokri et al[10], specifically the CNN models and tried to adjust the hyper parameter tuning in order to get the best model in every case. In the same spirit we tried to alter the Shokri et al.[10] proposed attack model architecture and documented our results, along with creating plots to elaborate further on the points we make throughout this whole chapter.

3.1 Attacker's Dataset

So far, we stressed the importance of providing the attack mechanism with a high quality dataset, with respect to similarity to the target's training data set. This data set is also called "Shadows' Dataset". In this section, we explore the different ways an adversary can acquire this kind of dataset and determine the best way to do it, under various context.

Statistical Based Generation First, main technique of acquiring a data set to start our attack with, is Statistical based generation. Statistical based generation[5][10][13], utilizes informed sampling techniques in order to make a synthetic data set that is from the same distribution as the target data set. This method, assumes that the attacker has some access to meta-data about the distribution, such as the mean and standard deviation of the marginal distribution for most features.

This technique utilizes, statistical information to apply informed sampling, based on adversary's a priori knowledge, resulting in the very good basis for our attack, meaning that the membership inference confidence is quite high and the attack performs accordingly outside of a local-training context. According to Shokri et al[10], shadow data sets must be acquired from the original data distribution in order for the attack model to fully capture the target model's behaviour on data set, sampled from the very distribution, on samples of which, the target model has been trained on.

Nevertheless, Salem et al[9], argues that this assumption could be relaxed and that in particular cases of MIA, it is not even necessary in any degree.

Query Based Generation An additional way of effectively acquiring a ”Shadow’s Data set” is to utilize the target model API, combining it with a local search type of algorithm, generating synthetic data. This algorithm generated initially by Shokri et al[10] and discussed further in[5][9] [13] seems to be a respectful alternative over a white-box derived, statistical based generation method.

The basic idea behind this technique is that a randomly generated binary interpretation of a data point, can be transformed to a legitimate data point using the pattern recognition mechanism of the target model that is usually quite sensitive on the input feature distribution, because of the over-fitting on the training data set. Firstly, we generate a random binary vector and after that we progressively try to adjust the vector to fit the distribution space, with respect to each feature. At the same time, we define some score, thresholds considering the predicted label confidence and the adjustment mechanism details, that ensure we do not waste time and resources on trying to ”improve” a ”broken” data point.

Algorithm 1 Data Synthesis using the target model, Shokri et al[10]

Require: class : c

$\boldsymbol{x} \leftarrow RandRecord()$ ▷ Initialize Record Randomly
$y_c^* \leftarrow 0$
$j \leftarrow 0$
$k \leftarrow k_{max}$
for $iteration = 1 \dots iter_{max}$ **do**
  $\boldsymbol{y} \leftarrow f_{target}(\boldsymbol{x})$ ▷ Query the target model
  if $y_c \geq y_c^*$ **then** ▷ Accept the record
    if $y_c > conf_{min}$ and $c = argmax(\boldsymbol{y})$ **then**
      if $rand() < y_c$ **then** ▷ Sample
        return $\boldsymbol{x}$ ▷ Syntetic Data
      end if
    end if
    $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
    $\boldsymbol{y}_c^* \leftarrow y_c$
    $j \leftarrow 0$
  else
    $j \leftarrow j + 1$
    if $j > rej_{max}$ **then** ▷ Many consecutive rejects
      $k \leftarrow \max(k_{min}, \lceil \frac{k}{2} \rceil)$
      $j \leftarrow 0$
    end if
  end if
  $\boldsymbol{x} \leftarrow RandRecord(\boldsymbol{x}^*, k)$ ▷ Randomize k features
end for
return $NULL$ ▷ Failed to synthesize

Shokri et al[10], proposes Algorithm 1, a random generation based algorithm, that tries to converge to a data point that resembles samples from the target distribution. This algorithm penalizes many failed attempts and dismisses records that does not seem to ”improve” their synthesizing score over time. In addition, Shokri et al[10], integrates some hyper-parameters that control the quality of the data set and could be adjusted to satisfy the majority of attack cases.

Although the algorithm is successful in general, there are some downsides with respect to

the data quality and completion duration. This algorithm is generating new features totally at random, meaning that convergence is not promised and that the final data might be of poor quality. At the same time, data that cannot have a binary representation, like images, where features might be composed in a vector space and not in a smaller dimension space, are almost impossible to be synthesized by this algorithm given its binary nature. Any changes on the latter cause execution time and search space to explode exponentially and drop the algorithm's utility by a lot.

Active Learning Active learning[5][13], as a semi-supervised technique, tries to take a small complete data set, allegedly sampled from the same distribution as the target data set, and labelize a huge unlabeled data set. The task of label assignment is always a difficult one, meaning that the adversary might waste time and resources, whether he decides to use a more brute force way. At that point he can use a small representative labeled data set, derived by the larger one with a brute force technique and automate the task of label assignment. This technique might be combined with a black box statistic based generation, where we can generate a large number of input data, but with no knowledge on their respective labels.

Region Proximity Based Generation The last method we refer to is Region Based data point generation. In this method, we try to apply perturbation on a given data point x, labeled with class y and we ensure that $dist(x, x'), x' \in Perturbations(x)$ is under some predetermined threshold. This attack could be combined with some of our white box approaches, like the white box version of the statistical generation, or could occur before an active learning pass, in order to generate more "unlabeled" data beforehand. Note that all the perturbations that we generate must get the same label y as the original data point x.

3.2 Shadow Models

Throughout this section, we describe the reason that we need shadow models, the shadow training process and the impact they have to overall attack performance.

3.2.1 Shadow Models Promise

As we have already stated, a Membership Inference Attack will focus solely on the behaviour of the target model on specific data and not to the data itself. This is the real reason that a data set from the same distribution is required and why the attack succeeds even if the attacker's data set is disjoint to the target data set. The target model has a specific type (i.e. Logistic Regression, Support Vector Machine, NN, etc), is tuned in a certain way (adversary has no knowledge on tuning) and is trained on data from a particular distribution, that would effect the classification boundaries and the overall behaviour of the model on any given input x.

In the same spirit, we must note that the Shadow Models are just another way to sample our very own data set to train the final attack model. You can think a Shadow Model, as a simulation of the target model, that provide us with attack training samples and the ground truth on input's membership, in order to build a supervised model that infers membership

on any given input confidence vector. Following, we go through some definitions and mathematical notation.

We define a Shadow Model, $F_{shadow_i}, i \in \mathbb{N}$, which we train on a slice of the provided attacker data set, its Shadow Data set $D_{shadow_i} \subseteq D_a$. Think of each $D_{shadow_i} = \{\boldsymbol{x}, y\}_{i=1}^{N}$, where an adversary, could give $\boldsymbol{x}$ as input to the target model and receive a probability vector output, $\boldsymbol{p} = (p_1, p_2, \ldots, p_k)$, in a k-labels, classification problem.

The Shadow Models Bundle, promise that, whether $D_{shadow_i}, \forall i$ is similar to the target model's data set, distribution wise, they simulate the behavioural characteristics of target model, on "members" and "non-members", referring to the produced confidence scores. Following, we explain how the adversary can train the shadow models and what impact the shadow training has to the quality of attack model's training set.

3.2.2 Shadow Training

To start with, the adversary shuffles and divides D_a, to N different subsets, usually of equal size, where they might be overlapping[10][13]. Following this division, each $F_{shadow_i}(\cdot)$ is trained on its respective data set $D_{shadow_i}^{train}$ and its performance is evaluated on $D_{shadow_i}^{eval}$. Note that

$$\forall i D_{shadow_i} \cap D_t = \emptyset$$

The training process set up, is one of the most crucial decisions the adversary has to take, since in many cases training is heavily affect the shadow models' behaviour and the degree of generalization and over-fitting of the attack model. For example, hyper-parameter tuning, early stopping and learning rate decay usage must be decided before training starts and can be applied to all or some randomly chosen shadow models in order to sample from a larger attack hypothesis space. These little tricks increase utility of our attack, while at the same time, they also add complexity to it, increasing the need for a large sized D_a and more shadow models, than the conventional attack proposed by Shokri et al[10].

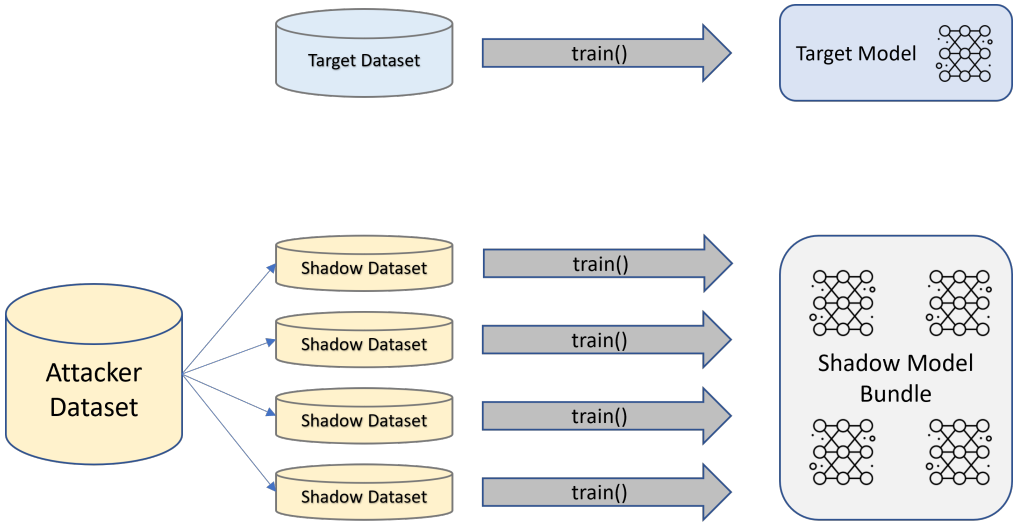

Figure 3.2: Target and Shadow Training[10]

As the figure above suggests, Shadow Models are usually of the same type as the target model[5][10][13] and in most cases of MLaaS, we use the very same trainable API-Service to produce the final Shadow Model Bundle. We will now continue with some theoretical notes and details about the impact of meta-tuning the shadow model bundle.

3.2.3 Shadow Models Impact

Firstly, we must consider the number of total Shadow Models in the bundle. As a general rule of thumb, Shokri et al[10] proposes that, the more shadow models one adversary trains, the better the resulting attack model is. This conclusion, derives from the fact that the more shadows' behavior, varies, the better the attack model, captures the pattern of this behavior in ”members” and ”non-members” input data points. Nevertheless, Salem et al[9], argues that, with only one shadow model, we can score similar attack performance to Shokri et al's attack. In this way, we relax the assumption that we need a large number of shadow models to acquire a high scoring attack and we might even solve the resource consumption problem, making the attack set up a little easier. Nevertheless, from our own experimental results on reportedly various data sets, we notice that the difference in the number of shadow models, plays a crucial role to overall attack success, concluding that, due to the fact that MIAs are data-driven attacks, the number of needed shadow models differs for each given data set. This is reasonable, whether the reader considers the fact that every different data set, from a different distribution, would require different handling in order to explain the meta-data hypothesis space, given the data distribution and a classification algorithm.

In the same spirit, we must consider the type of the shadow model, as a main factor of the attack's success. As we have already stated, Shokri et al[10] and others [5][13] suggest that the shadow models must be as much alike, to the target model, as possible. This assumption, satisfies the Shadow Model Promise and ensures that we achieve the highest possible scores, given that we have managed to explain the exact behavior of target model, on ”members” and not. Once again, researchers such as Salem et al[9], argue that this assumption could also be relaxed. They specifically propose a more stochastic method of choosing the shadow model type, meaning they incorporate an ensemble of shadow models and provide a total output confidence vector, aggregated in some way (i.e. concatenate confidence scores, take avg/max). However, Salem et al[9], indicates that for this ensemble alternative to work, the target model type should be one of the ensemble's supported types. Unfortunately, in a complete black box scenario, where we do not have access to a trainable model of the same type as target, it is highly unlikely we manage to successfully execute the attack without incorporating some element of guessing, although there is no assurance that this method works in general, or even in the majority of the complete black-box attack scenarios.

Finally, we could consider the necessity of a Shadow model bundle. As we have explained so far, the Shadow Bundle is the most accurate way of simulating the target model and execute a well informed and elaborated attack using a supervised machine learning attack framework[5][10][13]. On the other hand, the target model, might be over-fitted enough that the adversary would execute the attack with a simple threshold checking, given the confidence vector or the per input loss function value[5][9][15]. In this kind of membership inference attacks, there is no need for shadow models, or even for an attack model at all. Concluding, this metric based attacks are a good alternative in a total black box setting, but there is no reason to pick them over a supervised ML based attack.

3.2.4 Building the Attack Model's Data set

At this phase of the attack, we already possess a bundle of trained Shadow models and their respective training and evaluation data sets. We now shift focus on building a data

set that can be used to train a binary classifier that decides the membership status of $\boldsymbol{x} \in \mathsf{R}^{\mathsf{d}}$, given its confidence vector $\boldsymbol{p} = (p_1, \ldots, p_k)$, acquired by the target model.

We first start by constructing two sets, per shadow model, $D^{in}_{shadow_i}$ and $D^{out}_{shadow_i}$, where they contain "member"-labeled samples and "non-member" samples, respectively. Specifically, we can define the sets as it follows

$$D^{in}_{shadow_i} = \{\boldsymbol{p}^{(j)}, 1\}^n_{j=1}, \text{ such that } \boldsymbol{p}^{(j)} = F_{shadow_i}(\boldsymbol{x}^{(j)}),\ \boldsymbol{x}^{(j)} \in D^{train}_{shadow_i}$$

and

$$D^{out}_{shadow_i} = \{\boldsymbol{p}^{(j)}, 0\}^n_{j=1}, \text{ such that } \boldsymbol{p}^{(j)} = F_{shadow_i}(\boldsymbol{x}^{(j)}),\ \boldsymbol{x}^{(j)} \in D^{eval}_{shadow_i}$$

By concatenating and re-shuffling those slices, we create a final data set that can be utilized to train a binary MIA classifier.

3.3 Attack model

In the following few pages, we describe the set up and training procedure for the final attack model of a membership inference attack. We also raise some points, about the type of the model and the tuning based on the resources we have gathered.

Definition and Training Process At this stage of the attack, the adversary possesses a synthetic dataset similar to the target dataset and has created a bundle of trained shadow models, providing himself with a "ground-truth" enriched dataset of confidence vectors. Let us remind to the reader that, a MIA would learn to recognise and distinguise the patterns of target model behaviour in "members" and "non-members", that being the sole reason we use the shadow models output confidence vectors.

Shokri et al[10], argues that the best way is to create another model bundle, in which each class has its own MIA classifier. This means that we end up with k classifiers and for a given input x, labeled as y, we need to use the appropriate MIA classifier to infer the membership status. So let us define the attack model's dataset

$$D^c_g = \bigcup_i^N \{(\boldsymbol{p}^{(j)}, \mathbb{I}(\boldsymbol{x}^{(j)} \in \cup D^{in}_{shadow_i})) :\ y^{(j)} = c\}^n_{j=1}$$

and the attack model

$$F^c_g \colon [0,1]^k \to \mathbb{N}_2$$

Having those definition, the adversary can use a ML training scheme and derive a bundle of k models that he uses as described.

Aggregated Attack Model In our experiments we alter the architecture of the final attack model by incorporating the "true label" of input $\boldsymbol{x}$, as a model parameter and letting the selected ML algorithm interpret this feature as needed. Specifically, we define

$$F_g \colon \mathbb{N}_k \cup [0,1]^k \to \mathbb{N}_2$$

as our attack model and as we are about to see in our experimental work, the results are relatively, if not exactly, the same. We also have to redefine the attack model dataset as

$$D_g = \bigcup_{i}^{N} \{(y^{(j)}, \boldsymbol{p}^{(j)}, \mathbb{I}(\boldsymbol{x}^{(j)} \in \cup D^{in}_{shadow_i}))\}_{j=1}^{n},$$

meaning that the last term of each tuple is the membership status and the rest are the confidence vector concatenated with the true label.

Attack Model Impact Finally, we explain the attack model's effect on the overall attack performance. Most importantly, we would consider the attack model's type, because as we have repeatedly argued, MIAs are inherently data driven attacks, meaning that the model, which interprets meta-patterns in D_g the best, would be the preferred classifier, for the task. By the term meta-patterns, we refer to the fact that the original data space is transformed, with respect to the behavior of the shadow models.

In many cases, when the target and shadow models are NN classifiers, the adversary usually utilizes a shallow NN classifier as well [5][9][10][13]. On the other hand, when a statistical approach is used for making predictions, the adversary incorporates this kind of prediction models to both shadow and attack model architecture[5]. Finally, when the adversary attacks in a total black box scenario, with no knowledge on target model architecture, then the adversary could use an ensemble algorithm[5], or, more commonly, a simple threshold based approach, to build the attack model, utilizing any target model leaked metrics. It's important to understand, that these are just general guidelines and that there exist many variations[5] of Shadow-Attack model architectures, where the design process is always considering the target data set's characteristics and the behavior of each respective model type on it.

