

Universidad ORT Uruguay  
Facultad de Ingeniería

**Application of Private  
Aggregation of Teacher  
Ensembles framework for  
malicious web request detection.**

Entregado como requisito para la obtención del  
título de Ingeniero en Sistemas

Sebastián Sosa - 198955

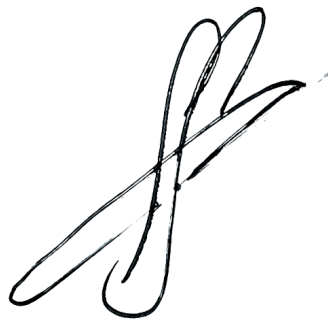
Tutores: Ramiro Visca y Sergio Yovine

**2021**

# Declaración de Autoría

Yo, Sebastián Sosa, declaro que el trabajo que se presenta en esta obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba el Proyecto;
- Cuando he consultado el trabajo publicado por otros, lo he atribuído con claridad;
- Cuando he citado obras de otros, he indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- En la obra, he acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente qué fue contribuido por otros, y qué fue contribuido por mí;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

A handwritten signature in black ink, consisting of several loops and a long horizontal stroke extending to the right.

Sebastián Sosa

**18/03/2021**

# Agradecimientos

Agradezco a mis tutores, Dr. Sergio Yovine e Ing. Ramiro Visca, por la guía y ayuda brindada en la realización de este trabajo.

# Abstract Español

El presente trabajo surge como una investigación vinculada a los proyectos de cátedra de Inteligencia Artificial y Big Data, y tiene como objetivo realizar un estudio de la técnica PATE: Private Aggregation of Teacher Ensembles [1].

Esta técnica permite el entrenamiento de modelos de aprendizaje automático con ciertas garantías de privatización dadas por privacidad diferencial [2]. Por otro lado, el proyecto tiene como objetivo la construcción de una herramienta de software extensible que implemente dicha técnica. Ésta herramienta permite entrenar distintos tipos de modelos de redes neuronales, y soporta la utilización de distintos mecanismos de agregación y de privatización relacionados a la técnica PATE. Luego la misma se aplica para entrenar un detector de consultas web malignas en base a un conjunto de datos sensibles, logrando no solo altos niveles de precisión, sino que también garantías robustas de privacidad.

# Abstract

The following work is part of research work related to the Artificial Intelligence and Big Data department, and its objective is to perform a study on the technique PATE: Private Aggregation of Teacher Ensembles [1].

This technique allows for the training of machine learning models with strong privacy guarantees provided by the differential privacy framework [2]. Furthermore, the project will consist of the development of an extensible software tool that implements the PATE training technique, and allows for the training of arbitrary neural network models with multiple aggregation and privatization mechanisms relevant to the technique. A classifier that detects malicious web requests using a dataset with sensitive information is then trained using the PATE framework, achieving both high precision scores and strong privacy guarantees.

# Palabras clave

Privacidad diferencial; PATE; Private Aggregation of Teacher Ensembles; Redes neuronales; Modelos de aprendizaje automatico; Detección de consultas web maliciosas; Inteligencia Artificial

# Keywords

Differential privacy; PATE; Private Aggregation of Teacher Ensembles; Neural networks; Machine Learning models; Malicious web request detection; Artificial Intelligence

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Differential Privacy</b>	<b>10</b>
2.1	Composition Theorem . . . . .	11
2.2	DP-SGD . . . . .	12
2.3	Private Aggregation of Teacher Ensembles (PATE) . . . . .	14
2.3.1	Laplacian Aggregation Mechanism . . . . .	15
2.4	Distributed Learning . . . . .	17
<b>3</b>	<b>Analyzing the privacy loss in PATE</b>	<b>18</b>
3.1	Data independent privacy analysis . . . . .	20
3.2	Data dependent privacy analysis . . . . .	20
<b>4</b>	<b>Experimental results</b>	<b>22</b>
4.1	Experimental Setup . . . . .	22
4.2	Results . . . . .	25
4.2.1	Privatization of student data . . . . .	25
<b>5</b>	<b>Software engineering efforts conducted</b>	<b>29</b>
5.1	Modelling of the solution . . . . .	29
5.2	Experiment management . . . . .	32
5.3	Multi-GPU training . . . . .	33
<b>6</b>	<b>Conclusions and lessons learned</b>	<b>34</b>
<b>7</b>	<b>Bibliographic references</b>	<b>36</b>

# 1 Introduction

The recent growth of available data and computing power, combined with the advancement of machine learning algorithms, led to vast improvements on the performance of a multitude of tasks that can be framed as supervised machine learning problems [3].

This opportunity does not come however without additional challenges, for a substantial volume of training data is often required to get the models to perform at a satisfactory level. This data is not only typically expensive to gather and pre-process, but can sometimes contain sensitive information that should be protected from third parties.

This imposes additional restrictions when publishing the data, regardless of modifications that are applied to remove personally identifiable information. This comes from the fact that multiple attack mechanisms have been described that are capable of reidentifying people from such datasets [4] [5] [6] [7]. Furthermore, even trained models that perform inference on new unseen samples are susceptible of attacks that attempt to extract information from the models' training data. [8]

Differential privacy (DP) provides a robust framework that allows for the definition of privacy guarantees with respect to individuals who participate on a dataset that is queried for information. This tool allows for the specification of upper bounds that describe the maximum amount of information revealed about any particular individual who participates in a dataset that is being queried. By formalizing privacy in such a way, systems can be built where it is clear what is the tradeoff between the utility that is obtained from querying the data, and the privacy leakage that is incurred by doing so.

PATE - Private Aggregation of Teacher Ensembles, is a technique that enables the training of machine learning models of arbitrary architecture in a way that its privacy guarantees can be described through differential privacy. The technique proposes to train multiple teacher models on disjoint sets of sensitive private data, and then use this teacher ensemble to guide the training of a student model with public, unlabeled data. The student training data is sent through each teacher

model to obtain a label prediction, and an aggregation of all teacher’s predictions is used as the training sample label. The aggregation mechanisms employed can vary, although the general idea often consists in counting how many teacher models predict each class as being the most probable, adding noise to this count, and then picking the most probable one.

The intuition behind PATE’s privacy guarantees is that if multiple distinct teacher models agree on an input label, no private data of their training examples was leaked since the conclusion was arrived as a consensus and no particular model is revealing too much information. If, however, there’s a strong disagreement among the teachers and the most probable class is likely to be defined by a single model’s prediction, the random noise added by the aggregation mechanism will play a bigger role in defining the output, therefore protecting the individual model predictions.

Under the Uruguayan legislative data privacy environment, Law N<sup>o</sup> 18.381: “Right of access of public information” states the civilian’s right to access public data gathered by governmental institutions. However, Law N<sup>o</sup> 18.331: “Law of protection of personal data” requires the anonymization of personally identifiable information before the release of a dataset. This set of regulations brings new challenges when building and releasing machine learning models that take advantage of such sensible datasets, as the original data cannot leave the organization’s premises for training models. Furthermore, once a model is built, measures need to be taken to ensure no attack mechanism can be applied to extract knowledge about the training samples, which the model may have memorized [9].

Such challenges motivate the following project, in which the claimed privacy guarantees of the PATE technique will be assessed, and it will then be applied to the problem of malicious web request detection. The goal of this work is to apply such a technique in a scenario where governmental agencies have sensible information that can provide value to the general public if machine learning models are built upon them, but robust privacy guarantees with respect to citizens whose data is used need to be set in place.

