Privacy Preserving Inference via Model Extraction Against Third-Party NLP Cloud Adversaries

by

Sarina Xi

Supervisor: David Lie
April 2024

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

The widespread use of MLaaS cloud applications is accompanied by increasing data privacy risks, where third-party cloud adversaries can potentially exploit and abuse sensitive user data. However, many data privacy schemes sacrifice model data utility and performance efficiency in exchange for privacy. This thesis presents and investigates a privacy-preserving inference infrastructure that uses model extraction methods to maintain model performance while preserving privacy. In particular, we take inspiration from knowledge distillation and active learning techniques to build our framework and test it in the NLP domain. We compare and evaluate different means to preserve privacy in the infrastructure and perform analyses of their effectiveness. We show that employing a public dataset in our model extraction inspired inference infrastructure provides strong privacy while maintaining model performance, outperforming state-of-the-art methods such as using differential privacy noise.

Acknowledgement

I would like to thank Professor David Lie for his guidance and help in the past year. I am grateful for this opportunity to deepen my knowledge in ML privacy and explore this field. Furthermore, I would like to thank Cassie Li, who is working on the image domain of this inference infrastructure, for her support and guidance, not only as a graduate student mentor, but also as a friend. Finally, I would like to thank my friends and family for the encouragement they have given me during this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With an increasingly data-driven society, resource-intensive Machine Learning as a Service (MLaaS) applications are often too computationally expensive to run on local devices [1, 2]. As a result, many MLaaS run on cloud computing platforms such as Amazon Web Services, Microsoft Azure, Google Cloud, and more [1, 3]. However, the use of third-party cloud services increase the risk of data abuse and exploitation [4, 5]. In the natural language processing (NLP) domain, text sequences are collected and sent to the cloud to execute the desired language processing task. During this process, third-party adversaries can unknowingly collect personal data for monetization or conduct other harmful activities, such as identity theft or discrimination [6]. Therefore, data protection during inference against adversarial third-party cloud services is crucial, especially in fields that utilize ML to analyze sensitive or proprietary information, such as healthcare [7] and law enforcement [8]. Recently, the importance of such data privacy preserving inference infrastructures is further reinforced by the rapid development and integration of generative AI into our daily lives, such as the use of ChatGPT [9] and Gemini [10]. This thesis explores and analyzes several data privacy inference infrastructures against cloud adversaries in NLP tasks.

State-of-the-art (SOTA) software data privacy protection schemes often preserve privacy by performing data anonymization [6, 11], either through the use of encryption [12] or differential privacy (DP) guarantees [13, 14]. However, both of these methods have shortcomings.

DP-inspired methods often trade off privacy protection with data utility [15]. On the other hand, encryption schemes such as homomorphic encryption suffer expensive computational overheads as well as data utility degradation to guarantee privacy [16, 17]. This thesis aims to study a privacy preserving inference infrastructure that has a strong privacy guarantee while preserving data and model utility in the NLP domain. Two NLP tasks, binary spam detection and multi-class sentiment analysis, are used to assess the performance of the inference infrastructure and compare it against existing techniques. The infrastructure itself takes inspiration from model extraction attack techniques [18], including knowledge distillation and active learning [19].

Different from traditional teacher-student knowledge distillation [20, 21], where information is optimized to transfer from a complex teacher model to a smaller and simpler student model, the proposed infrastructure focuses on privacy preservation. Therefore, in the experiments, the teacher and student architectures are the same for simplicity. The student will have the same capacity but pre-trained on a small amount of data, thus having inferior abilities compared to the teacher and may only be interested in learning a partial functionality of the teacher's abilities. In this case, a potentially malicious but accurate cloud model distills knowledge and is used to fine-tune a trustworthy but low-accuracy local model to learn the teacher's sub-tasks.

Several methods can be used to protect privacy in the teacher student knowledge distillation process. One way to preserve privacy is by applying DP noise on the sensitive dataset before querying the teacher. Another way is by generating synthetic data or finding similar data from an insensitive public dataset to replace the original sensitive text messages. Both methods will be explored in this thesis. In this thesis, we use text sanitation [22] to add DP-noise for the prior method. For the latter method, we explore using public datasets and similar functions that measure the distance on the vector representation of the text, common functions include cosine similarity [23] and Euclidean distance [24].