3.4 Conclusions & Motivations

Concluding our chapter on confidence vector membership inference attacks, we took a deep dive into the mechanism and the different attack phases. We have also explicitly addressed the characteristics that identify MIAs as data driven attacks and we have also pointed out the importance of acquiring a good data set on the attack end, along with some of the ways to acquire it, in order to realistically simulate the behaviour of target model. Following, we discussed, the matter of Shadow Training and the reasons of why is so important, if not crucial, to our attack. Finally, we focused on attack model and its data set's production, where we discussed the architecture and data formatting, respectively, as proposed by Shokri et al[10]. We also, defined our own alterations, referring to some of the above matters.

Nevertheless, the original membership inference attack, would, as we would see later, become less effective, whether we decrease the amount of output information, such as the number of labels and their confidence scores. The safest a model can be, is by providing the user with only the top-voted label and its confidence score. In the following chapter, we describe the effect of this situation on our original attack[10], while we are exploring an alteration of membership inference attack, called Label-Only MIA[2]. This attack exploits the same vulnerabilities as the original attack, but is more sensitive to the model's behavior on target data, achieving similar performance, at the same time.

4. LABEL ONLY MEMBERSHIP INFERENCE ATTACKS

Mitigating Confidence-Based MIA As we have seen so far, membership inference attacks aim to predict the membership status of a user, in a target model's dataset, based on the fact that a ML model predictions will score higher confidence, when the query sources from its training set. This behavior derives directly from overfitting of target model on its training set, where the model will learn training-specific data patterns, resulting to not generalizing well outside the training set, hence the difference in the confidence scores. A large amount of work has been submitted, in order to avoid these kinds of privacy leaks through algorithmic training of prediction models. The two basic solutions proposed by experts[2][9][10]; are

- **overfitting reduction** methods, such as regularization, dropout layers, early stopping, weight decay or simply increase the variety and size of training data, aim to fight the problem to its core and reduce overfitting[2][5][10]
- **confidence masking**, aim to hide as much information as possible from the adversaries, minimizing data leakage by slicing or adding noise to the final output confidence vector[2][6].

At this chapter, we focus on surpassing the obstacles, confidence masking defenses introduce, since they are considered countermeasures, specific to Membership Inference Attacks. Specifically, we are turning our attention towards the work of Choo et al[2] and their Label Only MIA framework. This new version of MIA exploits the same vulnerability, over-fitting, but with a rather interesting approach. Having in mind that an over-fitted model makes robust predictions on augmentations of data points, seen during training, we investigating the workings and the reasons behind these attacks' success.

4.1 Attacker's Dataset

As with the original version of membership inference attack, we start by describing the attacker's dataset, but this time, we focus on adjustments made to pave the way for the Label-Only version of the attack.

With respect to the means of acquiring the attacker's dataset, we can use anyone of the various ways we proposed in Section 3.1. The Label Only attack builds on top of the original attack's requirements and add some extra adjustments to each phase. Respectively to how an adversary would use this dataset, Choo et al[2]; propose the use of data augmentations for each record, that we use on the Attack's Model Dataset building phase. Following, we describe how Choo et al; uses perturbations of the attacker dataset and what purpose does they serve.

4.2 Data Perturbations

One of the most popular ways of acquiring more training data during a ML model's training, is augmenting the training data set. These data augmentations of the original training set, aim to increase the data variety the model sees through training, hence decrease

training variance and over-fitting, so that the model generalizes better in foreign data. Data perturbations can be applied in many datatypes, with the preferred candidate being images, during a classification, recognition or identification task.

Images, in Machine Learning, are usually interpreted as tensors, meaning they are multi dimensional data. At the same time, images are the only data where the user can easily notice some relation between the features of the image (i.e. facial characteristics of a person's picture). That is the main reason, that perturbations of images, are easy to understand and even easier to design. Most famous perturbations of images are consisted of translations, rotations, mirroring, filter convolution, contrast adjusting, cutting, blurring, etc. We are interested only to the first 2 types of perturbations, as proposed by Choo et al[2], because they provide us with convenient tuning and have a more subtle impact to the prediction of the target model.

Perturbations of a multidimensional data point, could be interpreted as "navigating" the vector space of the data. Specifically, more subtle perturbations aim to derive a new data point "relatively close" to the original point (according to some metric), while more aggressive perturbations, aim to derive a long distanced datapoint, with relation to the original vector. Rotations and translations are considered "subtle" perturbations, since a rotation/translations of couple of pixels would not ruin the image's feature correlation. On the other hand, filter application or aggressive, non supervised mirroring, would be considered "aggressive" perturbation techniques (i.e. object identified as "snow" being orange after filter convolution). So the reader might be having the question of how would perturbation help us to infer the membership status of a target datapoint, given a target model? The answer is really simple, but quite creative at the same time.

Imagine an adversary, who executes a confidence vector membership inference attack, but with only the top label, along with an optional confidence score, returned to him by the target model. This adversary, as we show at the Evaluation section, will get poor performance, which is a reasonable effect of minimizing privacy leaks through confidence masking. Nevertheless, our adversary, convinced that the target model suffers from over-fitting, notices that an overfitted model, would make more robust predictions, when input belongs to its training set. This observation is the key idea of the Label Only Framework of MIA. The fact that overfitted models, are less, prediction-wise, sensitive to changes/perturbations on their training data, could enable the attacker to infer the membership status of the target dataset, without the need of any confidence vectors at all. In a total black-box scenario this would mean, that the target model would only return a hard label, with no confidence score coming along with it. As we are about to see, this situation does not pose a serious threat for the attack's performance.

So now that we made our case about data augmentations, we describe the framework of using them along with some notation to set the foundations of Label Only MIA. Given a data point $(\boldsymbol{x}, y)$, we create the augmentation set $\{\boldsymbol{x}_1, \dots \boldsymbol{x}_N\}$. Every $\boldsymbol{x}_i$, is an augmentation of the original data point. The augmentation we use in our experiments[2] are image rotations withing 15 degrees of the original image orientation. More specific, we use rotations of magnitude $r \in [1, 15]$, generating $N \in \{1, 3, 5\}$ augmentations, in total, by rotating the image r degrees, left and right. Furthermore, Choo et al; proposes translations of the original image i, j pixels, horizontally or vertically, respectively, generating $N = 4d + 1$ augmentation, translated by, totally, $d = |i| + |j|$. During the evaluation section we also try to explore the effect the augmentation budget has in the Label Only attack context.

4.3 Shadow Models

As we established earlier, Choo et al[2]; sustains the same assumptions and attack architecture as Shokri et al[10]; with the sole difference that the target and shadow models return only a label. With respect to this change, we describe how we should adjust shadow models respectively, to fit the mechanics of the new attack.

4.3.1 Shadow Model Adjustments

Shadow models, following the assumption of using the target model as archetype, now return only the top label of each predicted result. The functionality, as well as the reason we are still using shadow models have not changed; shadow models still are imitating the target model's behavior and they still provide as with a basis of building a data-set to train the inference classifier model[2][10].

In the same spirit, we must notice that no shadow models use any of the augmented data, during their training phase. This refinement ensures that the Shadow training is not impaired by learning any more than what the target data-set's distribution allows the target model to learn. The reader should not forget that using augmentations in training, essentially normalizing the whole procedure, is yet another method that aims reduce over-fitting, but there is a catch. Whether the data-set continuous to fail the generalization diversity standards and the machine learning system has not escaped its previous low data regime, the model still over-fits and the label only MIA continues to score satisfying results (see Evaluation Section).

Having said that, we are ready to set up the Attack Model's data-set, using a similar method to the confidence based attack, with one small addition.

4.3.2 Building the Attack Model's data-set

As we have previously stated, augmentations of a record $\boldsymbol{x}$ are a set,

$$Aug(\boldsymbol{x}^{(k)}) = \{\boldsymbol{x}_1^{(k)}, \ldots, \boldsymbol{x}_N^{(k)}\},$$

of N augmentations. Let $(y_1, \ldots, y_N)^{(k)}, y_i = y^{(k)}$, be the set of labels we assign to all those augmentations. It is trivial to understand that all y_i labels are all equal to the original record's label, $y^{(k)}$, since they are all referring to perturbations of the same original record.

Our next step is to query all shadow models on their D_{shadow_j} and acquire the predicted labels for all perturbations. We use these predictions to form a new vector,

$$(Shadow_j(\boldsymbol{x}_1^{(k)}), \ldots, Shadow_j(\boldsymbol{x}_N^{(k)})) = (\hat{y_1}, \ldots, \hat{y_N})_j^{(k)}$$

representing the behavior of the model on all augmentations of a specific record $\boldsymbol{x}^{(k)} \in D_{shadow_j}$, meaning that it gives away details about how sensitive the model is to the specific input query.

To create the respective D_g's record , we have to create the boolean vector

$$\boldsymbol{b}^{(j,k)} = (\mathbb{I}(\hat{y}_1 = y_1), \ldots, \mathbb{I}(\hat{y}_N = y_N))_j^{(k)}.$$

We should note that, for each prediction on a $\boldsymbol{x}^{(k)} \in D_{shadow_j}$, the respective $b_i^{(j,k)}$ resembles on a picture of over-fitting for all "members". This picture might differ along the

different target models. For example using this paradigm on a trivial ”hard-memorizing” ML model, which remembers every input, would result to $b_{i\neq 0}^{(j,k)} = false$, where the original query's b-value would evaluate to true, for all ”members”, due to prediction by sere memory. On the other hand, non members would produce a vector of zeros. In this specific case, determining ”membership” status would be trivial. Usually, the robustness that [2]; refers to, is concerned with statistical models that classify input based on soft boundaries, enabling close-to-original perturbation to easily be identified as of the same class. Note that records in D_g interpret a normalized and vectorized form of a sensitivity measure, based on locality on the vector space.

Concatenating the true label and the predicted label with this boolean vector gives us the final inference record, on which we train our attack model.

At the end of this process we acquire the following data set

$$D_g = \bigcup_j \{(y^{(k)}, \hat{y}^{(k)}, \boldsymbol{b}^{(j,k)}) \colon (\boldsymbol{x}^{(k)}, y^{(k)}) \in D_{shadow_j}\},$$

on which we can now attack our inference classifier.

As we can see, the new attack data-set format, has captured the whole information the attacker wants the classifier to learn in a simple vector. More specific, we pass the true label, interpreting the inference classifier choice, the predicted label, a seemingly concise addition in order to inform the classifier for wrong target predictions and the boolean perturbation vector, which interprets the behavior of the model on the i-th perturbation of the target record. This boolean record could interchanged with the actual predicted label instead of the output of the true label equality condition check. Nevertheless, a boolean vector is way more subtle and normalized view of the information we are trying to interpret.

Now that we defined, the way we derive and the interpretation of records in our attack data-set, we are ready to discuss the refinements of the respective attack model, along with the impact it has on the attack.

4.4 Attack Model

Regarding the attack model's architecture, we are using an inference model, similar to the confidence based version of MIA, since the attack classifier is either a simple binary classifier, based on a statistical model, or a shallow NN, $F_g(\cdot)$, which we train on the attack dataset, D_g and use it to predict inference of the given member[2][10]. The definition of this model does not change for our attacks and although Choo et al, utilizes Shokri et al[10] type of attack model (one attack model per class), we move forward using our own version of the model, which, as we may observe later, would not impair the results of the Label Only attack.

As it is normal to assume, using a different kind of model, fitting for the inference problem at hand, is still quite important and, although we might fail to score as high as the original attack, in some cases a different model, than the best fitting model for a confidence based attack, would pose the perfect candidate for a Label Only MIA attack model. Nevertheless, to keep the comparison between the attacks as simple and intuitive as it can be, we use the same models in our confidence based MIA vs label only MIA studies (section ”Attack Evaluation”).

4.5 Label Only MIA on Categorical Data

In the previous section about Data Perturbations (section 4.3), we described the data types that we could apply intuitive and meaningful data augmentations. In this section, we go beyond that and understand why unconventional and non intuitive data perturbation in tabular categorical data, would allow us to generalize the Label Only framework to more than just image data sets. We start with our assumptions and we propose a method of generating random perturbations that actually allow us to integrate the Label Only Attack, in another data-set class.

To begin with, assume without loss of generality, that our categorical data are purely binary. This data set, $D \in \mathbb{R}^{N \times D}$ has been sampled from a distribution that determines a D-dimensional sample space. Training a model on this sample space, create a models that classifies queries based on binary vectors. On the other hand, the attack model, following the attack pipeline we described, would infer membership status of a given victim, solely on the meta-data acquired by the target's prediction confidence vector. Leaving the adversary out of the whole confidence vector is, seemingly, defending the model and hiding any vulnerabilities. Note that the attack model is being trained based on "prediction" data, meaning we do not care about the actual relationship of the features, but only the behavior of the model on them. This, gave us the idea of incorporating a new way of executing a label only attack that use the following data augmentation schema, during the Attack Data set acquisition phase.

Naive Binary Perturbator Let $P(\cdot)$, be the perturbation function, that accepts a data point $\boldsymbol{x} \in \{0,1\}^D$ and outputs an augmentation set $Augmentations(\boldsymbol{x})$. We propose a naive version of this binary augmentation mechanism, in which we use a number of random perturbations of the binary vector $\boldsymbol{x}$, that satisfy a distance threshold. We also define the distance between original and perturbation points, $dist(\boldsymbol{x}, \boldsymbol{x}_p)$, as the number of bits that differ, with respect to their place in the binary vector, also known as hamming distance.

How it works This perturbation mechanism, produces an predefined number of perturbations of the original data point, by utilizing different combinations of the binary features, with no regard to the relationship between any k given features. This very behavior is the reason behind its naive nature, since random selection of features, might meet the end-goal of its creation, but it does not maximize the attack's performance as much as it actually could. Let us think, a normal ML model creation process. We start at data preprocessing, feature extraction, model training, model evaluation and model selection at the very end. The phase of feature extraction, is a key stage in the whole process, since it defines the overall performance threshold in any case. In our case, we could think of the feature extraction process as the random selection of $\boldsymbol{x}$'s features to mix up. So, given the non determinism, why would this mechanism work?

Why it works As we have already stated, MIAs are data driven, or meta-data driven attacks, since their sole focus is the target model's behavior on the target data set. This very fact, in combination to the provided perturbations, enables the Label Only framework to exploit, any target vulnerabilities, since the perturbations would introduce totally foreign and maybe even non legit, queries that would stress the model and expose its

over-fitted state, does it exists. So we could realize that, respectively to the naive perturbator mechanics, one could propose a mechanism to maximize the explained variance of the meta-data, meaning the sample behavior of victim model on target data set, and in this manner generate the minimum number of perturbations to achieve the best possible attack performance.

The need of tuning Our proposed perturbator, utilizes some hyper-parameters, used to enhance and control the non determinism, but it also provides a threshold framework that does not overload our attack model with irrelevant and redundant features. To start with, we provide a max_n perturbation threshold, that reduces the number of generated perturbations up to a maximum. This specific parameter accommodates as a space threshold since our data points count would now be a multiple of max_n, meaning a great increase on the need for memory. Following, we provide a max distance threshold, that tunes the number of features we want to change in its generated augmentation of a data point. This threshold is related to the distance definition, we mentioned before and interprets a ”sphere” of data points, from which we get to choose to ”replace” our original point. Finally, we provide the user with a sampling choice. This particular hyper parameter, enhances the randomness of our perturbator and one can use it to study the difference between choosing different feature sets, on which we base our augmentation sets.

As an additional note, it would be interesting to estimate the total number of perturbations, our naive perturbator would produce, in order to elaborate on the aforementioned threshold, relative to a single point's total augmentation set size. Let n be the number of binary features and $max\_dist = d$. We now have to produce all the perturbation for distance, $dist \in \{1, \ldots, n\}$ meaning we would have

$$|P(\boldsymbol{x})| = \prod_{i=0}^{d} \binom{n}{i}$$

perturbations in total. For example for a vector $\boldsymbol{x} \in \{0, 1\}^2 0$ and max distance of 2, we would ultimately get 3800 augmentations of the original data point; too many features for any simple attack model.