Once the PATE technique is validated, we propose an extension of the technique that enables the privatization of the training data of the student. This helps extend the applicability of PATE to scenarios in which privacy guarantees need to be ensured with respect to the student’s training data, before being sent to the aggregator.



## **Outline**

In Chapter 2 the necessary previous knowledge and a description of the PATE training technique is presented. In Chapter 3 the mechanism by which the privacy cost is computed in PATE is presented. In Chapter 4 the problem to which the PATE technique will be applied is explained in depth, along the experimental methodology followed, the proposed improvements to the PATE technique and the obtained results. In Chapter 5 the software engineering efforts that supported this research are described, and our conclusions and lessons learned are presented in Chapter 6.

## 2 Differential Privacy

Obtaining useful information from a database without compromising the privacy of the individuals who participate in the study can be described as a guarantee made by a database curator to its participants. The guarantee is that an observer can't determine a single participant's data by performing queries to the database, no matter what external information is available to them.

To quantify the amount of information of any participant that is revealed through such queries, a useful approach is to measure the maximum variation of the query output that results from the removal any individual from the dataset. Intuitively, the variation of the query output can be considered a source of information that indicates certain properties of the removed participant. For example, if the average height of people in a dataset increases after the removal of a single individual, this can be used in combination with other data points to determine the height of the removed individual.

Let  $\mathcal{M}$  be a randomized algorithm that queries a database with domain  $\mathbb{N}^{|X|}$ . This algorithm is  $(\varepsilon, \delta)$ -differentially private if  $\forall \mathbf{S} \subseteq \text{Range}(\mathcal{M})$  and for all parallel databases  $x, y \subseteq \mathbb{N}^{|X|}$  such that  $\|x - y\| \leq 1$ :

$$P[\mathcal{M}(x) \in \mathbf{S}] \leq e^\varepsilon P[\mathcal{M}(y) \in \mathbf{S}] + \delta \tag{2.1}$$

By formulating differential privacy as such, the data subject can be protected against arbitrary risks no matter what additional information the attacker employs, and no post-processing of the output can make the algorithm less private [2].

This definition of privacy has the added benefit of introducing quantitative values  $\varepsilon$  and  $\delta$  with which to measure privacy loss. It is also possible to determine the privacy loss of a group of  $k$  subjects rather than that of a single individual by defining databases  $x, y \subseteq \mathbb{N}^{|X|}$  such that  $\|x - y\| \leq k$ , or compound the privacy loss across multiple queries performed to the database thanks to the composition theorem defined below.

## 2.1 Composition Theorem

Often times, it is convenient to understand the privacy impact of performing a series of mechanisms on a database, instead of just one. In particular under the context of PATE, where a machine learning model is queried repeatedly, it is of interest to understand the total privacy leak of the model after  $k$  queries.

The following composition theorem is a tool that allows for the computation of a total privacy bound, given the  $(\varepsilon_i, \delta_i)$  leak of individual mechanisms.

### General Composition Theorem

Let  $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \rightarrow \text{Range}(\mathcal{M}_i)$  be an  $(\varepsilon_i, \delta_i)$ -differentially private algorithm for  $i \in [k]$ . If  $\mathcal{M}_{[k]} : \mathbb{N}^{|\mathcal{X}|} \rightarrow \prod_{i=1}^k \text{Range}(\mathcal{M}_i)$  is defined to be  $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \dots, \mathcal{M}_k(x))$ , then  $\mathcal{M}_{[k]}$  is  $(\sum_{i=1}^k \varepsilon_i, \sum_{i=1}^k \delta_i)$ -differentially private.

What this theorem tells is that the composition of two differentially private algorithms  $A$  and  $B$  is itself a differentially private algorithm  $C$ , and that by knowing the privacy leak of  $A$  and  $B$ , the one of  $C$  can be computed. This will be of value when performing  $N$  identical queries on a database, given that knowing the individual privacy leak of one of them would allow for the computation of the overall privacy leak.

## 2.2 DP-SGD

Differential privacy provides a robust framework with which to measure the privacy guarantees provided by an algorithm that queries a database. In order to utilize such tools under the context of machine learning algorithms, there are multiple approaches that allow for the training of models with differential privacy guarantees. One of them is DP-SGD [10], a method that proposes a modification to the stochastic gradient descent algorithm so that models have provable privacy guarantees, expressed in terms of differential privacy.

Bear in mind that the traditional mini-batch stochastic gradient descent algorithm can be described as follows:

---

**Algorithm 1:** Mini-batch Stochastic Gradient Descent

---

**Input** : Learning rate  $\alpha$ , model parameters  $\theta$ , number of iterations  $m$ ,  
loss function  $L(\theta(x), y)$

**Output:** Model parameters  $\theta$

- 1 Initialize  $\theta$  randomly.;
  - 2 **for**  $i = 1$  to  $m$  **do**
  - 3     Sample a minibatch of training samples  $(x, y) \in S$  ;
  - 4      $\theta := \theta - \alpha \nabla_{\theta} L(\theta(x), y)$
- 

To ensure that stochastic gradient descent is a differentially private algorithm, two points need to be addressed [10]:

1. The sensitivity of each gradient should be bounded, to limit the impact each individual training point has on the gradient computation.
2. The behaviour of the algorithm needs to be random, to obfuscate whether a particular sample was included in the training set.

Clipping gradients computed on each training sample allows for limiting the impact each training point has on the model parameters. This guarantees that each gradient has a known maximum Euclidean norm. Afterwards, sampling random noise from a given distribution and adding it to the clipped gradients addresses the second point.

The resulting training algorithm with the modifications described above is as follows [10]:

---

**Algorithm 2:** Mini-batch DP-Stochastic Gradient Descent

---

**Input** : Learning rate  $\alpha$ , model parameters  $\theta$ , number of iterations  $m$ ,  
loss function  $L(\theta(x), y)$

**Output:** Model parameters  $\theta$

```

1 Initialize  $\theta$  randomly.;
2 for  $i = 1$  to  $m$  do
3   | Sample a minibatch of training samples  $(x, y) \in S$  ;
4   | gradient =  $\nabla_{\theta} L(\theta(x), y)$  ;
5   | gradient = clip(gradient) ;
6   | gradient = noise + gradient ;
7   |  $\theta := \theta - \alpha * \text{gradient}$  ;

```

---

## Suitability for this work’s problem

On the setting described where malicious web requests need to be detected, data is distributed across different organizations. Since it is desirable that models are trained in a distributed manner, where each stakeholder has their own data, the approach described below on Section 2.3 is better suited than DP-SGD.

PATE also has the benefit of offering a privacy loss that’s independent of the number of training epochs, whereas the privacy loss scales linearly with DP-SGD as each new minibatch step queries the training data and contributes to the overall privacy leak as described by the general composition theorem [2.1].

## 2.3 Private Aggregation of Teacher Ensembles (PATE)

In [1], Papernot et al. introduce a technique to train supervised machine learning models of arbitrary architecture in a differentially private manner, by employing an ensemble of teacher models that provide labels to a student model. The mechanism through which these teachers transfer knowledge to the student is constrained in such a way that the privacy loss incurred during student training can be computed and bounded. This student model can then be used to do inference on new data, without causing additional privacy loss thanks to the post-processing property of differentially private algorithms.

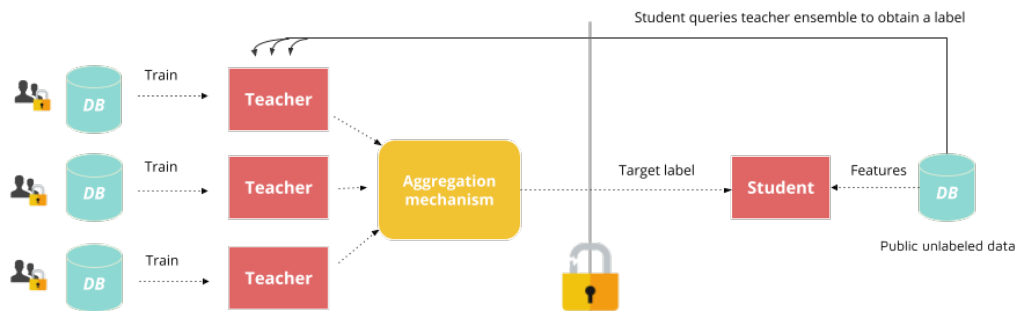


Figure 2.1: PATE training algorithm

Each teacher is trained on a disjoint subset of the training data and then used to generate class predictions of the student training data. These teacher predictions are then aggregated according to an aggregation mechanism, and the result becomes the label used for training the student model.

Whereas DP-SGD requires modifications to the training algorithm and is less suited under scenarios where data is distributed, the aggregation mechanism of

PATE makes more assumptions about the machine learning task to perform. This is due to the fact that an aggregation strategy needs to be devised that not only summarizes the teacher predictions effectively, but allows for the accounting of privacy loss incurred during student training.

To illustrate these two algorithm design requirements, consider the machine learning problem of classification, where a single class is expected to be returned by the algorithm. In order to implement an aggregation mechanism for PATE on such a problem, a way to average the class predictions of all teacher models needs to be devised, as well as a method for computing the privacy loss that is incurred during this averaging. The laplacian aggregation mechanism is an example of such algorithm; in [2.3.1] and in [3] we present how these two points are addressed.

### 2.3.1 Laplacian Aggregation Mechanism

The aggregation mechanism employed in this work consists of adding noise sampled from a Laplacian distribution to the teachers' class prediction count. For a given student training sample  $x$ , given the label count of teacher predictions  $n_j$  for each possible class  $j$ , one can define the aggregation mechanism as follows:

$$f(x) = \arg \max_j \left\{ n_j(x) + \text{Lap}\left(\frac{1}{\gamma}\right) \right\}$$

Where  $\text{Lap}\left(\frac{1}{\gamma}\right)$  is a real number sampled from a Laplacian distribution with location 0 and scale  $\frac{1}{\gamma}$ . The parameter  $\gamma$  influences the amount of noise added to the query, and therefore the privacy guarantee that can be provided.

To illustrate this aggregation mechanism, consider a binary classification problem where 10 teachers are used to label student samples. The 10 predictions are then counted resulting in a vote count of 8 positive votes and 2 negative votes. Each one of these vote counts is then added random noise (real numbers A and B) which will be sampled from a Laplacian distribution, and the most voted class after the noise addition — in this case still the positive class, will be used as the student target label.

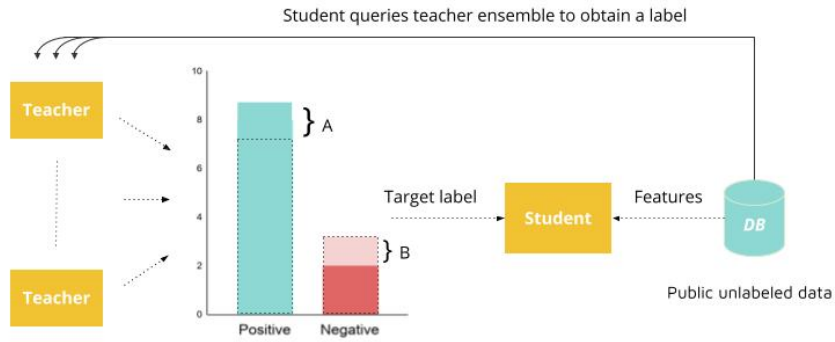


Figure 2.2: Diagram of an aggregation mechanism of the PATE algorithm

By keeping track of the aggregated labels and each teacher prediction, the privacy loss incurred during the generation of these synthetic labels can be computed as explained in chapter 3.



## 2.4 Distributed Learning

The proposed solution for malicious web request detection might be deployed in a distributed network, should these research efforts show promising results. If this is the case, a related area of research known as distributed learning might be relevant, as it enables the use of multiple devices to perform the training of large neural networks.

As the training data required for novel applications increases in orders of magnitude, so do the I/O and compute requirements of the systems used to train models. Given that demand for processing data has increased at a faster pace than the growth in computing power, combined with the fact that data is often found in a distributed fashion or is too big to store on a single device, the use of distributed systems for training such models has become commonplace [11].

Distributed learning encompasses the collection of techniques that allow to distribute the workload required for the training of models across multiple devices, called worker nodes.

Although every distributed training algorithm has unique properties that make the communication patterns used between multiple nodes different, there are two architecture patterns that describe most distributed systems:

- Data parallelism: the data is split into different nodes, and the same training algorithm is applied on each node to a subset of the data. The same model is available to all nodes, which is either synchronized across all nodes or centralized. This strategy works for all machine learning algorithms.
- Model parallelism: the model is split into different nodes, and the data is propagated through them sequentially. Whether this technique can be applied depends on the model architecture - some models don't have a natural way in which they can be split. An example of applying model parallelism to a neural network consists of putting groups of layers on different devices: in the forward pass, a node passes the activation of the last layer to the next node; on the backward pass the gradients are sent to the previous nodes.

In this work, a data parallelism approach was taken to train distributedly the teacher ensemble.

### 3 Analyzing the privacy loss in PATE

Differential privacy provides a strong standard against which to measure the privacy loss incurred during training of the student model. This framework allows the specification of privacy guarantees that a query that analyzes a private database can offer; in the context of machine learning, the query could be considered doing inference with a trained model on new, unseen data.

The model has acquired knowledge from the dataset with sensible information, and each time it is queried a certain amount of privacy loss is incurred. Given that the privacy loss is compounded across all queries done according to the composition mechanisms described in [2], it would be desirable to have a way in which to bound the total amount of privacy loss  $(\epsilon, \delta)$  incurred during the training of the model, so that it can then be used for inference an arbitrary amount of times, with guarantees that the total privacy loss will be bounded by  $(\epsilon, \delta)$ .

Indeed, this is what PATE proposes to do when training the student model by querying the teacher ensemble: as knowledge is transferred from the ensemble to the student, the privacy loss incurred is tracked. This chapter will describe the way in which this bound on the privacy loss can be computed.

Let  $\mathcal{M}$  be a randomized algorithm with domain  $D$  and range  $R$ ,  $d, d' \in D$ , and  $o \in R$ . We define a random variable  $c$  such that:

$$c(o) \triangleq \log \frac{P[\mathcal{M}(d) = o]}{P[\mathcal{M}(d') = o]} \tag{3.1}$$

with  $o \sim \mathcal{M}(d)$ .