4.6 Conclusions

In this chapter, we focused on an obfuscated version of the default confidence vector attack and, more specifically, in the Label Only[2] version of the attack. Choo et al[2]; constructed an attack framework, which can, not only overcome obfuscations of confidence score information, but also, as we see in the next chapters, break many popular defences that aim to mitigate confidence based MIA by ”masking” or hiding the output confidence scores. Following the road-map to this MIA version, we set the motivation for the Label Only Attack and describe how this attack works. One of the most crucial components of the attack was the augmented query data-set we used as input to the model. We essentially created a framework, where each original input query is interpreted by a binary vector. Each element of the vector would represent the fact that perturbation's predicted label being equal to the original predicted label. Respectively to this modification, we established a new attack model definition, with similar architecture to the one on the confidence based attack, but with a different interpretation to solve the task at hand. In the same spirit, we argued on the fact that this attack would also succeed in data sets of

different data types, relaxing Choo et al's[2]; assumption, where they focused on applying meaningful perturbations on the input queries, such as image translations and rotations. At this point, we also proposed an augmentation framework, for data sets with categorical features. Although we stated that the latter is fairly naive in its perturbation choices, we elaborated why such a framework would perform satisfactorily, given that proper tuning would be provided. At the end, we also proposed some motivations and thinking paths to improve this unorthodox perturbation framework, by maximizing the explained variance of the meta-data, our attack model is trying to explain.

So far, we have been arguing that the Label Only attack is an enhanced, more powerful, version of the original Membership Inference Attack. This reputation, is based on the fact that the most famous defences against confidence based Membership Inference Attacks, namely confidence "masking" defences, such as Memguard[6], or any other output obfuscation method, cannot protect user privacy against Label Only Attacks and the reason is quite intuitive. Confidence masking defences, use noise injection or vector slicing to hide information about target model's behavior on target data-set, but, in order to retain utility, they always return the true predicted label[2], meaning that the attack framework could be mitigated to a Label Only Attack, with the respective assumptions and constraints. This fact means that, although masking defences might confuse the original confidence based MIA, Label Only attack's performance, would not be impaired at all. Having established that, how could we reduce the effect of label only attacks? Furthermore, given we achieve our goal, how are we sure that there are no effect on the outliers of the data-set, which models are known to learn by heart, essentially "over-fitting" to them; how are we protected from this outlier attack? As an attempt to answer all these questions, we explore the most popular defences and frameworks that ensure basic privacy promises, with respect to the target ML models.

5. ATTACK EVALUATION

5.1 Experimental Set up

For most of our experiments, we utilize identical or similar in most ways, neural networks from the work of [2][10]. These NNs are usually simple and fast to train, while they serve as a reference point to previous work[2][5][10]. With our focus turned on target models, we try to keep the architecture and training process the same, especially in the image-based experiments, where CNNs are utilized. Furthermore, the attack models, besides the subtle changes we proposed in chapter 1, are the same shallow NNs that Shokri et al[10] used in their original work.

On the other hand, all the statistical models, used with tabular data sets like Adult, Purchase, etc, are hand-picked from a pool of available sk-learn models, following the set-up, that previous work established[5][10]. In most of the cases, we do not diverge from the previously tested ML architectures, while there are cases, where we slightly change the victims' and attack's settings.

5.2 Metrics Used

With respect to the metrics used to evaluate our attacks' success, we use the proposed [2][5][10] precision, recall, as well as the AUC score. Furthermore we define one metric for measuring target models' vulnerability to a MIA attack, which be expressed in a unit of its own.

Let $Acc(Model(D))$, be the accuracy of model $Model(\cdot)$ on dataset D. In the same spirit, let $Loss(Model(D))$, be the loss value of $Model(\cdot)$ with input D, given an arbitrary loss function. We define the following quantities

$$A = \frac{Acc(Model(D_{target}))}{Acc(Model(D_{attacker}))}$$

and

$$L = \frac{Loss(Model(D_{attacker}))}{Loss(Model(D_{target}))}$$

assuming that $D_{target} \cap D_{attacker} = \emptyset$, since we need a data sets of both members and non-members to estimate the victim's vulnerability degree.

Given A and L we can define the following quantity as our unitary vulnerability metric

$$Vuln(Model, D_{attacker}, D_{target}) = \log 2 \times \frac{A \times L}{A + L}.$$

We can notice that due to the fact that, $A \geq 1$, $L \geq 1$, we get $Vuln(\cdot) \geq 0$. In theory, the greater this quantity is, the more vulnerable the model is to our MIA attack, with $Vuln(\cdot) = 0$, meaning that the model is theoretically, bulletproof to a MIA, while the actual model vulnerability can increase to ∞, without any constraint.

5.3 Datasets

The following sections are dedicated to introduce the datasets we used to evaluate our experiments.

CIFAR-10 is a benchmark dataset, used during evaluation process of image recognition and classification algorithms[1]. It is composed of $50,000$ training and $10,000$ evaluation images, labeled with 10 classes in total. Every class has $6,000$ images. Cifar-10's images are 32×32 RGB images and are ideal for our experiments on how MIAs affect image classification algorithms. We come back to this dataset quite often among our studies results displaying. For our experiments we use different slices of Cifar-10 dataset to show the attack performance, without the effect of any bias across those slices.

MNIST dataset[2] is another benchmark dataset, where each 32×32 image, represents a handwritten, centered digit. There are $70,000$ images, of which $60,000$ of them are part of the training slice, while the rest consist the evaluation set. We also, use slices of the dataset to study the effect of MIAs on MNIST dataset.

UCI Adult (Census Income) datset[3] is consisted of $48,842$, records of 14 features each. Features contain information about each record's age, education, gender, marital status, occupation, etc. This data set is used to train a binary classifier, that predicts if a person makes more that 50K a year or not. We use different randomly sampled slices of those data.

Purchase K data set[4], is similarly acquired to Shokri's et al[10], Purchase dataset, with the sole difference that Kaggle's "Acquire Values Shoppers" Competition[5] has changed the dataset format a bit. The final dataset, is consisted of binary records, where each feature is assigned 1 if the customer has acquired the described product or 0 if not. To label this dataset, we use a K-Means algorithm in order to create $5, 10, 20, 50$ and 100 labels, that interpret a customer type. At the end, we use the derived data sets to train a new classifier that decides the customer's type and we study the effect of labels' amount to our MIA attacks. The slices we use for target model's training dataset, D_t, and attacker's dataset, D_a, are as balanced as possible, through random sampling.

5.4 Confidence Based Membership Inference Attack Evaluation

We start with the default, confidence based membership inference attack and display some of our experimental results, along with our notes and comments on the latter. In order to recreate our attack, we provide a python library, implementing most of the attacks and their alterations, which one could find in this GitHub repository.

[1] https://www.cs.toronto.edu/ kriz/cifar.html
[2] https://yann.lecun.com/exdb/mnist/
[3] https://www.kaggle.com/datasets/wenruliu/adult-income-dataset
[4] https://www.kaggle.com/datasets/vissarionmoutafis/purchase-datasets
[5] https://www.kaggle.com/c/acquire-valued-shoppers-challenge/data

5.4.1 Target Overfit Effect

There is a variety of ways a model could result to over-fit on its training set. In this study, we explore the effect of the target data-set size/diversity and the number of training epochs to the attack performance. The point of this study is to elaborate on the fact that over-fitting is a sufficient factor for a successful and accurate attack.

In both runs, we use the CIFAR-10 data-set and a CNN model as target model. The latter has 2 filter layers of 32 and 64 filters, respectively, pipelined into 1 max pooling layers each and a tanh activation between each layer. At the end of the CNN, we use a dense layer with 128 nodes, connected to an output layer of c nodes, where c is the number of the classes on the data-set.

Overfitting: Extended Training While studying the number of training epochs effect, we use an early stopping callback, during first training to acquire the best model. After that, we train the target model for 2 epochs and then attack the model. Our adversaries have in their possession 20 shadow models and 500 data-points for each one of them. Comparing them with the initial 50000 data-points of the target training data-set, it is really interesting to observe whether the attack succeeds, given the extreme gap between the training set's size and the adversaries' data-set. Note that Shokri et al[10] argues that the dataset size, especially in the case of CIFAR-10, is indicative of the diversity the target model has witnessed, meaning that we don't expect high performance. Nevertheless, we expect to get accuracy over the baseline gap-attack model[15].

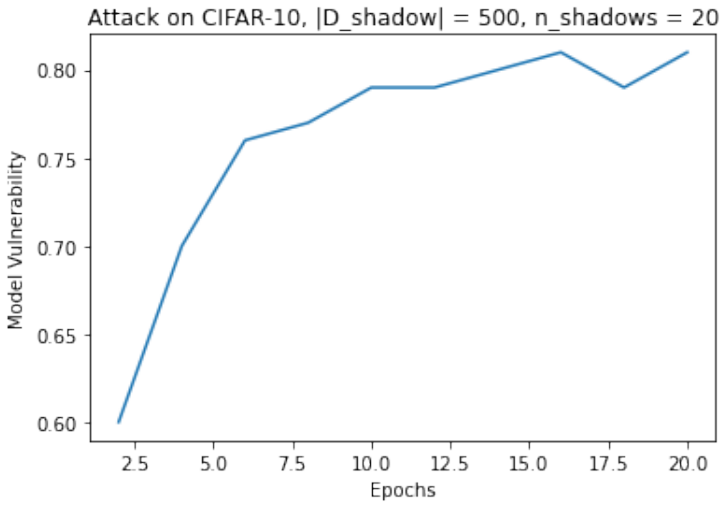

Figure 5.1: Target Model vulnerability metric with respect to epochs of target model training. Our vulnerability metric quantifies the degree of over-fit in the target model. We observe that the more we train the model the more it over-fits into its training data set

As we could see in Figure 5.2, the precision score hits a relatively good high-score of almost 0.76. This maxima, indicates that the model was the most vulnerable at the 16-th epoch, while before and after that point the scores seem to decline. It is valid to assume that, at the left of the maxima, before 16 epochs of training, the model was increasingly becoming more vulnerable, meaning that it over-fitted more on its training set.

Following, as Figure 5.3 indicates, we notice that the recall of the model behaves similarly to the precision score, while it displays a smaller gap between its minima and maxima. This fact drives us to the conclusion, that the attack model is biased towards the ”member”

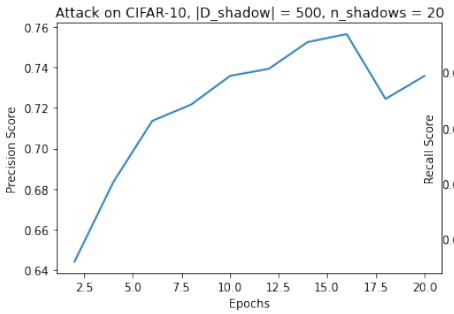

Figure 5.2: Attack's Precision

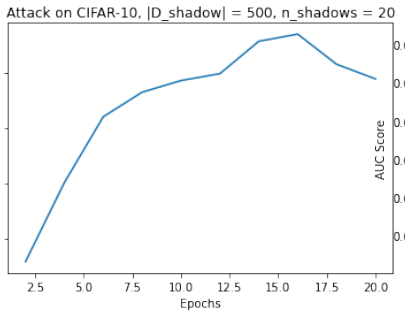

Figure 5.3: Attack's Recall

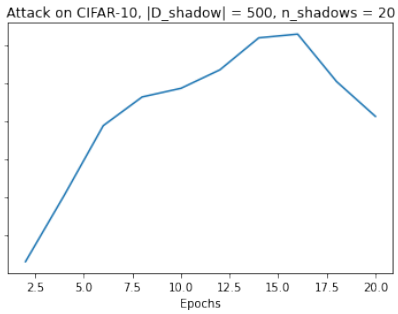

Figure 5.4: Attack's AUC Score

labels, resulting into high precision scores and relatively low recall. In other words the model prefers to classify input queries as "members", indicating a low accuracy attack classifier. Nevertheless, note how the attack model scores over the baseline and how the AUC score in Figure 5.4 is resembling the very same behavior as attack's recall.

In order to understand the vulnerability degree of our model, we could observe the vulnerability metric we proposed earlier in this chapter (Figure 5.1). The figure above displays a constant and, most of the time, monotonous increase in the model vulnerability metric, throughout the training, which might trouble us, referring to the reason that our attack fails to perform better, as the training proceeds. The reader might have noticed that, our main concern from the beginning of this experiment was the large gap between the attacker knowledge of the data-set and the knowledge that the target model acquires during training, no matter how simplistic its structure is.

All of the above indicators, drives us to believe that there is a strong connection between the target and attacker data-set, size, diversity rate and explainability of the input data space. In the following study, we try to explain the target data-set relation to the performance of the attack, from the scope of over-fitting, model vulnerability and attack sensitivity.

Overfitting: Training Data-set Size To start with our study on the effect of the training set size, we must firstly state that, there is no promise the target model is not over-fitted further as we increase the training data set size. For example, whether we decide to add n thousand different perturbations of the same image to its training data set, we would harm the model, privacy-wise, since, although we increased the data-set size, the final model is extremely good at predicting different alterations of the image we augmented, while it does not be respectively confident on predicting other images, or even images in other classes. Having said that, we should try to explain the results of this study with a careful mind, knowing that there are more behind the success of this attack, that the over-fit degree of the attack model.

Once again, we see that the precision of the attack, in Figure 5.5, hits a very high maxima, over 0.8 units, meaning that our attack can identify most of the "members" and classify them correctly. This, seemingly expected, result, could only be explained, whether we assumed that the target data set size, is highly related to the attack precision. From the rest of the precision graph we can see that for training sizes under 10000 records, the attack has high precision, while there is a dramatic performance decrease when attacking a target trained on a data set of 10000 records and a linear decrease as we double the last target data set size. This behavior points out how important the target data set size is, in order for MIAs to succeed.

However, in rest of the figures, we notice a rather odd behavior. In Figure 5.6, the recall of

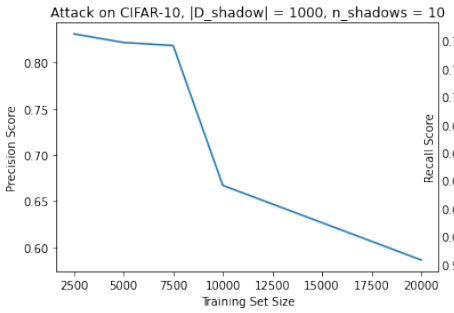

Figure 5.5: Attack's Precision

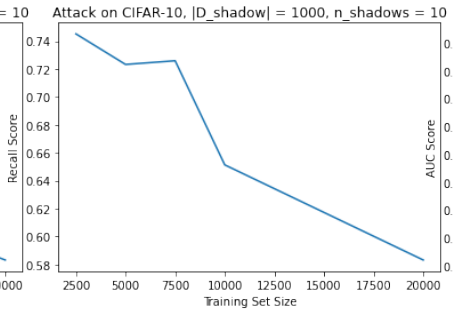

Figure 5.6: Attack's Recall

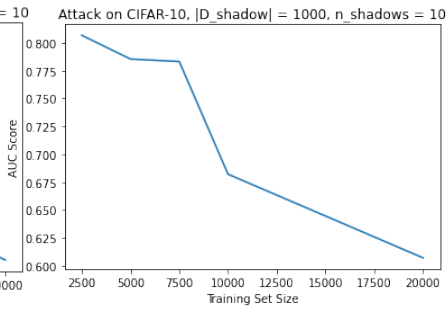

Figure 5.7: Attack's AUC Score

the attack, starts at a global maxima of almost 0.75 and then decreases, respective to the attack to a target model trained on 5000 data points, but then we notice that it increases again to a local maxima. After that the recall measurements follow the same behavior as the precision with a dramatic fall and a relatively linear decrease. The strange behavior we have noticed, given it is relatively unremarkable to the rest of the plot, might have been the outcome of a lucky guess from the attack model. We investigate this behavior more, in the following sections.

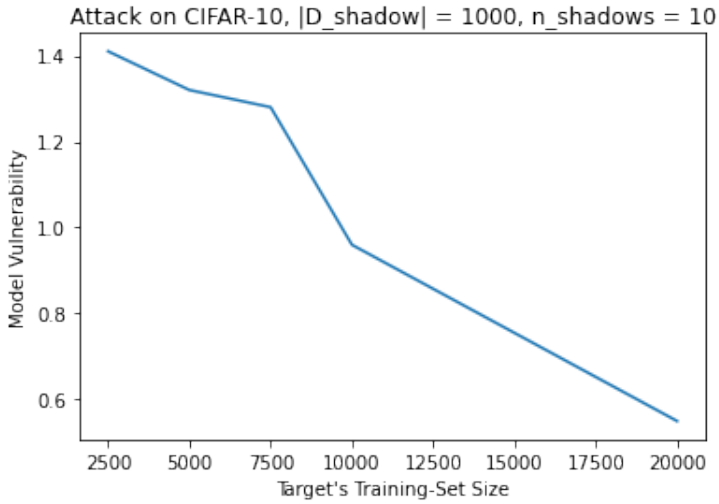

Figure 5.8: Target Model vulnerability metric with respect to target's training set size. As we expect, a larger more diverse data set enhances the target's generalization power and decrease the degree of over-fitting, hence decreasing the value of the target vulnerability metric.