Now, in order to study the privacy loss of PATE, it is convenient to rephrase the definition of differential privacy given in Eq. 2.1 into an equivalent formulation as follows. A randomized algorithm  $\mathcal{M}$  is  $(\epsilon, \delta)$ -differentially private if for any adjacent databases  $d, d' \in D$  and output  $o \in R$ :

$$P_{o \sim \mathcal{M}(d)}[c(o) > \epsilon] \leq \delta \tag{3.2}$$

The above formulation enables studying the privacy loss of the PATE mechanism by resorting to a series of properties from probability theory in order to derive the tail bound of random variable  $c$ . For such, we need to recall a few definitions and theorems.

### Moment of a random variable

The  $n$ -th moment of a random variable  $X$  is  $\mathbb{E}[X^n]$ . In particular, the first moment is the mean,  $\mu_X = \mathbb{E}(X)$ . The  $n$ -th central moment of a random variable  $X$  is  $\mathbb{E}[(X - \mathbb{E}[X])^n]$ . The variance is the second central moment  $\mathbb{E}[(X - \mathbb{E}[X])^2]$ .

### Moment generating function

An alternative specification of a random variable probability distribution is its moment generating function, which is defined as  $M_X(\lambda) = \mathbb{E}[e^{\lambda X}]$  for all  $\lambda \in \mathbb{R}$  where it is finite, which includes at least  $\lambda = 0$ .

$M_X(\lambda)$  can be used to compute the  $n$ -th moment of  $X$ : it is the  $n$ -th derivative of  $M_X(\lambda)$  evaluated at 0.

### Chernoff-Cr amer bound

Let  $X$  be a centered random variable such that  $M_X(\lambda) < +\infty$  on  $\lambda \in (-\lambda_0, \lambda_0)$  for some  $\lambda_0 > 0$ . Then, for any  $\beta > 0$ :

$$P[X \geq \beta] \leq e^{-\psi_X^*(\beta)}$$

where  $\psi_X^*(\beta) = \sup_{\lambda \in \mathbb{R}_+} (\lambda\beta - \psi_X(\lambda))$  and  $\psi_X(\lambda) = \log M_X(\lambda)$ . This inequality is usually expressed as

$$P[X \geq \beta] \leq e^{\psi_X(\lambda) - \lambda\beta} \tag{3.3}$$

Now, in order to analyze the privacy loss of PATE we proceed as follows. Let's start by defining:

$$\alpha_{\mathcal{M}}(\lambda) \triangleq \max_{d, d'} \log \mathbb{E}_{o \sim \mathcal{M}(d)} [e^{\lambda c(o)}] \tag{3.4}$$

Therefore, for the random variable  $c$  defined in Eq. 3.1, it holds:

$$\psi_c(\lambda) \leq \alpha_{\mathcal{M}}(\lambda) \tag{3.5}$$

Taking  $\beta$  to be  $\varepsilon$  in Eq. 3.3, it follows:

$$P_{o \sim \mathcal{M}(d)}[c(o) \geq \varepsilon] \leq e^{\alpha_{\mathcal{M}}(\lambda) - \lambda \varepsilon} \quad (3.6)$$

Thus, a mechanism  $\mathcal{M}$  would be  $(\varepsilon, \delta)$ -differentially private for every  $\varepsilon, \delta$  satisfying:

$$e^{\alpha_{\mathcal{M}}(\lambda) - \lambda \varepsilon} \leq \delta$$

With a little rewriting, we have:

$$\varepsilon \geq \frac{1}{\lambda} \left( \alpha_{\mathcal{M}}(\lambda) - \log \delta \right)$$

Hence, for a given  $\delta$ , we can obtain the *minimum* bound of the privacy loss which could be ensured with such  $\delta$ :

$$\varepsilon = \min_{\lambda} \frac{1}{\lambda} \left( \alpha_{\mathcal{M}}(\lambda) - \log \delta \right) \quad (3.7)$$

Provided  $\alpha_{\mathcal{M}}(\lambda)$  can be computed or bounded, Eq. 3.7 gives a means for bounding the privacy loss of a mechanism with a given confidence. We apply it in the following sections to analyze the privacy loss of PATE with a Laplacian aggregation mechanism.

## 3.1 Data independent privacy analysis

Consider adjacent databases  $d$  and  $d'$  where label counts  $n_j$  differ by at most 1 in each coordinate. Let  $\mathcal{M}$  be the mechanism that reports  $\arg \max_j \{n_j + \text{Lap}(\frac{1}{\gamma})\}$ . Then,  $\mathcal{M}$  satisfies  $(2\gamma, 0)$ -differential privacy. It can be shown that, for any  $\lambda$  [1]:

$$\alpha_{\mathcal{M}}(\lambda) \leq 2\gamma^2 \lambda(\lambda + 1) \quad (3.8)$$

For an aggregation mechanism with noise  $\text{Lap}(\frac{1}{\gamma})$  which is  $(2\gamma, 0)$ -differentially private, this inequality can be used together with the previous properties to bound the privacy loss over  $T$  queries to the teacher ensemble to  $(4T\gamma^2 + 2\gamma\sqrt{2T \ln \frac{1}{\delta}}, \delta)$ -differential privacy [1].

## 3.2 Data dependent privacy analysis

The bound on the privacy loss obtained above could be made smaller provided we bring into the picture the actual predictions delivered by the ensemble of teachers.

Indeed, a smaller  $\alpha_{\mathcal{M}}$  could be computed if we take into account the fact that when quorum among teachers is strong, the majority outcome has overwhelming likelihood, so the privacy cost is smaller when this outcome occurs. The following theorem, proved in [1], provides a *data-dependent* bound on  $\alpha_{\mathcal{M}}$ :

**Theorem:** Let  $\mathcal{M}$  be  $(2\gamma, 0)$ -differentially private and  $q \geq P[\mathcal{M}(d) \neq o^*]$  where  $o^*$  is the most probable outcome as determined by the teacher ensemble predictions. Let  $\lambda, \gamma \geq 0$  and  $q < \frac{e^{2\gamma}-1}{e^{4\gamma}-1}$ . Then,

$$\alpha_{\mathcal{M}}(\lambda) \leq \log \left( (1-q) \left( \frac{1-q}{1-e^{2\gamma}q} \right)^\lambda + qe^{2\gamma\lambda} \right) \quad (3.9)$$

In order to upper bound  $q$  for our aggregation mechanism, the following Lemma is also proved in [1].

**Lemma:** Let  $n$  be label score vector produced by the teacher ensemble, for a database  $d$  with  $n_{o^*} \geq n_o$  for all  $o$ . Then:

$$P[\mathcal{M}(d) \neq o^*] \leq \sum_{o \neq o^*} \frac{2 + \gamma(n_{o^*} - n_o)}{4e^{\gamma(n_{o^*} - n_o)}}$$

Thanks to this bound on  $q$  that depends on the teacher agreement,  $\alpha_{\mathcal{M}}$  can be bounded for a specific response of the ensemble to a query of the student. This allows computing a more accurate (data-dependent) bound on the actual privacy loss incurred by PATE for a *given* set of queries of the student.

# 4 Experimental results

After validating the experimental results achieved in [1] by replicating the experiments done with the MNIST[12] and SVHN[13] datasets and achieving comparable results, the PATE framework was applied to the problem of malicious web request detection. On this chapter, the results of that work will be presented, along with proposed improvements on the technique to ensure the privacy of the training data of the student model.

## 4.1 Experimental Setup

In order to classify web requests, a dataset of 651.602 labeled requests was assembled. To construct the features, the URI of a web request is tokenized and a bag-of-words approach is used to compute the uni-grams present in each request. The term frequency-inverse document frequency (TF-IDF) of the 500 most frequent tokens across the entire dataset is then computed for each sample. These tokens include terms such as “javascript”, “php”, “sfish”, “?”, “login”, among others.

**TF-IDF:** The term frequency-inverse document frequency is a statistical measure that represents how important a term is to a document, with respect to the dataset in which that document is found. In this work, a term is a token and a document represents a single web request sample.

The relative importance of a term depends not only on the frequency with which it appears in the document, but also on the inverse of the frequency with which the term appears on the entire dataset. This approach helps to reduce the importance of frequent terms that don’t add much information, such as ‘the’ or ‘a’ in the context of the English language. For a particular term  $t$  found on document  $d$  in the dataset  $D$ , the TF-IDF[14] is defined as follows:

$$TF - IDF(t, d, D) = TF(t, d) \cdot IDF(t, D) \quad (4.1)$$

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}} \quad (4.2)$$

$$IDF(t) = \frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in it}} \quad (4.3)$$

Once the dataset is constructed using the strategy described above, it is split as follows:

- 233518 samples for the training and evaluation of the teacher models.
- 800 samples per teacher for calculating the optimum threshold for considering the classifier’s output as a positive prediction. This point will be explained further below.
- Between 500 and 1000 samples for the training of the student model, varying per experiment.
- 5000 samples for the validation step of the student model.
- 200 samples for calculating the optimal threshold of the student model
- Remainder samples for test set.

Following the same strategy as the original paper [1], ensembles of 100 and 250 teacher models were trained and used to generate labels for the student training samples, using the laplacian aggregation mechanism.

Given the unbalanced distribution of the training set - 95% of samples are not malicious, a threshold of 0.5 that would split the model’s output between positive and negative samples would yield sub-optimal results. Therefore, the receiver operating characteristic curve is calculated for a subset of samples, and the threshold that maximizes the difference between the true positive rate and false positive rate is picked as the optimum one.

**Receiver operating characteristic curve:** The ROC curve[15] plots the true positive rate with respect to the true negative rate at different classification thresholds, and is a tool that allows to measure the performance of a classifier at these different thresholds. Consider that a classifier outputs the probability of a sample having a positive class as 0.6. A standard threshold of 0.5 would determine that such sample has a positive class, whereas a higher threshold of 0.8 would indicate that the same sample has a negative class.

True Positives = Positive cases correctly classified

True Negatives = Negative cases correctly classified

$$TPR = \frac{\text{True Positives}}{\# \text{ Positive Cases}}$$

$$TNR = \frac{\text{True Negatives}}{\# \text{ Negative Cases}}$$

Increasing the threshold results in more items being classified as negative, therefore increasing both the true negatives and the false negatives. By calculating the ROC curve, it is possible to pick a threshold that maximizes the difference between the true positive rate and the false positive rate, which is the metric of interest in this work.

A simple fully connected neural network architecture was used for both the teacher and student models, with a single real-valued output:

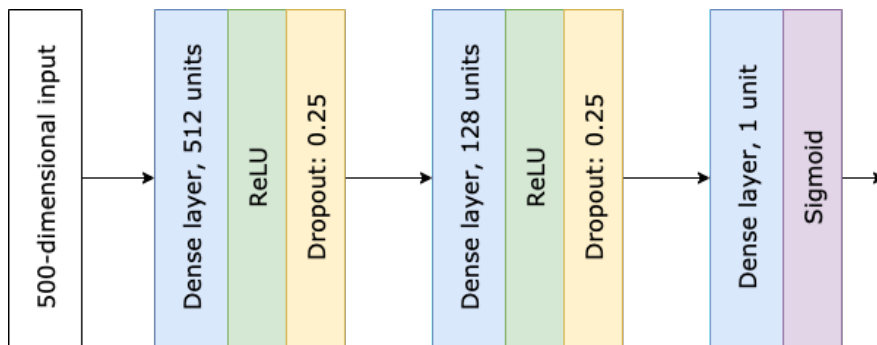


Figure 4.1: Neural network architecture



## 4.2 Results

On all experiments, an aggregation mechanism was applied where noise sampled from a laplacian distribution with  $\gamma = 0.05$  was applied to the label counts of all teacher predictions. The data-dependent  $\varepsilon$  recorded was obtained along  $\delta = 10^{-5}$ .

# of Teachers	# of student samples	TPR	TNR	$\varepsilon$
100	1000	81.2%	94.5%	5.32
250	1000	84.2%	93.5%	0.39

Table 4.1: Privacy cost and precision metrics obtained

An increase in the number of teacher models used allows for higher levels of privacy guarantees. One should note, however, the context in which this system would be deployed: a government’s intranet which has multiple departments and organizations, each with sensible data on which a separate model could be trained. Under this distributed setting, it would be reasonable to expect that, depending on the volume of data per organization, one or a couple of models per organization could be trained. The feasibility of training 250 or more teacher models will depend in large part on the volume of web requests collected and labeled within each organization.

### 4.2.1 Privatization of student data

Under the traditional PATE architecture, the student sends its samples to the teacher ensemble for the labels to be generated. Under the distributed setting that’s proposed in this work, the student might belong to a third party organization which might not necessarily trust the system that hosts the teacher ensemble and aggregation mechanism.

It is reasonable to assume that the student itself might expect certain privacy guarantees for the data it is sharing with the teacher ensemble, and for that purpose this work proposes the addition of random noise to the feature representation of the student data, before its sent to the ensemble. Recall that the feature representation chosen for this problem is a bag of words, which doesn’t directly reveal what the sequence of tokens actually is, as that sequence is broken. However, the set of bag of words representations of the entire dataset can be interpreted as a probability distribution of the frequency of occurrence of each token. By sampling from this probability distribution, likely requests could be reconstructed, and privacy lost.

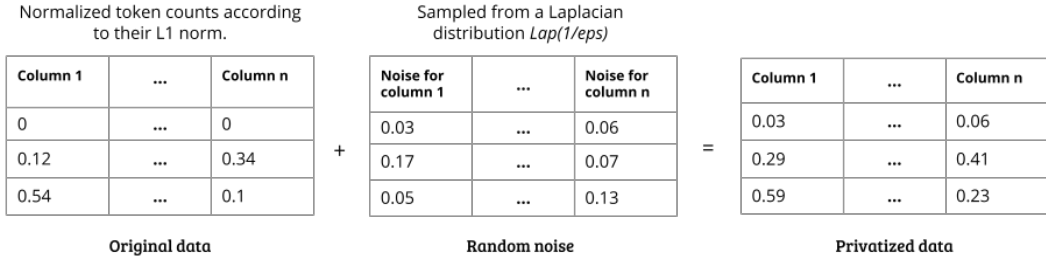


Figure 4.2: Visualization of the privatization process applied to the student data

Figure 4.2.1 illustrates how a noise vector sampled from a Laplacian distribution with scale  $\epsilon$  and mean 0 is added to the feature representation of each request before being sent to the Aggregator. This follows the definition of a Laplacian Mechanism, where given a function  $f : \mathbb{N}^{|X|} \rightarrow \mathbb{R}^k$ , a Laplacian Mechanism is:

$$\mathcal{M}_L(x, f(\cdot), \epsilon) = f(x) + (Y_1, \dots, Y_k)$$

In our case,  $f$  would be the identity function, and as proved in [2], this mechanism preserves  $\epsilon$ -differential privacy. Once the Aggregator returns a label, the original feature representation can be used along it to train the student.