In the last figure (Figure 5.8), we display the plot for the model vulnerability metric, acquired for models, trained on different data set sizes. The vulnerability metric line-plot follows similar behavior to the rest of the graphs, starting at an unseen high of 1.4 units, meaning that the model is very vulnerable to membership inference attacks, while resulting to a score of 0.6 units, on attacking a target model, trained on 20000 data points. Combining our notes from all four graphs, we can see that the last attack, although surpasses the base gap-attack's performance, do not satisfy one's expectations, since even the precision score is relatively low (under 0.6 units).

To conclude our study on the relationship between target model over-fitting and attack performance, we could notice that MIAs are inherently biased to "members", hence the high precision scores in both studies, while the attacks' accuracy, recall and overall sensitivity on the input queries, witness some interesting results. In the first study, we could see that the more we try to over-fit on a given data set, the more vulnerable the model would

become; but there is a pitfall. Due to the remarkable differences on target and attacker data set, in both size and interpretation power of the data space, the shadow models fail to simulate the target model predictive behavior. This situation becomes more obvious in the second experiment, where we elaborate on model performance and target's vulnerability, with respect to the target's training set size. Our results indicate that the more similar target's training set is, to the shadow models', the better attack performance we yield. However, there is yet one another important factor that we have not considered, the shadow models number. In the following section, we investigate the overall effect of the shadow training on attack performance and we elaborate on whether it is important to the attack. Note that there are many researchers who argue this opinion[5][9][13][16], making us more interested in investigating the reasons behind shadow training being such a vital part of Shokri et al[10]; version of MIA.

5.4.2 Shadow Training Effect

In this study, we investigate the tuning of the attack, regarding the Shadow Training configurations and we evaluate our results based on precision, recall and AUC score of the attack. The model vulnerability metric, while useful to display in each case, does not concern this study, since we use only one model as a target.

With regard to the set up, we keep the same settings as in the previous experiments in order to have a common reference point, while interpreting our findings.

As Shokri et al[10] and others[5][13] have noted, the most important stage of the attack is the Shadow Models training. The tuning of this training is equally important, since it gradually leads the attack to better assumptions about the target behavior, hence builds up to a higher-performance MIA. The two main "hyper-parameters" we decide to look at, are the number of the shadow models we train during this phase and the size of each shadow model's data set, D_{shadow_i}. Note that we introduce a new graph type, which might be a little hard to interpret at the beginning, but we will lead you right through it.

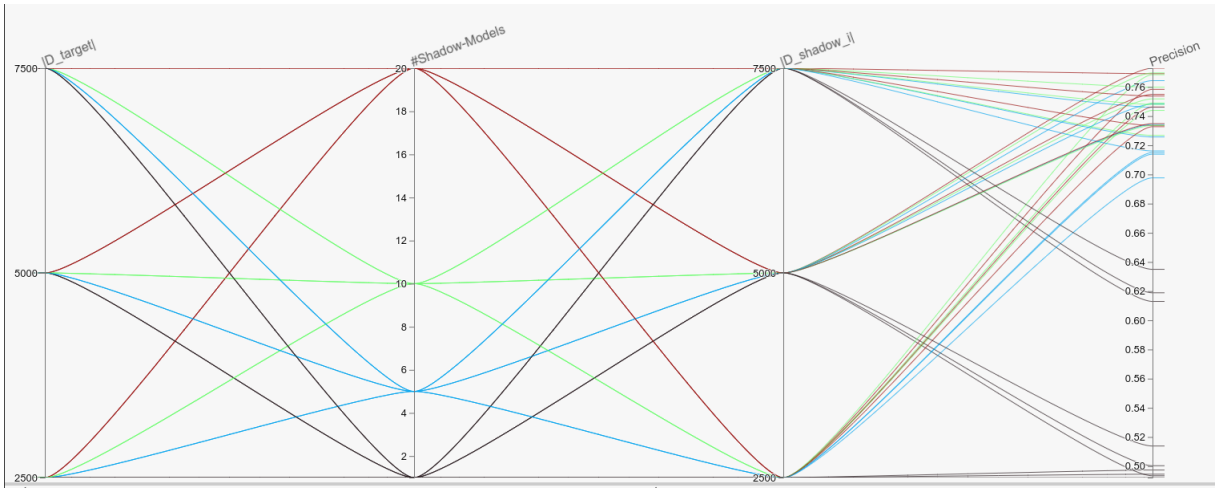

Figure 5.9: Attack's Precision

To start with, we explain how to interpret Figure 5.9, the precision plot. From left to right, the first column displays the target's training size, taking 3 distinct values $(2500, 5000, 7500)$. The following two columns refer to number of shadow models and $|D_{shadow_i}|$, respectively, while the last column, far right, displays the value of the AUC score metric. As we can

see, there is coloring, indicating relationship between different attack set-up settings and the final AUC Score we get.

To get into the specifics of the precision plot, we can firstly notice one main notion. The higher performance attacks, meaning the red, green and some blue lines, indicate that the more shadow models we train, the better performance we acquire, which is the expected behavior according to Shokri et al[10]. Furthermore, it is quite trivial to point out that when the attacker has more knowledge over the dataset, than the target, the MIA scores high in the precision metric. At the same time, as the target's training data set increases in record count, the precision score declines, with the number of shadow models, constraining the rate of decline by balancing the closing of the knowledge gap, with the knowledge over the target ML model.

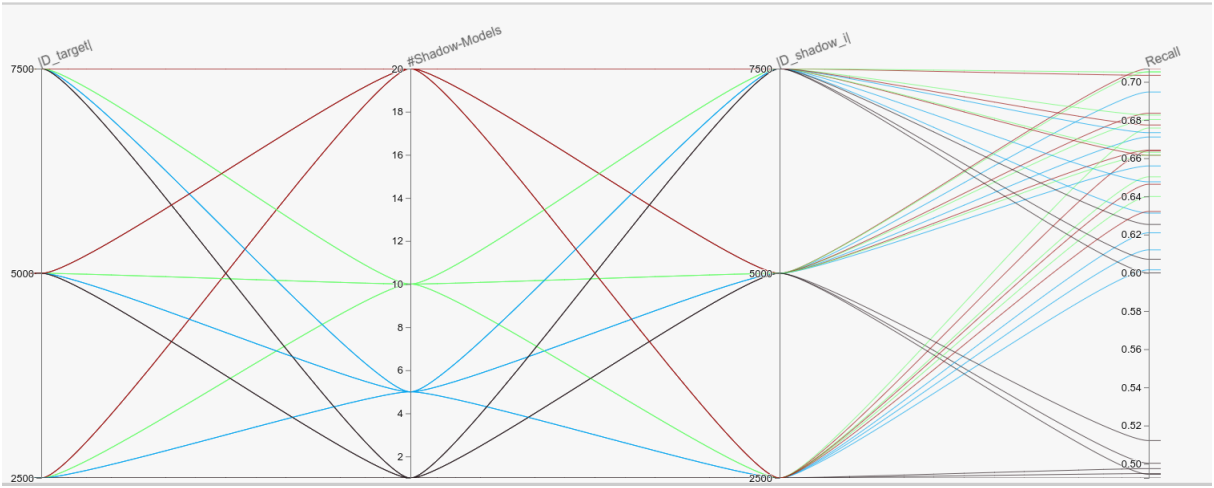

Figure 5.10: Attack's Recall

When it comes to recall (Figure 5.10), we expect from previous experiments to be remarkably lower than precision, but in this case we notice that, under appropriate training, the recall scores skyrocket next to precision. Nevertheless, recall plot, seem to ”spread out” more smoothly, in comparison to the precision scores that create clusters inside specific score ranges and the transition across the experiments are steep and non gradual. In the same spirit, we could observe that recall scores are indicating when an attack is poorly tuned(gray plot-lines), resulting in recall scores under the baseline threshold, 0.5. The latter, indicates a great knowledge gap between the target and the adversary, essentially, referring to the shadow models number and the shadow data set size. For instance, during a MIA with 1 shadow model, trained on 2500 data points, against a target model, trained on a data set of size 7500, the attack recall falls under the baseline, elaborating on the previously discussed knowledge gap of the attacker-target model pair.

The AUC Score plot (Figure 5.11), follows the notion of the recall graph, but manifests a sole difference; the AUC score is not falling under the baseline threshold. This indicator witnesses that, although the respective attack setting is poorly tuned, the target model is still vulnerable and the attack classifier recognizes it, without the power to yield membership information out of it.

As we have seen so far, the shadow training is an significant component of the general attack pipeline and more importantly, the number of the shadow models, n, we choose to utilize, essentially set a lower threshold for our attack performance. The attack's sensitivity to attacker data set quality and quantity is highly related to n as one could notice from

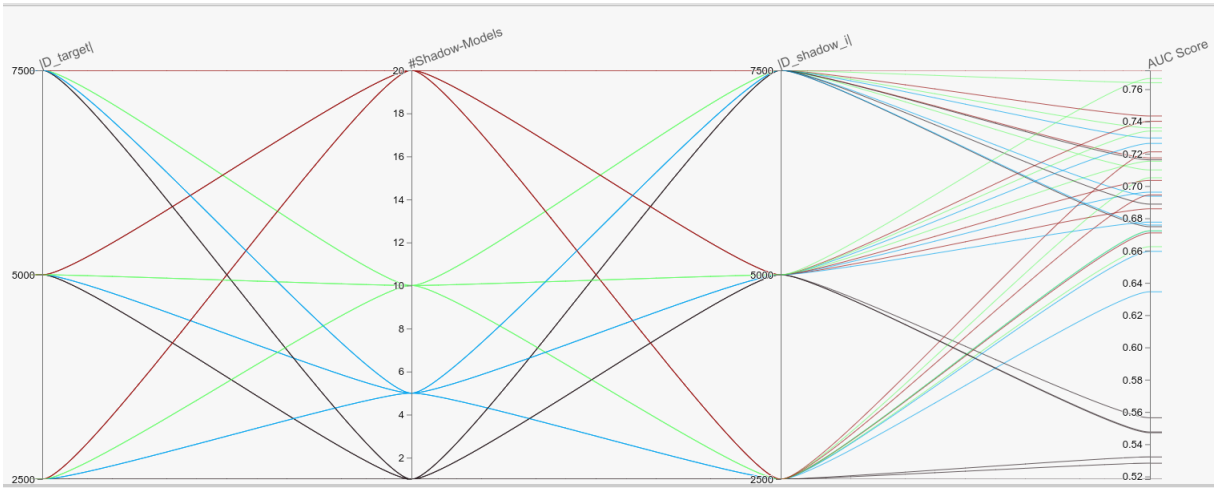

Figure 5.11: Attack's AUC Score

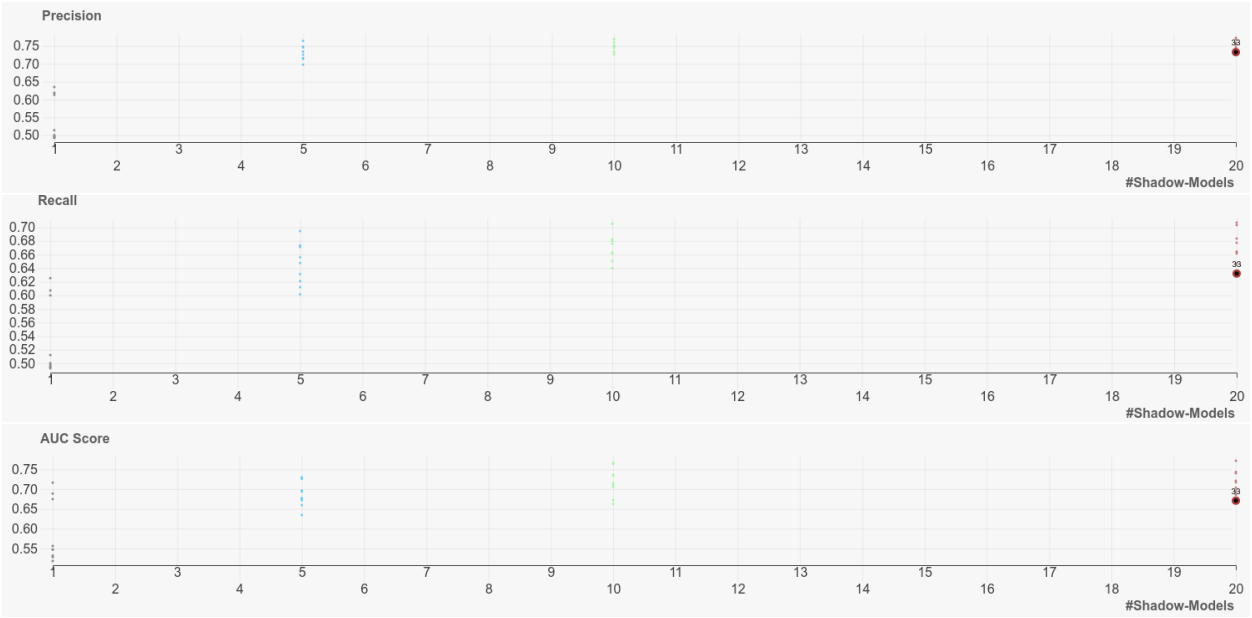

Figure 5.12: Attack's Precision, Recall & AUC Score with respect to the number of shadow models

Figure 5.12. We can see that the clusters of the attack scores are getting more concentrated, the more n increases, while at the extreme case of training 1 shadow model, the attack scores' variance is fairly high.

5.4.3 Classes of the Dataset Effect

Until now, we have explored the effect of target model vulnerability on the attack and the relation of MIA performance and Shadow Training's tuning. Now we are ready to change study domain and investigate how does the data space affect MIAs[5][10].

In this study, we focus in the effect the amount of labels assigned to the data set, have on the attacks performance. We utilize the Purchase-K group of data sets, which is constructed using unsupervised methods, making it a good candidate for such a study and we deploy NNs in the place of target, shadow and attack models. Next, we provide the reader with metric plots, respective to the number of data set classes and proceed to evaluate the attacks and explain our findings.

Before we start, note that the experimental set up for this study, along with the target-vs-attack settings, are the same as in Shokri et al[10] investigation.

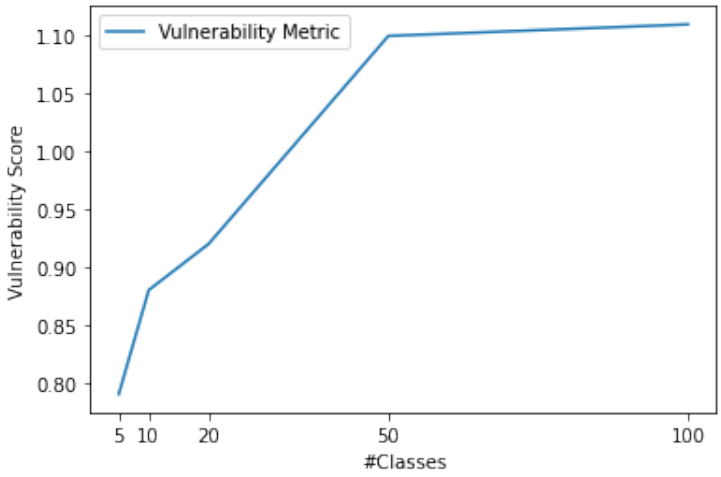

Figure 5.13: Target Model vulnerability metric for Purchase data sets. Note how the increase on the number of assigned classes, increases the target's vulnerability up to 1.1, suggesting a very vulnerable target model. As expected, the attack performance follows the same trend.

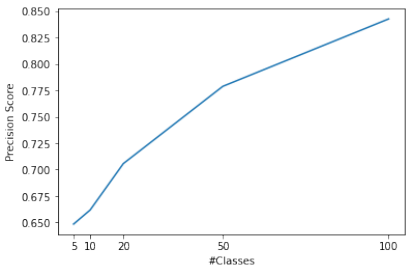

Figure 5.14: Attack's Precision

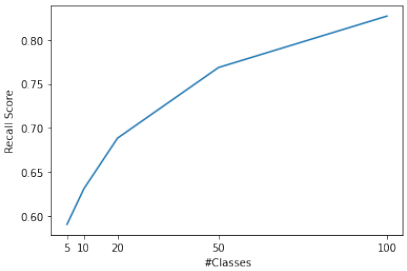

Figure 5.15: Attack's Recall

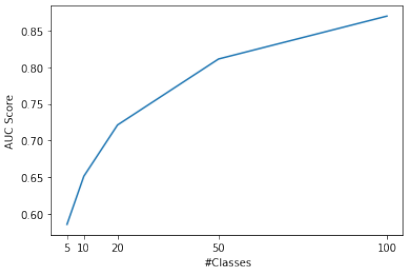

Figure 5.16: Attack's AUC Score

Starting with Figure 5.14, the precision scores seems to rise almost linearly to the number of classes, a result which could only derive from the fact that the more classes we divide our data to, the more information we leak about their characteristics and the target's behavior on them. It is safe to assume that by decreasing the amount of classes, there are more data, regarding quantity and versatility, to the data set, meaning the final target model generalizes better and the attack is harder to perform well.

We could notice similar results in the other 2 figures, referring to recall and AUC score of the MIA, with focus on the fact that recall reaches similar scores as the precision, indicator of high attack accuracy and good discrimination between the 2 membership status flags. But what is the reason the attack is succeeding in extensively divided data sets? To answer this question we must take a look at the model vulnerability metric report.

By studying Figure 5.13, we conclude that the model's vulnerability to a MIA attack is related to the number of classes in the data set. It is relatively trivial to understand that the more classes the data set is divided into, the less data points are assigned to each class, meaning the model has less examples to train on for identifying a given class. This situation leads to a decreased generalization degree, which is quite apparent from the vulnerability metric plot, which is specifically specifically manifested in the transition from assigning 10 classes, to assigning 20 and 50 classes to Purchase users' data points. This steep incline in the vulnerability plotting, indicates a high degree of over-fit, along with a

high advantage for the adversary, who achieves a high confidence membership inference attack.

5.4.4 Missing Features Effect

In the next experiment, we try to investigate the character of the relationship between target and attacker data quality, with respect to missing, or invalidated data features. For the purpose of this study, we use the ADULT data set, which includes both categorical and numerical features for each row. We utilize Random Forest Classifiers, for both target and attack models, while we design the shadow models to resemble the target model, as much as possible, in order to follow the proposed set up[5][10][13]. Finally, to interpret data loss, we simply replace some a percentage of features, with either the mean or the median of the column, based on common techniques against missing data. Following, we present the precision and recall boards of our experimental findings, along with some comments, regarding the attack behavior on different settings.

Let

- $|D_a|^-\%$: percentage of missing features from the attacker data set
- $|D_t|^-\%$: percentage of missing features from the target data set

Table 5.1: MIA Precision Scores on ADULT with missing features

$\|D_a\|^-\%$ over $\|D_t\|^-\%$	0%	5%	10%	15%	20%	25%	50%
0%	0.78	0.72	0.66	0.64	0.63	0.6	0.6
25%	0.78	0.7	0.66	0.63	0.62	0.6	0.6
50%	0.78	0.7	0.65	0.63	0.61	0.6	0.6
75%	0.78	0.7	0.66	0.64	0.62	0.61	0.59

Table 5.2: MIA Recall Scores on ADULT with missing features

$\|D_a\|^-\%$ over $\|D_t\|^-\%$	0%	5%	10%	15%	20%	25%	50%
0%	0.6	0.59	0.57	0.57	0.57	0.56	0.57
25%	0.6	0.59	0.57	0.57	0.56	0.56	0.57
50%	0.6	0.58	0.57	0.57	0.56	0.56	0.56
75%	0.6	0.58	0.57	0.57	0.56	0.56	0.56

By observing Table 5.1 and Table 5.2, we could derive the conclusion that no matter how much of the data are corrupted in the attack data set, the fact that, the model is still vulnerable to MIA, over-compensates the information gap. Furthermore, it is obvious that the higher the missing data percentage in the target data set gets, the less accurate our attack gets, since both recall and precision drop, with the latter declining 0.18 units from attacking a complete data set, to attacking a data set with a missing feature rate of 25%. The severe drop of precision, along with the similar recall score, would also indicate a more stable attack, with small variance between different settings, but with low scores, posing no important danger to the model (recall score of slightly over 50%).

The interesting bit of this study is the fact that, adding obscurity into the data set, by removing, for instance, features and nulling random values, does not guarantee that the model is safe against MIAs. Having said that, any data keeping vendors that decide to

make scrubbed user-data public, put their clients' privacy on a great risk since, adversaries could surpass the restrictions posed by the obstruction of private information and infer the membership of individual targets that, at first sight, might not appear to be part of the target data set.

5.5 Label Only Membership Inference Attack Evaluation

With regard to the original MIAs, Label Only attack can be compared to the confidence based attack on a plethora of image data sets like CIFAR-10, CIFAR-100, MNIST, etc[2][10]. In order to re-create and compare these two attacks, we can use our own frameworks, created solely for experimental purposes.

From out results, we acquired similar results to Choo et al[2], where the recall and precision of Label Only Attack, are quite close to the original attack results. We provide some statistics from our attacks as a reference point

- Attack on CIFAR-10, with 5 shadow models, $|D_{shadow_i}| = 2500$ and a target CNN model of similar architecture to Shokri et al[10] trained on 2500 data points;
  - **Confidence Vector MIA** scored precision of 0.85 and recall of 0.80
  - **Label Only MIA** score precision of 0.82 and recall of 0.79
- Attack on MNIST, with 5 shadow models and $|D_{shadow_i}| = 2500$ and a target CNN model trained on 2500 data points
  - **Confidence Vector MIA** scored precision of 0.53 and recall of 0.51
  - **Label Only MIA** scored precision of 0.51 and recall of 0.51

So as we can see, since label only attack builds on the foundations of the confidence based MIA, the first behaves quite similar to the latter, while being an improved version of the original MIA, due to the extra "pre-processing" phase to enrich the rather restricted attack data set.

We now proceed to a comparison between the confidence based and the Label Only attack, on the Purchase data sets, using the categorical perturbation engine we referred earlier, on the Label Only chapter.

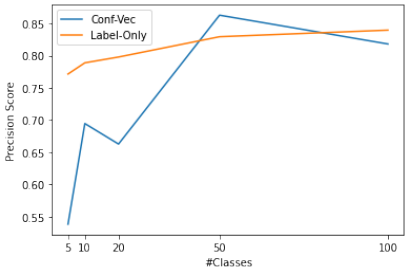

Figure 5.17: Attack's Precision

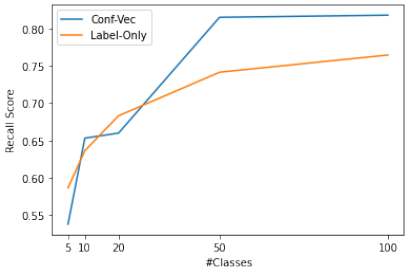

Figure 5.18: Attack's Recall

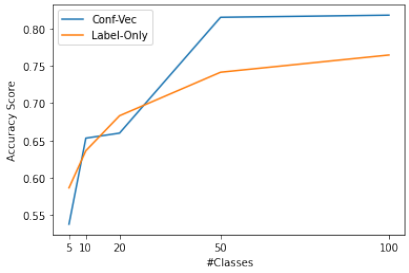

Figure 5.19: Attack's AUC Score

As the Figure 5.17 displays, the precision of the 2 attacks is progressing differently, as we increase the data set class assignment. For the confidence based attack, we notice that the progress is relatively steady and converges a little over 0.8 units of precision, while for the Label Only attack, the precision is constantly high, over 0.75 units, only to

be surpassed by the confidence vector precision score, while attacking the Purchase-50 data set. In the last data set, Purchase-100, we notice similar scores with label only to be slightly better. In order to get a better idea of the reasons behind this behavior, we proceed to the examination of the rest of the metrics.

Figures 5.18 and 5.19, are providing further insights, relative to the behavior of the Label Only attack. Essentially, we observe that, the 2 attacks seem to scale gradually and on a similar pace, with label only version being ahead for the first couple data sets, but then left far behind, regarding to performance scoring, from the original confidence based attack. Furthermore, the final performance gap between these two versions of MIA, is as large as 0.05 units, which might seem small, but in the context of the attack performance upper and lower limits, its fairly significant.

Combining all three plots, we can conclude the following

- confidence vector attacks are, often, more stable in behavior across all three metrics, meaning that the complete information about the output vector of probabilities, is of high importance no matter the data set.
- label only attacks can extend to more than image based data sets and perform as good as the respective confidence based attack
- between the two attacks, we should always go with the one that utilizes the most information on the attacker's end
- precision metric, is useful indicator of our model distinguishing the ”members” of the target data set, but it should not be the only performance metric in use, since it might not provide enough insights to evaluate the attack's power.

Summing up, we have explored the different settings on which a MIA might work and we have investigated the reasons that those attacks perform so well. We also, observed different versions and attack scenarios, while we applied our understanding of the attack mechanics to perform a distorted version of the Label Only Attack, which essentially, scored results, similar to the default attack, but it made clear to us, that the attack's success is a combination of knowledge of the data set features, the target model's tuning, or even the target's model-type. All these support the opinion that the over-fitting state of the target model is quite sufficient to perform a successful attack, but there are more factor to consider in order to properly defend our model from MIAs[2][5][7][10].

6. DEFENCES

So far, we have presented and described various versions of membership inference attacks. Now we turn our attention to the other side of the coin, defences against membership inference attacks. As mentioned in Shokri et al[10] and others[2][5][13], there are couple of mechanisms to effectively defend your model from the aforementioned privacy leaks, divided in two large categories

- over-fitting reducing defences, that were originally used out of the privacy context
- defences against specific inference attacks, that were designed to prevent this kind of attacks.

In the following sections, we document this defences and describe how they reduce the performance of the membership inference attacks. We start with the over-fitting reduction based defences, in order to elaborate further on why over-fitting is such a dangerous vulnerability, leading to privacy breaches. At the same time, we explore the argument that over-fitting is not a necessary assumption for a MIA to succeed. Later on, we turn our sight into defences against inference attacks and extend our research onto the most famous defences, confidence masking and differential privacy. Furthermore, we present couple of points on why, some of the defences, might fail in the face of the Label Only attack framework. As we have mentioned before, although many defences might work against the original MIA version, they might also lead to dangerous paths when the adversary uses a more advanced and sophisticated attack framework, like the Label Only one. Finally, we follow with a whole chapter, in which we present our experimental results, using couple of these defences.

6.1 Reducing Over-fitting

6.1.1 Training Phase Defenses

One the most effective and obvious stages of applying over-fitting preempting techniques could be the training of the model. During the model training phase, researchers use a variety of mechanisms to reduce the over-fitting[16] degree of the final selected model. Namely, one of these mechanisms is regularization, applied into the update rule of the models weights and used in both statistical models and Neural Networks. Other popular mechanisms are dropout nodes, usually seen in NN training routines and early stopping.

Regularization Let $f(\boldsymbol{x}; \boldsymbol{w})$ be our prediction algorithm and let $L(f, \boldsymbol{x}, \boldsymbol{w})$, be the loss function. In order to avoid over-fitting and try to penalize the weights responsible of causing it, we define a new loss function,

$$L_R(f, \boldsymbol{x}, \boldsymbol{w}) = L(f, \boldsymbol{x}, \boldsymbol{w}) + \lambda C(f),\ \lambda \in \mathbb{R}.$$

The function $C(\cdot)$ is called regularizer and according to the type of the function, the training procedure adjusts the weights respectively. The two most famous regularization functions are consisted of the l2-norm or Euclid norm and l1-norm. The l2-norm

$$||\boldsymbol{x}||_2 = \sum_i^d x_i^2,$$

is usually utilized when we want to penalize some weights more than others given the amount of over-fitting they add, leading to more stable models. On the other hand, l1-norm regularization

$$||\boldsymbol{x}||_1 = \sum_{i}^{d} |x_i|,$$

is often part of a feature extraction mechanism, although it also has pro-generalization effects, during model training. As we will see in following chapters, the regularization hyper-parameter λ, is a critical components of the defensive efforts, since the model's creator tries to balance the utility of the algorithm and the desired amount of user privacy.

Dropout Another common over-fitting reduction methods to combine with the aforementioned one, is dropout nodes. Dropout is a technique used in NN training and is interpreted as blocking out some node outputs between two consequent layers. This could be visualized in the figure below (Figure 6.1), where we can notice that, in the right version of the Dense NN, there are missing connections between randomly chosen nodes[3]. The mechanism of a dropout layer, works like a dampener of arbitrary node connections based in a probability p, which is called the dropout rate. This feature disables the information to propagate forward and in many cases prevents the model of learning dataset-specific patterns, by introducing a sense of uncertainty in the learning process, since there is a constant non deterministic information cutoff. In this way, dropout layers, prevent models (especially deep learning ones) from over-fitting to the training data set and enhances their generalization power[3].

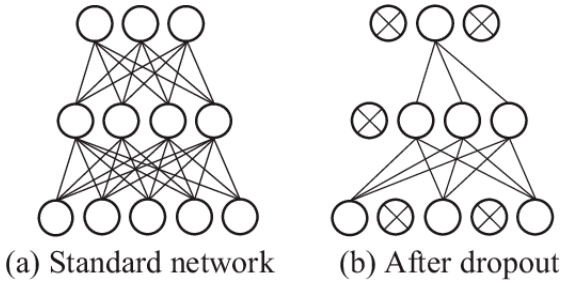

Figure 6.1: Dropout in NNs

Early Stopping One of the most famous and effective ways of avoiding poorly generalized models, is cutting off the training early, using a technique called early stopping[16]. With this technique, as the name suggests, the model stops its training, usually based on a given threshold, checked at the end of every epoch. Most common threshold is the difference of the validation error/score between epochs, given the validation set is present. This method essentially is monitoring the progress of the training and when the latter is starting to fade out, stops the training, since there is no reason to continue iterations, that could actually harm the final model generalization power. Early stopping could be even more sophisticated by adjusting the mechanism to apply triggers after a predetermined number of occurrences that indicate the need to stop early. Furthermore, a patience mechanism could be integrated into the technique in order to skip temporary obstacles in the learning processes, a usual phenomenon in deep learning.

It is important to note that all of the above techniques were not essentially created to defend statistical inference based attacks, but to prevent ML models from over-fitting. Having said

that, we could assume that by migrating the majority of vulnerabilities, inference attacks are not as effective as before. Nevertheless, there is research work[2][7]; arguing that even well generalized models could be affected by MIA attacks and argue that over-fitting is not a necessary assumption for the attack to succeed, but it sets a sufficient starting point for a successful MIA.

6.1.2 Data Driven Defences

Another way to reduce over-fitting and ensure that the the final model generalizes well is to provide the model with a diverse and representative training data set, or even utilize the data set characteristics to specifically defend the model from any adversarial attacks. Following, we present the most popular paths one can use, in order to enhance model's generalization power.

Data Sampling A simple way to ensure that the model does not over-fit on the training data points, is to acquire a representative, non skewed, unbiased data set, following a proper sampling technique that ensures the vast majority of the population can be interpreted. Having said that, our problems lie on the acquisition of this data set, since there is no cure-all that could improve the state of a biased data set, or one with outliers, which can distort the overall behavior of the model. To deal with this particular issue, machine learning researchers manufactured new techniques to enhance the representability and information completeness, the training data set can provide us with. Most of these techniques are based on inserting more randomness while sampling data, meaning that either we utilize the existent data set in a more creative way (k-fold or hold-out training), or we increase the population we are sampling from.

Data Augmentations Having explored an application of perturbing data during model training, we are now about to discuss the real reason data augmentations were originally created for. As we have already stated, augmenting an existing data set in a meaningful way, leads to a larger more diverse data set, where some records are similar in a way but explain a lot more of the original data distribution and enhance the power of the model to identify feature relationships, such as objects in picture[16]. Furthermore, data perturbations, combined with other statistical methods, are used by researchers, to further understand the distribution and perform a more informed feature selection and model design.

This process usually leads to well generalized models, but there is a catch; whether the original data set is skewed or fairly inadequate in terms of distribution representational power, the model still over-fits. As we will notice from our experiments, the Label Only framework performs similarly to the ”undefended” case, since making the target robust to perturbations, in combination with over-fitting on the training set, leads to further privacy leaks.

Adversarial Examples Finally, we talk about a more sophisticated application of perturbations, adversarial example training. In this technique, the user tries to train the model and at the same time, defend it by incorporating adversarial examples to the training procedure[17]. Adversarial examples are nothing more but perturbations of the initial data points, but they differ to the point that they are initially made to confuse the model and lead

to false predictions. The combination of ”adversarial” and ”standard” learning is key to this kind of defences, with the first trying to minimize the so called ”robust” loss, while the latter, takes steps towards ”standard” loss' minimum[17]. Nevertheless, Song et al[12][11]; proved that the adversarial training, would only enhance the privacy related vulnerabilities and make no difference to the performance of a MIA, while under certain circumstances, it might even increase the performance.

6.2 Defences Against Inference Attacks

In this section, we turn our attention to more case-specific defences, constructed for the sole reason of defending inference based attacks. Firstly, we explore the effect of confidence masking on attack performance, while, in the second part of our study, we introduce differential privacy and describe of how we could be utilizing its mechanics to defend our model.