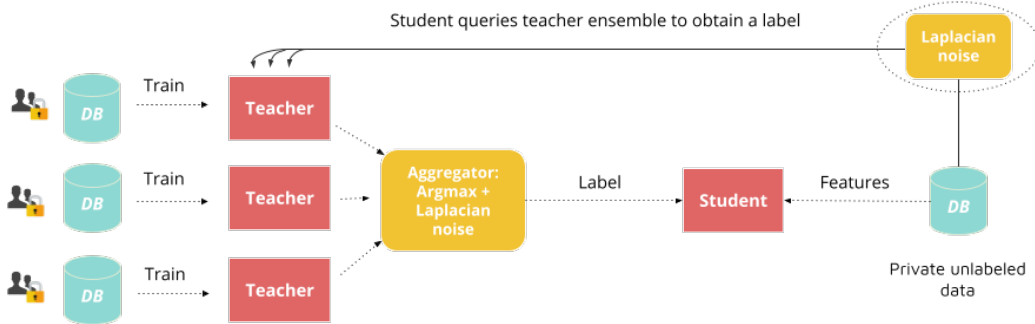


Figure 4.3: Visualization of the PATE training algorithm with the proposed improvement for privatizing the student data

For a given student  $\varepsilon$ , ten student models were trained by sending the samples to the same teacher ensemble ten times, each with noise sampled from a Laplacian distribution with scale  $\varepsilon$ , and then trained as described by the PATE algorithm. The privacy cost with respect to the teacher’s training data is computed as described in Chapter 3 for  $\delta = 0.05$ . The following table details the achieved teacher and student  $\varepsilon$ , and the true positive and negative rates obtained by the classifiers.

# of Teachers	Student $\varepsilon$	Teacher $\varepsilon$	TPR	TNR
100	0.001	9.18	0.82	0.95
100	0.01	10.56	0.83	0.94
100	0.1	13.24	0.76	0.97
100	0.5	23.51	0.84	0.94
100	1	23.51	0.82	0.96
100	5	23.51	0.85	0.93
100	10	23.51	0.82	0.95
100	50	8.18	0.82	0.94
100	100	7.06	0.84	0.94
100	no noise	5.32	0.81	0.94
250	0.001	0.52	0.85	0.95
250	0.01	0.58	0.80	0.95
250	0.1	1.00	0.87	0.93
250	0.5	6.30	0.86	0.93
250	1	23.51	0.84	0.94
250	5	23.51	0.84	0.93
250	10	7.20	0.91	0.85
250	50	0.47	0.84	0.92
250	100	0.40	0.81	0.94
250	no noise	0.40	0.84	0.94

Table 4.2: Relationship between student  $\varepsilon$ , teacher  $\varepsilon$ , and true positive and negative rates achieved for different teacher ensemble sizes and levels of noise added to student data.

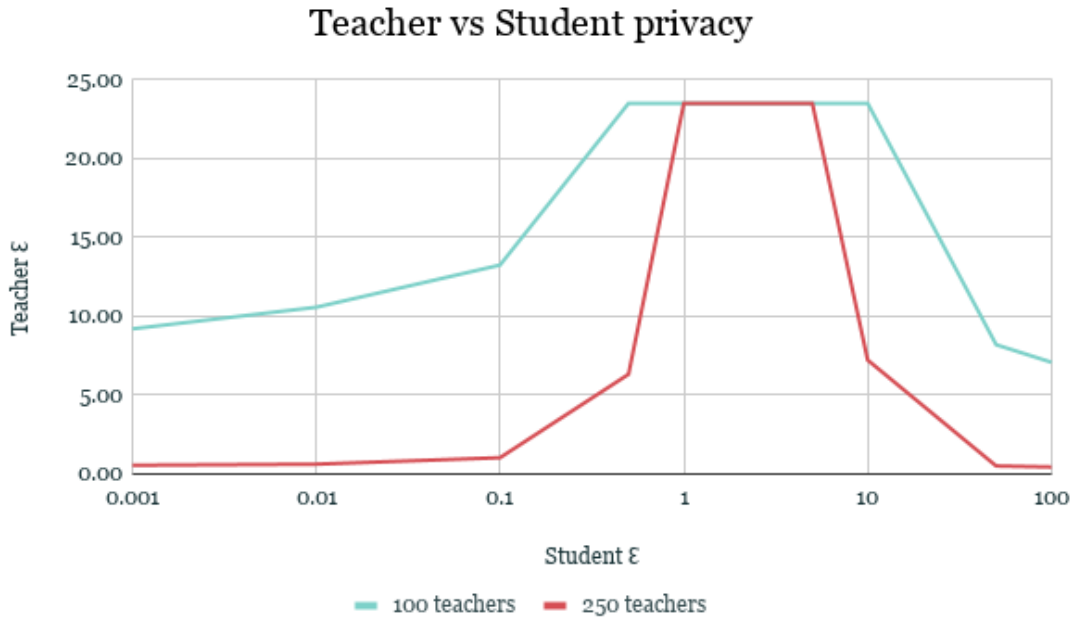


Figure 4.4: Relationship between the student and teacher  $\epsilon$ .

An interesting phenomenon observed is the relationship between the noise applied to the student data, and privacy leaked by the teacher’s dataset, measured as data-dependent privacy loss: as the noise applied to the student’s data increases, so does the privacy leak of the teacher ensemble. This can be attributed to the fact that as the data sent to the ensemble becomes noisier, it becomes harder for the teachers to predict the correct label, therefore the consensus among them decreases, which results in a higher data-dependent  $\epsilon$ .

For teacher ensembles of 250 models and student  $\epsilon$  of 0.1, the student model achieves true positive and negative rates above 87%, with a small privacy leak. This validates the hypothesis that robust privacy guarantees can be ensured for such a problem, provided the data can be partitioned in a sufficiently large ensemble of teacher models.

# 5 Software engineering efforts conducted

Several strategies were utilized to conduct the series of experiments required in a maintainable manner, that allows for code reuse and to manage the complexity of the result tracking of the experiments.

## 5.1 Modelling of the solution

Although there are a couple of open source implementations of PATE, it was decided to implement the framework from scratch so as to facilitate the extensibility of the tool. To do so, the problem was modelled as detailed in the UML diagram 5.1, where both teacher and student models inherit from the same class that's responsible for the performance of the training of the model.

Since the network architecture might change from problem to problem, that responsibility is delegated to the abstract class `Network`, which allows for multiple architectures to be used, such as convolutional neural networks for computer vision problems, and a fully connected neural network for the problem of malicious web request detection.

In order to perform the aggregation of the teacher models predictions, an abstract class `Aggregator` is implemented by `LaplaceAggregator` to perform the Laplacian Aggregation Mechanism proposed in [1], and `GaussianAggregator` for the Gaussian Aggregation Mechanism proposed in [16]. `Dataset` provides utility functions to perform the data splits between teachers' and students' training, validation and test sets, besides the preprocessing and data loading.