6.2.1 Confidence Vector Masking

As one of the first defences against Membership Inference Attacks, confidence vector masking, is designed to provide the minimum amount of necessary information to the user-end. In this notion, ML services, provide the user with a part, or the whole, confidence vector, obfuscating the view of the data for the adversary and possibly mitigating any over-fitting related, privacy vulnerabilities[5][10]. The most famous defences in this category are label slicing or confidence restriction, noise injection into confidence vector and Memguard, which is seemingly, a noise addition technique, but with some refined alterations during the injection process.

Confidence Restriction Applying restrictions to the confidence vector, was initially documented by Shokri et al[10]; in order to experiment on the effect of the attackers knowledge on confidence vector and the relationship of this information to the overall attack power. As the name suggests, the ML services, provides users with a segment of the final output vector, usually including the first n top scoring labels, along with their confidence scores. In this manner, the model becomes more robust to membership inference attacks, since the better part of the meta data, referring to the models behavior on training and foreign data, is concealed or ”masked”, resulting in attack surface reduction. Nevertheless, Shokri et al[10] along with other researchers[2][5][13], have proved this argument false, by displaying a similar and sometimes identical attack performance, when the provided label number has fallen significantly. We come to an agreement, with these results in our own evaluation sections (see the next chapter), testing our attack on CIFAR-10 data set, in particular. So one could see that, a seemingly trivial way to obfuscate the target model's behavior, are quite meaningless, when over-fitting takes over.

Noise Injection Another famous mechanism to prevent privacy leaks in machine learning models is injecting random noise into the output probability vector. In this version of confidence masking, Shokri et al[10] study the how small additions of normal noise affects the MIA framework scores. At this point, we should notice that random noise addition, not only might change the probability scores of a specific label, but it could actually

lead to a complete change of the labels ordering, hence the output vector indicates misclassification. Having said that, the data scientist has to estimate the return of such an integration, with respect to the loss of model utility and preserve a balance between the privacy loss and the utility loss. On the other hand, as Shokri et al[10] and other papers[2][5]; display experimental results, arguing that, this defence, falls into the same pitfall as the previous one, since unorganized attempts to obfuscate output confidence scores do not stop the effect that over-fitting commands on the model's behavior on training data.

Memguard The last and most effective confidence masking method was proposed by Jia et al[6] at 2019 and it is widely known as the Memguard mechanism. It incorporates the noise injection method, along with adversarial examples construction methods, to create a pipeline which obfuscates the final ouput vector of the model, but reserves model-utility guarantees. Essentially, Memguard is an active attack, meaning it attempts to attack the adversary's attack model. As we have previously discussed the attack model is nothing more that a binary classifier, which could be attacks with the aid of adversarial examples. The general idea behind this noise injection method, lies to the assumption that the ML engineer can inject specific noise that confuses the attacker model and result into misclassifing the membership status of a victim user, thus introduce further uncertainty to the adversaries predictions and preserve privacy. Moreover, this defence mechanism guarantees that the original predicted label remains the same, hence model utility does not endure any important decrease[5][6]. According to researchers[6], the experimental results of this defences seem quite promising, resulting into closing the gap between the baseline attack and Shokri's version of the confidence vector attack[5][6][10].

The Label Only Danger The defence mechanisms we discussed in this section, focus on reducing the attackers information and harden the membership inference process, with the assistance of information obfuscation, either by integrating random or carefully crafted noise into the confidence score, or by cutting off the better part of the output probability vector. On the attacker side, this poses a serious issue, most of the times, which could be tackled with the use of the label only framework. Assume that our model integrates 2 or more defences, for example Memguard along with total Confidence Restricting (1 label only), resulting in a, seemingly, well defended model. Label Only creators[2]; guarantee that providing the framework, solely with the predicted label, the first returns a high confidence membership status prediction. The reason behind why this guarantee holds is rather simple; confidence masking defences, focus on protecting the model, while keeping the utility intact, meaning that the predicted label is the same in both raw and obfuscated output vectors. Given this context, the assumptions of Label Only framework hold and the adversary can infer membership with accuracy very similar to the default confidence-based attack.

6.2.2 Differential Privacy

At this section, we introduce the differential privacy paradigm and explain how we could use it in privacy preserving machine learning.

Randomized Response One of the first and most famous privacy preserving mechanisms is randomized response, used in surveys, which item of study concerns private and

undisclosed information about the participants. In order not to stigmatize the participants and acquire accurate and legitimate data, researchers answer a ”yes or no question” according to the following paradigm[4]:

1. Flip a coin,
2. If landed on heads, answer truthfully
3. Otherwise, flip another coin,
   (a) If landed on heads, answer negatively
   (b) Otherwise, answer positively

In the same notions, researchers, came up with a more sophisticated and more effective, in terms of utility, mechanism, called differential privacy.

Definition of Differential Privacy Before introducing the reader to DP, we define the following;

- A **Randomized algorithm** M, of domain A and discrete range of results B, is associated with the mapping $M: A \rightarrow B$
- **Distance** between databases A and B is defined as the l_1-norm, $||A-B||_1$, meaning the count of the items in which they differ. For example id A and B differ in a single row, then $dist(A, B) = 1$.
- **Adjacent databases**, are 2 databases, A and B, that have distance of 1.

Now we are ready to define Differential Privacy. Given a randomized algorithm M, of domain $N^{|A|}$, we say that this mapping is (ϵ, δ)-deferentially private, whether $\forall S \in Range(n)$ and $\forall x, y \in N^{|A|}$ s.t. $||x - y||_1 \leq 1$, then

$$Pr[M(x) \in S] \leq e^{\epsilon} Pr[M(y) \in S] + \delta.$$

Note that the probability space is over the coin flips of the mechanism M.

General Applications To start with, differential privacy is a concept usually integrated in data analysis systems, where the analyst is allowed to perform aggregation queries on them. Utilizing differential privacy, the system’s administrator makes sure that there is no way, the attacker can distinguish whether 2 databases are adjacent, leading to a privacy leak about the individual data point, that consists the difference between them. As one can understand, if we set $\delta = 0$, then we can have a (ϵ)-differential private mechanism, but what ϵ really means in this context? By carefully interpreting the above mathematical formula one could understand that

- the larger the ϵ gets, the less the databases seem alike
- the smaller the ϵ gets, the harder it is for an data analyst to derive

Researchers usually characterize this quantity as the *privacy loss* of the data analysis systems, defining it as

$$Loss(M) = \log \frac{Pr[M(x) \in S]}{Pr[M(y) \in S]} \leq \epsilon$$

Theoretically, we could have a totally differential private mechanism, with no privacy loss, where no difference is ever observed between 2 databases, but this system is useless in any data analysis process. Here is where δ comes to play, often interpreted as the **privacy cost**, added to the privacy mechanism to enhance utility of the data analysis system, but decreasing its privacy guarantee. Having said that, differential privacy is ultimately a way to ensure that the output of the mechanism is independent of database adjacency, meaning query results are most likely similar, regardless of individual row existence. In this way, the ability of an arbitrary adversary to recognise individual data points between 2 neighbouring databases, is bounded by ϵ.

Applications in ML Training Next point of conference is how such a system is integrated in an ML pipeline. For starters, we could argue to use the mechanism before training, resulting into a pre-processing stage integration, which should be applied to all input data points, making it quite trivial to the adversary to attack our model, since this pre-processing pipeline is available in order to provide the users with the model. More effective ways of applying differential privacy mechanism into the ML pipeline, are integrating it into the training phase[1][5][8], or even at the deployment phase, by adding customized noise to the final trained weights of the ML model[14].

Private Training DP-SGD The first and most common way of adjusting ML training to promise the differential privacy guarantee, is to enhance the weight adjustments with a noise addition mechanism. In their paper Abadi et al[1], proposed a new version of the stohastic gradient descent algorithm

1. Initialize weights, θ_0, randomly
2. For t training steps of total T
   (a) Take a **random sample** with probability $\frac{L}{N}$, where L is the sample size and N the dataset size
   (b) **Compute gradient**
   (c) **Clip gradient**, based on a hyper-parameter C
   (d) Add **normal noise**, from $N(\sigma^2 C^2 \boldsymbol{I})$, where σ is the noise multiplier
   (e) **Update weights**

At the end of this procedure, we have an algorithm where every step is (ϵ, δ)-deferentially private, resulting, by the summation rule[4], to a $(q\epsilon, q\delta)$-differentially private algorithm, where $q = \frac{L}{N}$, being the sampling rate[1]. Another important part of this algorithm, which is derived easily from the theory, is that the noise multiplier, σ, inversely proportionate to ϵ, meaning that the more noise we add, the more differential private "we are". As Abadi et al argues, there is a serious need of tuning σ and C in order to balance privacy and model's utility. According to their experiments, studying the mechanism's behavior on different values for epsilon and delta, there is a capacity of up to 8 units of epsilon quantity, while the value of delta is always bounded from the respective data set's size. It is important

to note here that, differential privacy principles[4] want δ to always be smaller, than the inverse of the data set's size, in order to keep privacy guarantees, without unexpected leaks. Another, quite remarkable, finding of those experiments is that, privacy-wise, we can afford an upper bound of privacy loss of values greater that the standard differential privacy mechanisms[1][4].

Differential Obfuscated Predictions Another effective technique of applying the differential privacy mechanism on a ML pipeline is described in Wu et al's[14] work. Their method, namely Bolt-On Differential Privacy, integrates a differential privacy machine at the end of the training, by perturbing the final weights, using a similar noise injection engine as the one above. This method can easily be appended at any given iterative training algorithm, since it is treating the latter as a black box, by adjusting only the final weights, without changing the base-algorithm process. Bolt On technique is allegedly better that DPSGD, since the convergence of the algorithm is faster, while the privacy-model utility gap seemingly closes by a lot, compared to the in-training noise injection mechanism[14]. Wu et al[14], reports results similar or better to the DPSGD method[1][8], while the overall models utility, with respect to resources and time management, is improved. Furthermore, Wu et al, indicates that this method is derived by the combination of output perturbation methods, which we previously described, along with the idea of noise injection in the weights of the ML model. Having said that, it is safe to trust the experimental results against Label Only Framework are valid, since this mechanism breaks the assumption of real and adjusted prediction labels to be the same.

7. DEFENCES EVALUATION

7.1 Over-fitting Reduction Defenses Evaluation

In this section, we focus our research in general mechanisms that prevent over-fitting in machine learning training. Over-fitting, being a highly sufficient factor of a successful MIA, consists the majority of privacy-breach dangers in machine learning. Nevertheless, the readers must be aware that eliminating over-fitting does not mean that their models are completely safe against MIAs, since there are attack scenarios where, although the model-victim is well generalized, the adversary target outlying victim records, on which the target model behaves differently, leading to a similar privacy leak[7].

In our research, we test our MIA frameworks against generalized models that use famous generalization mechanisms such as early stopping, dropout layers, weight regularization (decay). Furthermore, we elaborate on the reasoning behind failure of augmentations training to decrease MIA's success, along with some educated hypothesis on what effect does this type of training really have.

In our experiments, we use CIFAR-10 data set following Shokri et al[10] experimental set up, with the structure of the attack model, being the sole difference, since we are using the unified attack model we proposed earlier (chapter 3).

7.1.1 Dropout and Regularization Effect

In this experiment, we apply different training settings, with respect to dropout and weight decay tuning, in order to determine which method is more effective on defending against MIAs.

As long as our experiments settings, we tried different D_t sizes, sampled from a total of 50000 data points, in order to include the data set size effect on the training mechanism. On the adversary side, we fixed a single attack setting, with

- $|D_a| = 10000$
- 15 shadow models,
- $|D_{shadow_i}| = 7500$ records
- no dropout/regularization used in training the shadow models

We start with Figure 7.1, displaying a high level report of the dropout effect on the attack scores and the vulnerability metrics. Note that it follows a min, max, avg trend of the attack metrics, aggregated by the desired variable (i.e. drop-out rate). We observe a decreasing behavior in all 3 metrics, with the precision metric having the greatest gap between min, max and mean plots, meaning that it was less affected by the addition of dropout that the rest of the metrics. Nevertheless, as we stated before, a high precision attack, is not always the desired attack, since the steep fall of attack recall score, is a strong indicator of an effective defense against MIA. Furthermore, the estranged lines indicate non stable behavior among different training settings. For now we assume that, since the min plot is quite stable, it resembles a high rate of regularization, combined with the largest data set D_t, we provide the target model with, during training.

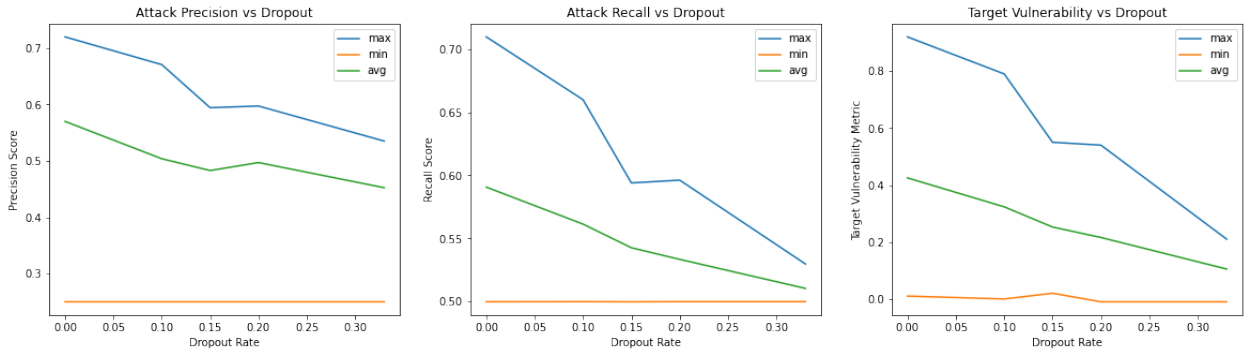

Figure 7.1: High Level View of Dropout Effect on Attack Performance and Target Model's Vulnerability

Following, we see that the model vulnerability resembles the recall line-plot, starting at a high of over 0.8 units and resulting a little under 0.2, referring to the max plot, while it displays a relatively stable behavior, first on the min plot and then on the monotonously decreasing mean plot. This indicator, leads us to the conclusion that although dropout is a fairly satisfying defence mechanism, there is a need for extra protection, since in low-sized data sets there is still a remarkable attack performance, even if it scores closer to base-line threshold.

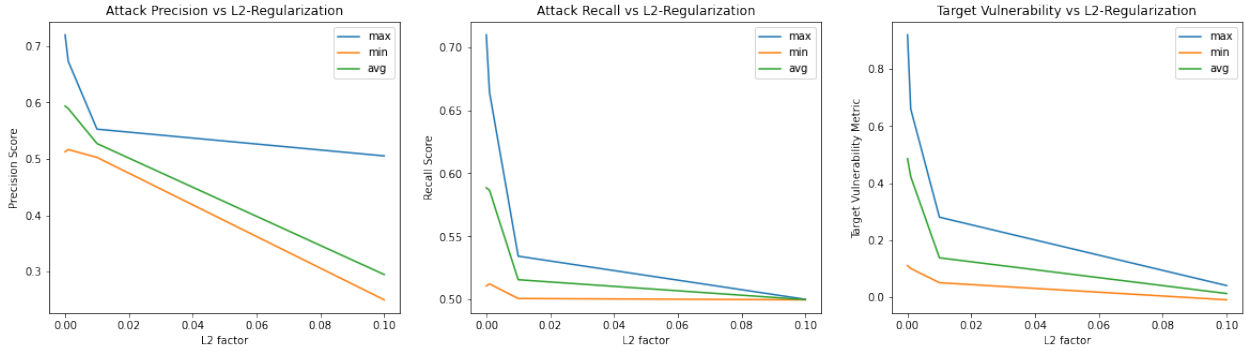

Figure 7.2: High Level View of L2-Regularization Effect on Attack Performance and Target Model's Vulnerability

In the same notion, we now take a look at Figure 7.2, the $l2$-regularization plot, where we find a rather different behavior, although it is quite expected. To start with, the model vulnerability and precision graphs, are indicating that small increments in regularization rates, drop the attack performance quite fast. It is also important to notice, how closely min, max and average plots are placed in all 3 graphs. This fact is strong evidence that L2 regularization in small proportions can defend the model from over-fitting and decrease the adversary's advantage over the target. Another conclusion we arrive to by watching Figure 2, is that L2 Regularization is a much more efficient way to decrease the chances of a successful MIA attack, since it drops the precision score below threshold in the average use case, preventing any kind of attack, including the threshold based attacks. Nevertheless, this means that the target model might lose some of its predictive utility, so for the best results the model designer should tune the model with respect to both predictive and privacy utilities.

In the next couple of pages, we display the detailed graphs from our experiments and we comment in the general MIA behavior over different target-model training settings.

In Figure 7.3, we see that the coloring coding of the attack behavior follows a logical

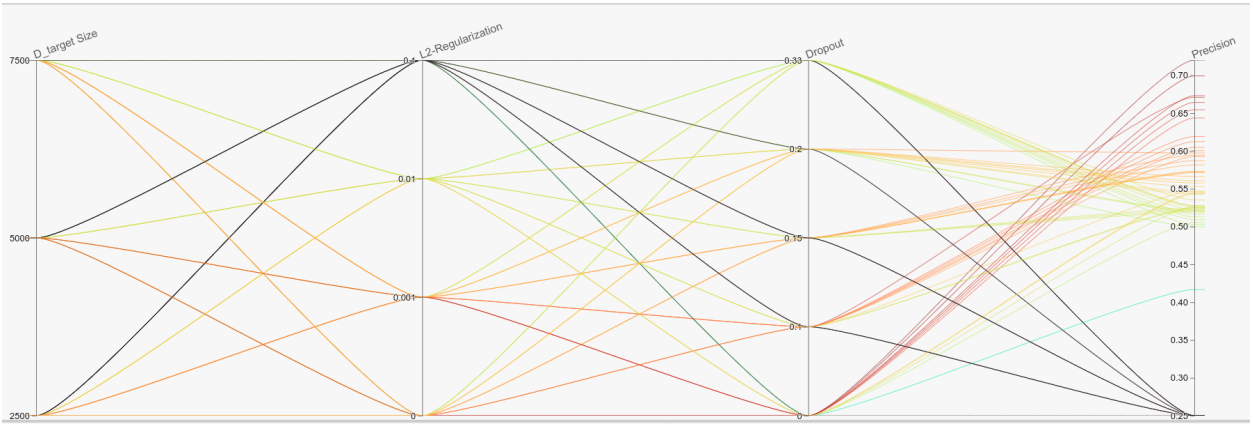

Figure 7.3: Detailed View of Dropout & Regularization Effect on Attack's Precision

map; the more red there is in the colors, the more dangerous the attack is, leading to higher performance. From the precision figure, we arrive to the conclusion, that higher regularization rates, lead to poor attack performance, as we have already suspected from the investigation of the two previous figures. On a second view of the figure, one could also notice a previously elaborated point; when the target model data set, D_t, is smaller that the attackers data set, D_a, then the attack is hitting its highest scores. On the other hand, it is now clear that proper tuning of dropout and regularization rates, can even handle this phenomenon, preventing the target model from over fitting and dropping the precision metric close to baseline.

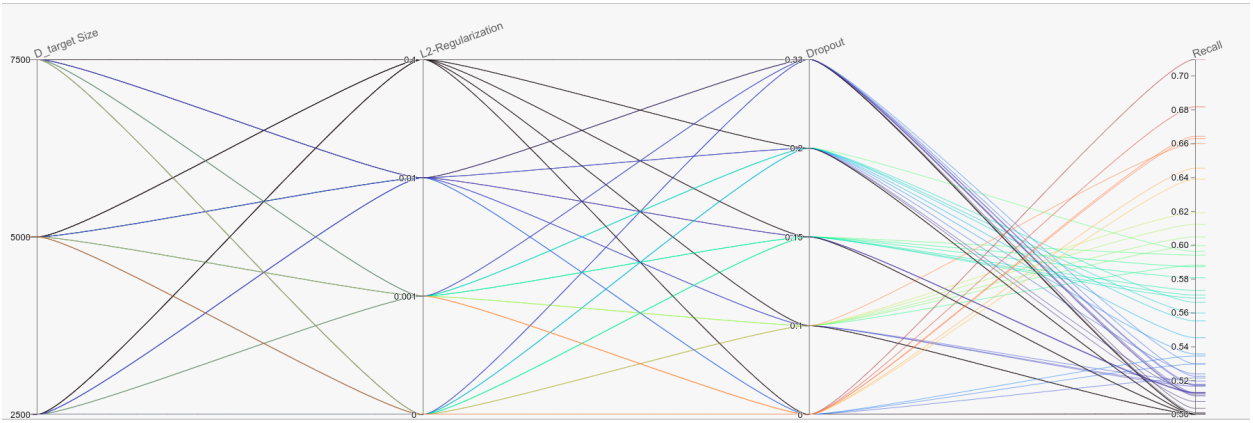

Figure 7.4: Detailed View of Dropout & Regularization Effect on Attack's Recall

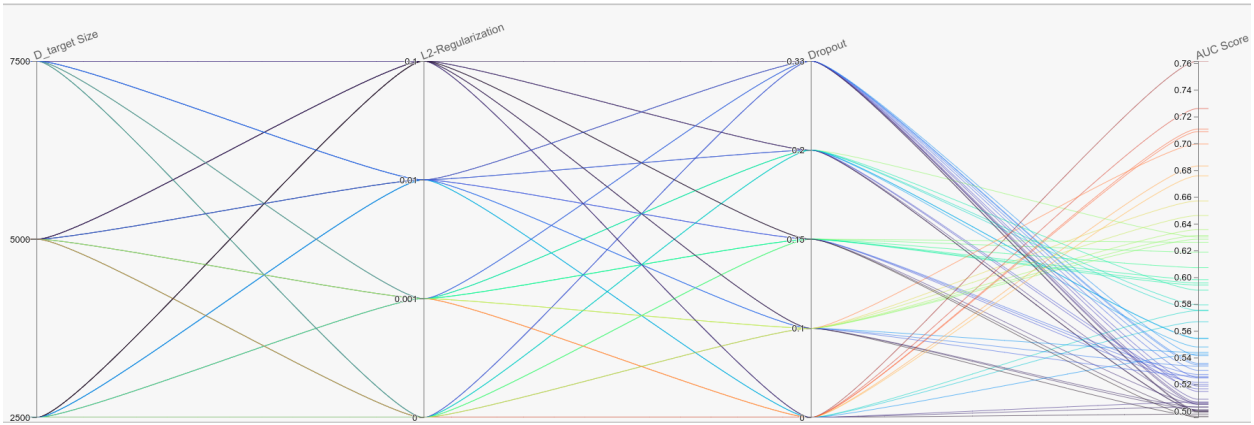

Figure 7.5: Detailed View of Dropout & Regularization Effect on Attack's AUC Score

We follow with the commentary of Figures 7.4 and 7.5, the recall and AUC score multi-dimensional graphs. The coloring is again leading to the same conclusion. As we have assumed earlier, l2-regularization during training, decreases the recall scores, earlier in the tuning scale, than dropout layers would. Since recall, indicative of how sensitive our attack model is on identifying ”members” correctly, is the most reliable performance metric for this attack, we conclude that both ”Target-vs-Attacker” data knowledge and l2-rate tuning, consist the most important defense mechanisms against attacks that exploit over-fitting. As usual, the AUC score follows the same behavior with recall metric, backing up further our point about attack model's sensitivity and its relation with anti over-fitting mechanisms.

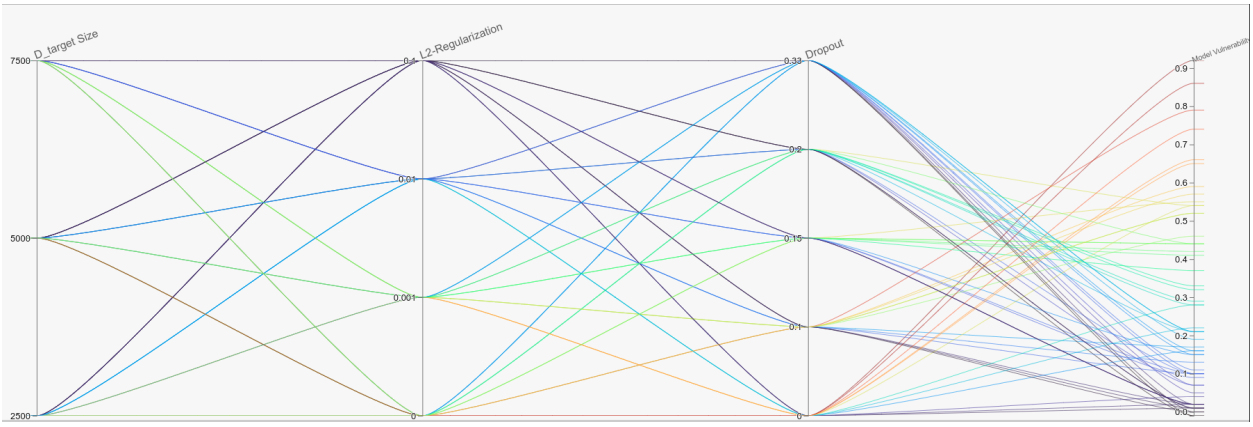

Figure 7.6: Detailed View of Dropout & Regularization Effect on Target Model's Vulnerability Metric

The last figure, Figure 7.6, refers to the model vulnerability scores throughout our experiments. Notice how the vulnerability metric scores are indicating a seemingly different result to the rest of the study; the dropout rate seems to be the main factor, that constrains the upper and lower limits of the vulnerability metrics. We may observe, how the more dropout we add, the smaller the metric values diverge and the lower this value-range is mapped in the ”Model Vulnerability” scale. However, if we notice the primary coloring of every distinct value range, we see through the illusion of the complex graph and understand that l2-rate is dictating the ranges. For instance, it is trivial to see that the ”colder” colors, the primary colors of the lower end in the metric scale, are mainly sourcing from high regularization rates.

7.1.2 Early Stopping Effect

Often utilized in research work[2][5][9][10][13], early stopping is considered one of the fundamental mechanisms to reduce over-fitting during training. In this part of our investigation, we display how effective early stopping is against leaking privacy information during training. Notice how early stopping is preventing over-fitting by stopping the training earlier than the max epochs given by the data analyst, hence prevents the model from becoming significantly more sensible to training data.

We follow similar set up as in the previous experiments and we focus in the effect of early stopping tuning before training, based on target's performance on a validation set. Note that we tune only the min delta parameter, while the patience hyper parameter is fixed to 7-10 epochs in order to ensure that the target model gets sufficient training and the latter is not cut off too soon.

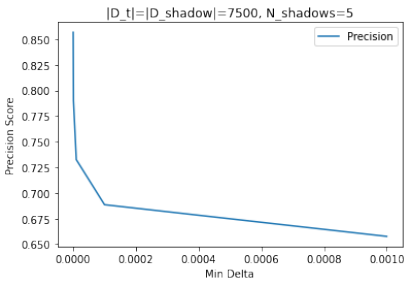

Figure 7.7: Attack's Precision

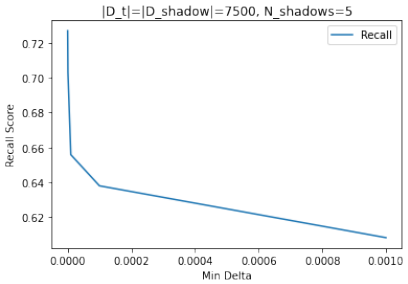

Figure 7.8: Attack's Recall

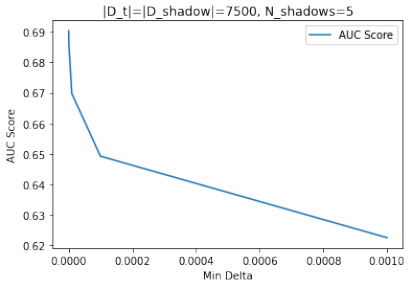

Figure 7.9: Attack's AUC Score

As we observe in the above figures, the effect of early stopping during training, is one of the most effective ways to defend against attacks that exploit vulnerabilities related to over-fitting. Considering that early stopping mechanism's min delta hyper-parameter is usually tuned from values in $\{10^{-i}, i \in \mathbb{N}^{+}\}$, we can notice that we allow pretty low increase/decrease of the monitored quantity and still end up with a fairly generalized model, which defends against a MIA attack on a satisfying degree. Nevertheless, we observe how the lowest scores we hit are still above the baseline threshold, indicating that no matter how much we reduce the over-fitting degree, we might still be vulnerable to a MIA[5][7][10].

7.1.3 Augmentation Training

In this experiment, we investigate the effect of augmentation training to MIA performance. We utilize our Label Only Framework since this is the most interesting aspect of this attack-defense combination and we take a different path to explain the attack's behavior on the respective defence.

The set up is the same as in the related bibliography[2][5][10]; and we use a slice of CIFAR-10, as the data set under attack. Firstly, we train an attack on a similar CIFAR-10 slice, using a fixed perturbation tuning, same as in [2]. Afterwards, we re-train the target model, but this time we integrate perturbations of the exact same settings as in the Label Only Framework, in order to make the model more robust to inference through perturbations. We keep track of the attack performance but most importantly, we trace the number of divergent predicted labels during the attack phase. We also display a distribution, of how many data points diverge from their originally predicted label, during the attack data set construction phase, both for queries inside and outside the target data set, in order to justify the attacks behavior onto the defended and undefended model. Following, we present a board of the metric scores that both attacks achieved and the figures that display the true-vs-predicted label divergence.

Table 7.1: MIA Performance on simple and augmentation enhanced training of the same model

Metrics/Model	Precision	Recall	AUC Score
Normal Training	0.85	0.83	0.85
Augmentation Training	0.84	0.80	0.80

Table 7.1 indicates that, the same attack scores similar performance on both the CNN model that was trained normally and on the one that was trained with an augmentation enhanced training data set. But why would this be the case? Why does every other over-fitting preventing defense works sufficiently, while augmentation training fails to that

degree? In order to answer these questions we will take a look to the label divergence bar-plots.

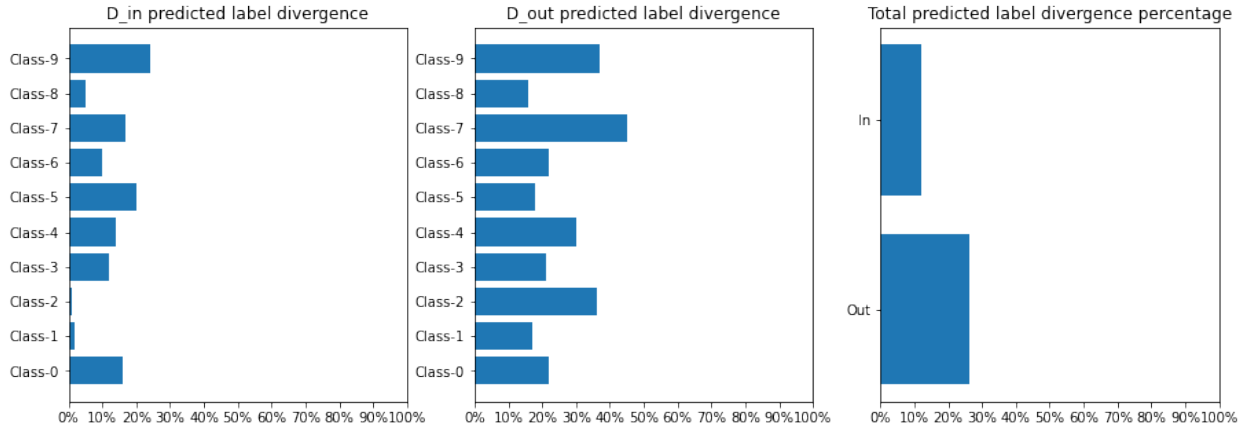

Figure 7.10: Label Divergence without Augmentation Training

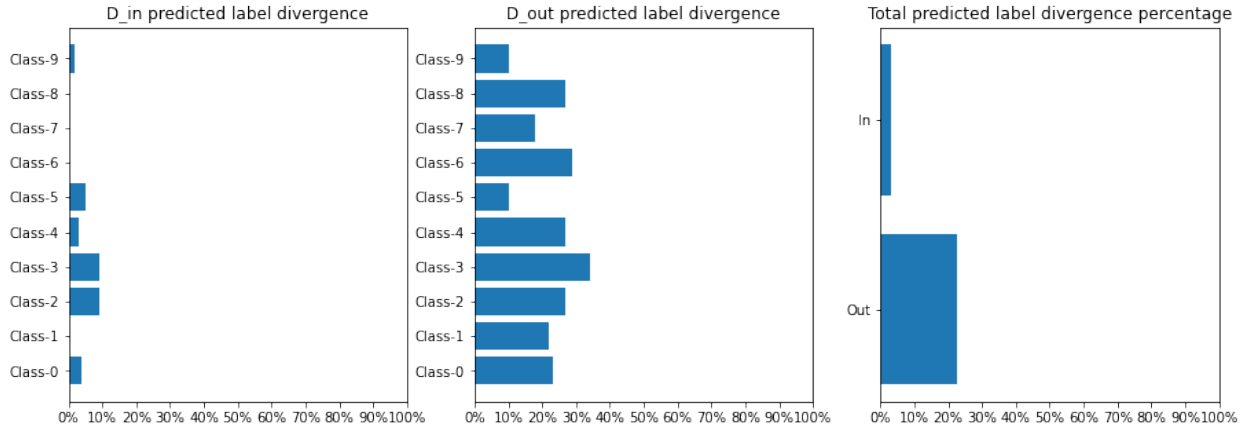

Figure 7.11: Label Divergence with Augmentation Training

We start by noticing the divergence distribution in the ”undefended” model. We notice that in most of the classes, the data points inside D_t are more robust to perturbations, with respect to the predicted label. This means that close-to-original perturbations, are most possibly being assigned the same label as the original data point[2].