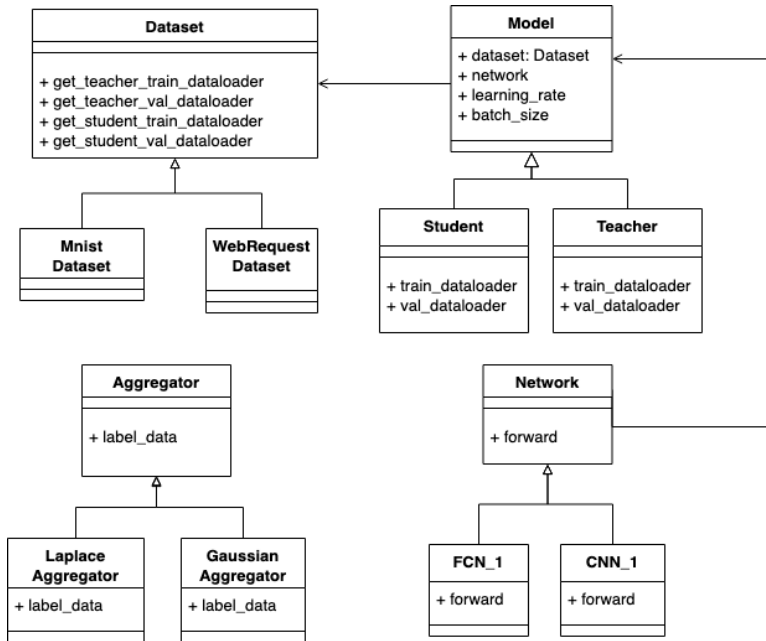


Figure 5.1: UML diagram of the solution

Given that the neural networks chosen proved to be capable of achieving high levels of precision, it was decided to fix the model architecture in order to explore how other parameters, such as the added noise, affected the model performance.

The solution was developed in Python, using the Pytorch machine learning library [17]. There are three scripts, `train_teachers.py`, `aggregate_labels.py` and `train_student.py`, that describe the full interaction between the abstractions to perform the complete training of teacher and student models. This modular approach allows for the experimentation of multiple aggregation strategies once the teacher models are trained, and for multiple different students to be trained once the aggregated labels are computed. For the sake of completeness, the entire behaviour of these three scripts is detailed on the sequence diagram 5.2

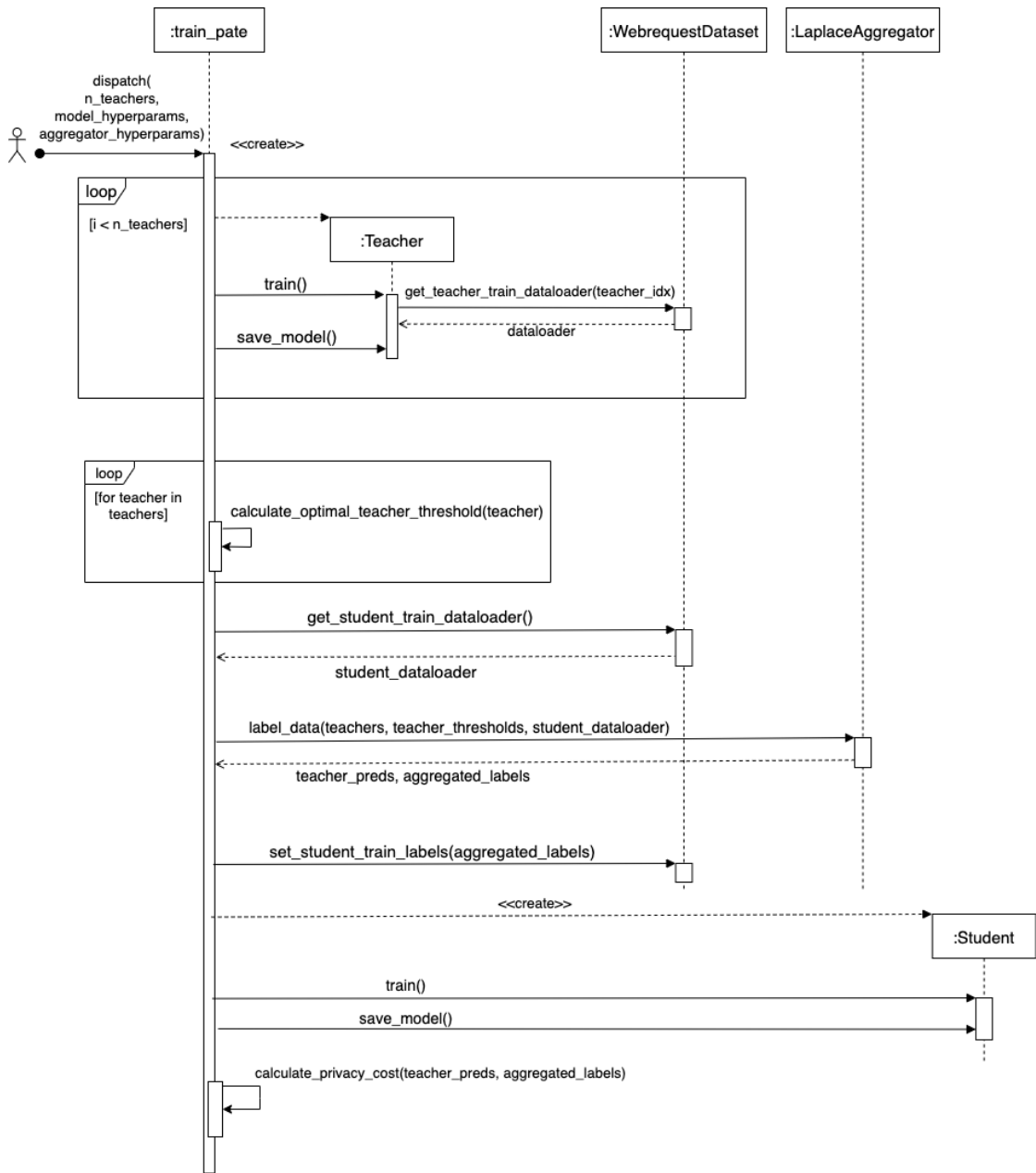


Figure 5.2: Sequence diagram that describes the interaction of the different abstractions

## 5.2 Experiment management

Throughout the entire experimentation period, more than 2500 experiments were conducted, and the metrics of the resulting models logged. In order to manage the complexity of this task, and enable the quick comparison of different experiments, an automatic logging and visualization tool was integrated with the solution: Weights and Biases [18].

By logging the experiment metadata, training and evaluation metrics and uploading them to a server, the tool allows for the sorting, filtering, tagging, comparison and visualization of multiple experiments simultaneously.