On the same spirit, we notice an exaggerated notion of this situation in Figure 7.11, where the data points inside D_t are highly robust to perturbations, which is expected since their perturbations was used during the model training. Furthermore, by interpreting the total label divergence distributions, we can conclude that the augmentation training made slight to zero difference to the the label divergence distribution of data points that do not belong in the target data set.

Taking everything under consideration, along with the fact that the target model still displays signs of over-fitting, we can conclude that this defense is highly reliant to the quality of the data, since training on the same data set and augmentations of it results into a similarly vulnerable model.

7.2 Confidence Vector Masking Defenses Evaluation

In this section, we investigate the effect of confidence masking defences against membership inference attacks. As Shokri et al[10] and others[5][7][9][13] have stated before, confidence masking defences, in their simplified version, are not effective against MIA attacks. The simplest confidence masking defence is the confidence slicing mechanism. In this defence the MLaaS administrator, reduces the information flow between the data analyst and the model, by providing the analyst with k labels, usually the top scoring ones. In this manner, we reduce the privacy leaks by decreasing the features the inference classifier could exploit to identify members of the target dataset. Nevertheless, as we shortly see, this kind of defences are not quite effective. This study focus solely on exploring the behavior of different attack settings on a context where a model applies confidence restriction mechanisms.

As usual we follow the Shokri et al set up and we use the CIFAR-10 dataset since it provides 10 classes, which is a sufficient label count to elaborate on our point. One important detail to have in mind is that both the target and shadow models, provide k confidence scores along with the respective label.

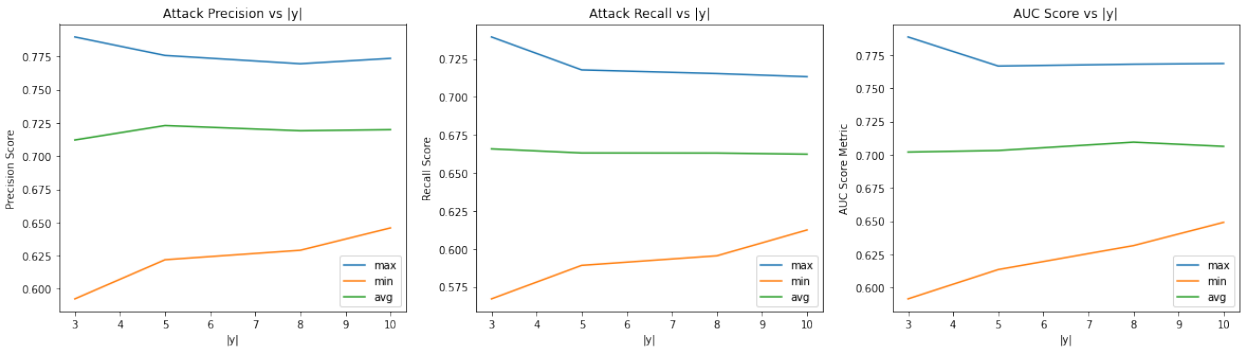

Figure 7.12: MIA Performance vs Confidence Score Slicing

Starting with an aggregated graph, in Figure 7.12, we easily observe that the min, max and average lines are quite stable and they diverge from each other by a lot, meaning that the attack performance is not correlated to the number of the confidence scores provided. Following, we can notice something quite interesting; the max plot is manifesting a global maxima while attacking a model that emits a confidence vector of 3 elements/probabilities. This indicates that, not only the attack is not affected by the output omission, but that smaller feature sets might even achieve higher MIA performance, leading us to believe that decreasing the attack feature set might work as a feature selection process. Having that in mind, we will proceed in a more detailed examination of our findings, considering the attack configuration along with the target training settings.

The figures in this section are referring to training models that utilize an early stopping mechanism with delta of 10^{-4}, with patience of 10 or 12 epochs. The same early stopping settings are used during the shadow model training, since the target model is provided as a black box to the attacker.

By observing the figures of precision, recall and AUC, we can understand that the attack performance has little to do with the size of the produced confidence vector and more with the attack settings that we discussed in the respective section of the Attack Evaluation chapter. Since this is the case, it is interesting to note how the variance of all three metrics is changing with respect to the confidence vector size (refer to the plots below the graphs).

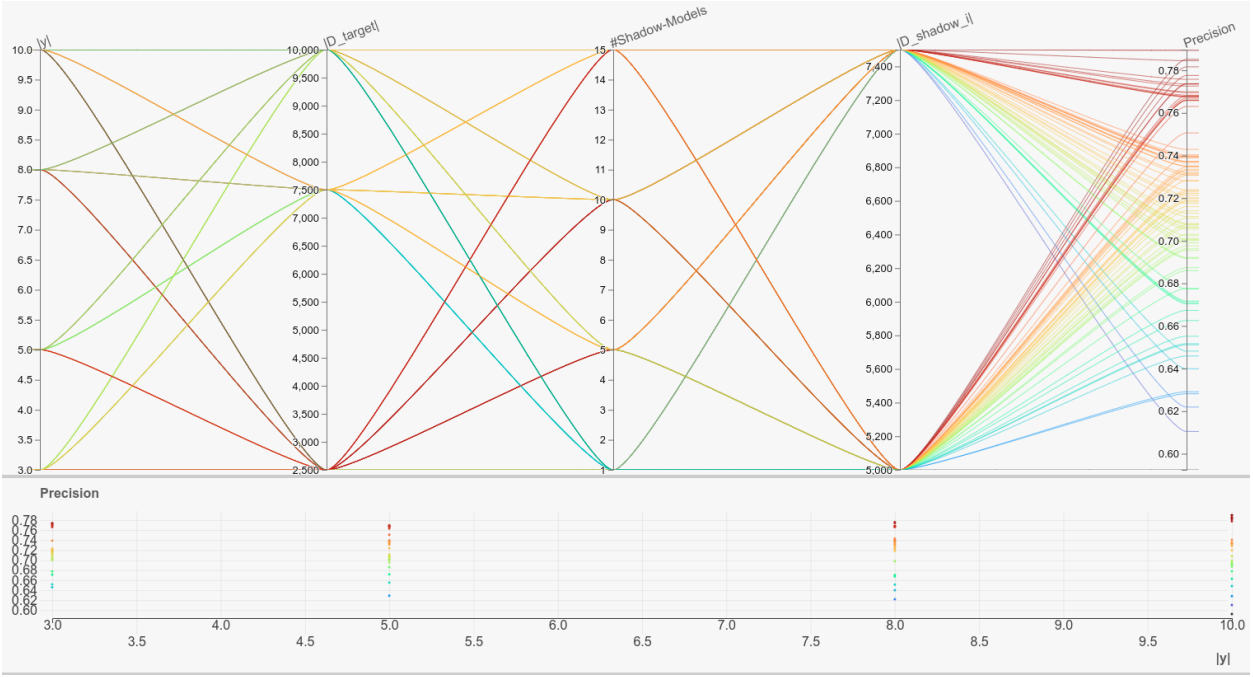

Figure 7.13: MIA Precision vs Confidence Score Slicing

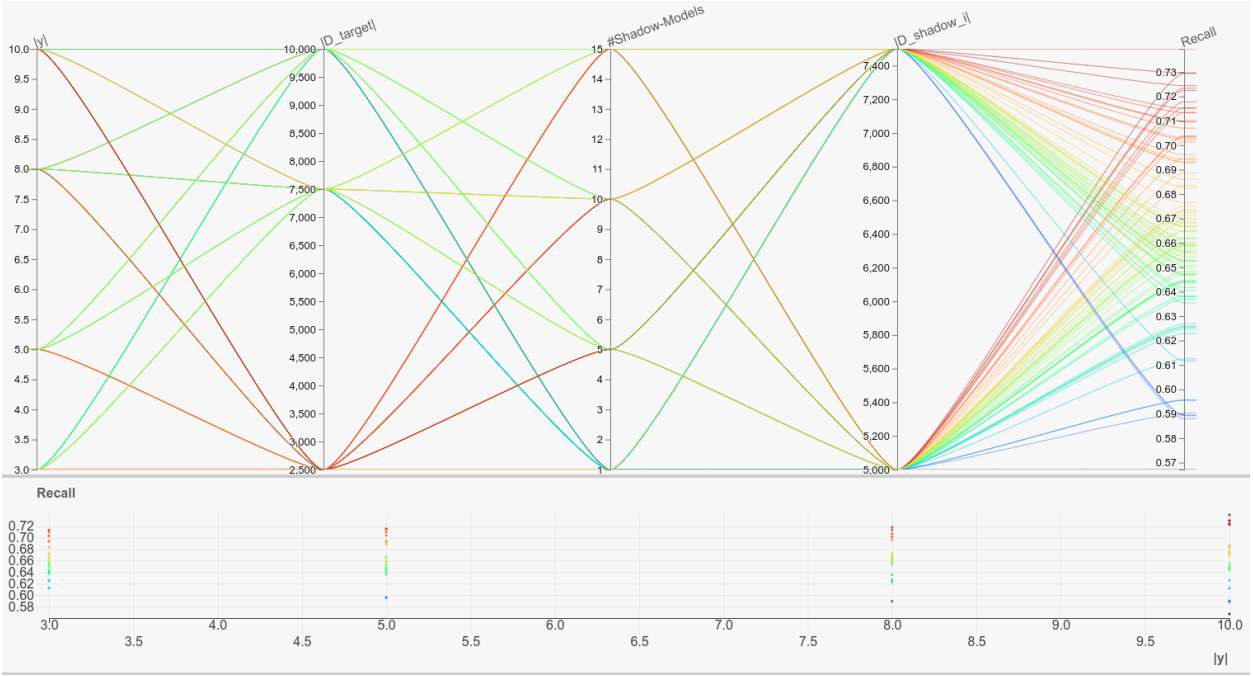

Figure 7.14: MIA Recall vs Confidence Score Slicing

We notice a constant ”spreading” of the attack score as we increasing the number of the classes provided to the user/analyst. This means that the attack model might be over-fitting on the produced attack dataset, since it displays a more stable behavior the smaller the output probability vector gets. In this spirit, it is important to notice again how much effect the number of shadow models has to the overall attack performance. One could notice that most red lines (high performing attacks) source right from the upper parts of the shadow model number axis, with the more blue or green (low performing attacks) to source from the lower half of this axis.

Finally, in this study, we did not touch the Label Only context. This might seem contradict-

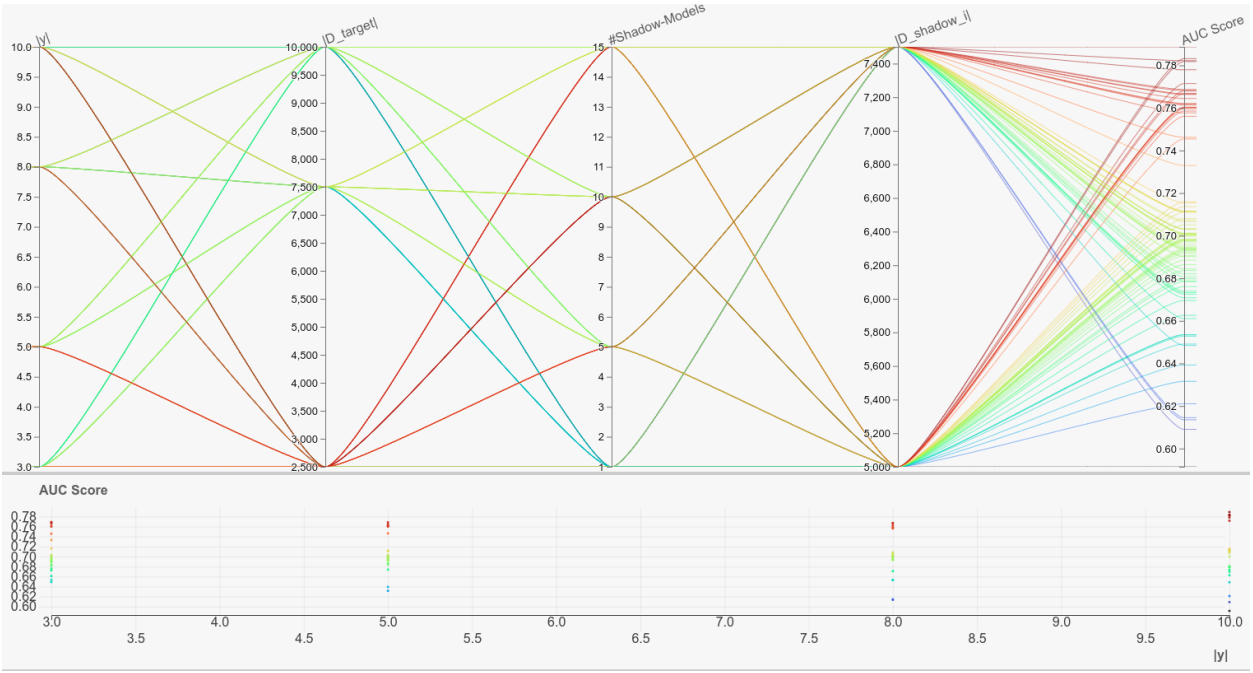

Figure 7.15: MIA Precision vs Confidence Score Slicing

ory to what we previously have stated, but remember that the Label Only context is the only confidence slicing defence mechanism that decreases the confidence based MIA, since the latter is highly dependent to the relation between the predicted label and the rest of the confidence vector. Taking away this context, as a defence against MIAs, is quite effective to attacks similar to Shokri et al's, but not enough to defend a more subtle and complex version of the Membership Inference Attack[2].

8. CONCLUSIONS AND FUTURE WORK

The goal of this thesis was to deep dive into the general notion of Membership Inference Attacks and understand the reason that this attacks could succeed in real life scenarios. Furthermore, we investigated alterations of the attack, along with defences that a model designer could utilize to prevent privacy leaks during model querying. Finally, we provided an attack framework that enhances our points and the readers understanding of the attack and relative defences.

Membership inference attacks pose a serious privacy threat for all users that are included in ML-related studies, when the responsible analysts ignore the privacy context that their experiments are being conducted on. From our experimentation with the attack and its alterations, we could argue that no matter how safe of a model seems to be, even if it generalizes well enough, it could breach users' privacy, after the proper exploitation of the inherit vulnerabilities of ML training.

On a different note, there are many countermeasures that we could integrate inside the ML pipeline to protect user privacy, with most effective of them being, over-fitting reducing methods, along with smart noise injection, during or after the ML training phase[1][4][8]. Nevertheless, simply reducing the over-fitting degree of a model does not promise that it would be robust to a carefully crafter MIA[7].

In our experimental studies, we displayed how dangerous and effective the confidence based MIA is, as well as its Label Only alteration, proving how subtle and well hidden the privacy vulnerabilities in an ordinary ML pipeline are. The membership inference research field, is continuously developing, meaning that there is further exploration on the inference-related vulnerabilities of widely used ML models.

Our proposition on continuing the experimental work of this thesis would be the construction of a tuning system for the label only attack framework, that could perform a perturbation-feature selection, in order to maximize the attack's performance. This could be accomplished via a GAN-like mechanism that tunes a model, defending against Label Only MIAs, like its Memguard[6] equivalent. More interesting results could derive by incorporating Differential Privacy into our defence mechanisms and apply tuning, aiming to close the gap between model's privacy and utility.

ABBREVIATIONS - ACRONYMS

MIA	Membership Inference Attack
ML	Machine Learning
SVM	Support Vector Machine
NN	Neural Networks
DP	Differential Privacy
CSV	Comma Separated Values
SGD	Stochastic Gradient Descent

BIBLIOGRAPHY

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2016.

[2] Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks, 2020.

[3] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations, 2015.

[4] Roth A. Dwork, C. *The algorithmic foundations of differential privacy.* 2014.

[5] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey, 2021.

[6] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples, 2019.

[7] 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, 2018.

[8] Md.Atiqur Rahman, Tanzila Rahman, Robert Laganière, and Noman Mohammed. Membership inference attack against differentially private deep learning model. *Trans. Data Priv.*, 11:61–79, 2018.

[9] 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, 2018.

[10] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2016.

[11] Liwei Song, Reza Shokri, and Prateek Mittal. Membership inference attacks against adversarially robust deep learning models. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 50–56, 2019.

[12] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 241–257, New York, NY, USA, 2019. Association for Computing Machinery.

[13] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*, 14(6):2073–2089, 2021.

[14] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey F. Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics, 2016.

[15] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting, 2017.

[16] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

[17] Aleksander Madry Zico Kolter. *Adversarial Robustness: Theory and Practice*. 2018.