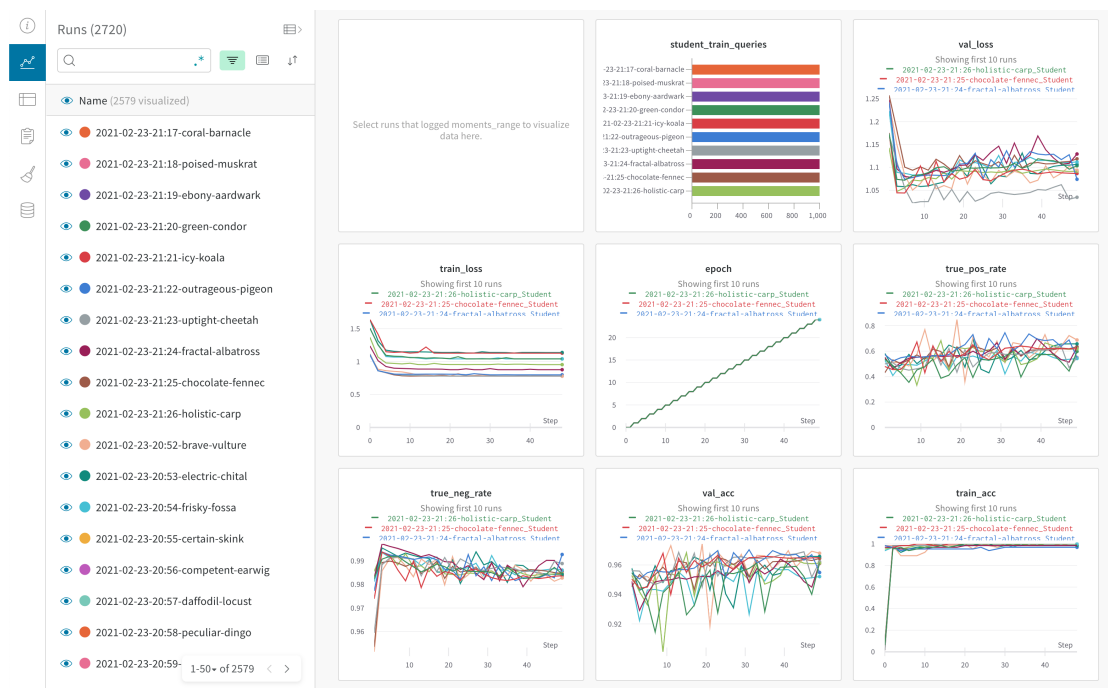


Figure 5.3: Interface of the tool Weights & Biases, which allows to compare metrics across experiments



## 5.3 Multi-GPU training

Training large deep neural networks on big datasets often supposes challenges with respect to the computational resources needed in order to train within reasonable time constraints. Even though our proposed architecture proposed in [4.1] has only 322.000 trainable parameters, training 250 of these models for the teacher ensemble does represent a significant challenge, when iterating on various experiments.

The training of these models was performed on Universidad ORT’s CECOFI cluster, on 4 NVIDIA GeForce RTX 2080 GPUs and taking advantage of a data splitting strategy across multiple GPUs known as `DataParallel`. Given  $n$  GPU nodes, the algorithm works as follows:

---

**Algorithm 3:** DataParallel algorithm for multi-GPU training [19]

---

- 1 Duplicate model onto the available GPUs;
  - 2 Split the batch into  $n$  subsets, and move each to a different GPU;
  - 3 Perform the forward pass in a distributed manner;
  - 4 Compute losses, aggregate them on main GPU, and send the loss back to each node;
  - 5 Compute gradients on each node;
  - 6 Sum up the gradients on the main GPU, update the model, and send copies back to other GPUs for the next iteration;
-

## 6 Conclusions and lessons learned

In this work the PATE framework was studied, and its claimed efficacy was tested by training differentially private models on reference datasets. Once the results claimed on [1] were validated, the technique was applied to the problem of malicious web requests detection, achieving both high levels of privacy and precision at the classification task.

The problem presented a series of interesting challenges, from the engineering aspects of managing the training of large ensemble of models, to the design choices needed to be taken to achieve promising results for the particular problem addressed.

A key design consideration that the solution needed to address was the extensibility of the tool developed, so that if promising results were achieved for this particular problem, it would be easy to retrain new models with different architectures, for related problems on new datasets. The aggregation mechanism should be replaceable, and the training might happen in a distributed fashion. These requirements motivated our desire to develop the solution in a modular fashion, with an emphasis on facilitating the tracking of all the experiments conducted along the way. I consider that this focus on implementing good engineering practices along a machine learning project helped me grow as a software engineer.

When iterating on the chosen model so as to achieve the best possible performance, a couple of challenges were identified. Firstly, the dataset was unbalanced with respect to the labels, which made our model more sensible to negative classes than positive ones. By plotting the ROC curve and choosing different thresholds that split our model probability's output between a positive and negative prediction, a better balance between true positive rates and true negative rates was achieved. A hyperparameter random search was used to find optimal model hyperparameters, and early stopping applied to avoid overfitting the training datasets.

Along the development of the project, two opportunities arised to present our

work: on a technical meeting organized by Tryolabs, a machine learning consulting company, and a privacy and data anonymization workshop organized by ICT4V. The focus on the first talk was to share knowledge on Differential Privacy and how the PATE framework can help achieve it, whereas on the second one our results applying this technique on the problem of malicious web requests detection were also presented. Besides giving visibility to our work and this field of study, these events also helped me to develop my communication and presentation skills, for which I'm grateful to have participated on them.

## 7 Bibliographic references

- [1] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” *arXiv preprint arXiv:1610.05755*, 2016.
- [2] C. Dwork, A. Roth *et al.*, “The algorithmic foundations of differential privacy.” *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.
- [3] A. Ng, *Machine Learning Yearning*. USA: Kobo, 2017.
- [4] A. Harmanci and M. Gerstein, “Quantification of private information leakage from phenotype-genotype data: linking attacks,” *Nature methods*, vol. 13, no. 3, pp. 251–256, 2016.
- [5] A. Narayanan and V. Shmatikov, “Robust de-anonymization of large sparse datasets,” in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 111–125.
- [6] L. Sweeney, A. Abu, and J. Winn, “Identifying participants in the personal genome project by name (a re-identification experiment),” *arXiv preprint arXiv:1304.7605*, 2013.
- [7] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, “Unique in the crowd: The privacy bounds of human mobility,” *Scientific reports*, vol. 3, no. 1, pp. 1–5, 2013.
- [8] T. R. Fredrikson, Somesh Jha, “Model inversion attacks that exploit confidence information and basic countermeasures,” 10 2015.
- [9] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, “The secret sharer: Evaluating and testing unintended memorization in neural networks,” in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 267–284.

- [10] M. Abadi, A. Chu, I. Goodfellow, B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *23rd ACM Conference on Computer and Communications Security (ACM CCS)*, 2016, pp. 308–318. [Online]. Available: <https://arxiv.org/abs/1607.00133>
- [11] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, “A survey on distributed machine learning,” Mar. 2020. [Online]. Available: <https://doi.org/10.1145/3377454>
- [12] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [13] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: [http://ufldl.stanford.edu/housenumbers/nips2011\\_housenumbers.pdf](http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf)
- [14] L.-P. Jing, H.-K. Huang, and H.-B. Shi, “Improved feature selection approach tfidf in text mining,” in *Proceedings. International Conference on Machine Learning and Cybernetics*, vol. 2, 2002, pp. 944–946 vol.2.
- [15] K. H. Zou, A. J. O’Malley, and L. Mauri, “Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models,” *Circulation*, vol. 115, no. 5, pp. 654–657, 2007.
- [16] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingson, “Scalable private learning with pate,” *arXiv preprint arXiv:1802.08908*, 2018.
- [17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [18] L. Biewald, “Experiment tracking with weights and biases,” 2020, software available from wandb.com. [Online]. Available: <https://www.wandb.com/>

- [19] W. D. Hillis and G. L. Steele Jr, “Data parallel algorithms,” *Communications of the ACM*, vol. 29, no. 12, pp. 1170–1183, 1986.