Finally, to preserve accuracy and maximize data utility, active learning is used. Through active learning, the student model will iteratively train on data that it struggles with or is most

uncertain about [19], hence actively learning. This technique is commonly used to implement powerful model extraction attacks on MLaaS [25, 26]. In this thesis, active learning is repurposed into a defense mechanism against adversaries instead of an attack, allowing the student model to learn from a small number of training samples while achieving good accuracy.

With the rapid emergence of MLaaS in our daily lives, the use of data privacy preservation techniques has become essential to protect sensitive information from potential adversaries. The development and study of privacy-preserving inference infrastructures in this thesis will provide insight into techniques that can allow more efficient data utilization and less performance degradation in addition to privacy preservation compared to SOTA methods.

Chapter 2

Background

This chapter builds on the introduction and will first discuss the adversaries this thesis tackles to give a clear picture of the threat model. Then, the next section describes existing software approaches to preserve privacy during inference, with an emphasis on encryption and DP schemes. Next, we will discuss model extraction techniques, notably knowledge distillation and active learning, and how they can be utilized in the proposed privacy preserving inference infrastructure. Finally, we will look into popular architectures used for NLP tasks and focus on LSTMs and Transformers, which we use in this thesis to test our framework.

2.1 MLaaS Adversaries

MLaaS was first proposed as an architecture to create flexible and scalable ML services [1]. Nowadays, this term refers to the expansive ML tools offered by cloud computing providers. The widespread use of MLaaS is accompanied by growing concerns about their safe usage, coming from both end users and service providers. In the literature, ML attacks commonly refer to attacks against model owners, broadly categorized into attacks that degrade the model utility and attacks that aim to gain knowledge of the system [27, 28]. The prior can happen in both training and inference phases, where poisoning attacks modify training data to disrupt

model training and evasion attacks perturb testing data during inference to degrade model performance [27, 29]. The latter are exploratory attacks [27] implemented during inference to gain knowledge about the system, whether on the model architecture, data, or trained parameters. Model extracting attacks is one such exploratory attack, more details about it will be discussed in Section. 2.3. In contrast, the adversary this thesis focuses on attacks individual end users instead of model owners. The proposed privacy preserving inference infrastructure aims to protect the sensitive data of the end users from adversaries that are malicious third-party cloud providers with high-performing ML models.

ML spam detection services are typically conducted in the cloud instead of locally due to storage and computational limitations [30, 31]. As data-hungry applications become more integrated into our lives, the risks of data abuse and exploitation are exemplified. For instance, while email spam services aim to detect and block harmful emails, they can potentially eavesdrop on message content for surveillance. In 2016, Yahoo was reported to help the US government track foreign terrorist activities by adapting a spam filter to scan private user emails [32, 33]. In the past decade, there have been numerous privacy lawsuits involving large tech companies for violating data protection laws [34].

The increased usage of MLaaS, especially black-box models, where there is little transparency to the internal workings of the ML model and the users can only get hard output predictions, makes it harder to know whether the user input data is well-protected or not. More recently, the hype and use of generative AI, such as ChatGPT, brought out new privacy concerns with the exploitation of public and private user input data for training [35]. With developing data privacy policies trying to catch up to the swift advancement in ML technology, it is ever more important to develop infrastructure that prevents the abuse and exploitation of user data. In the spam detection case, the proposed privacy-preserving framework can protect the original text messages during inference by producing a local model specialized in spam detection by querying the cloud model using insensitive data.

2.2 Existing Privacy Preserving Approaches

The objective of the privacy preserving inference infrastructure is to safeguard the privacy of sensitive user data against adversarial third-party cloud providers. Naively, the most straightforward approach to preserve privacy is to run all computations on local devices. However, MLaaS often only provides black-box models to protect trade secrets. Furthermore, resource constraints such as storage and hardware compute capabilities on personal devices inhibit the deployment of ML models on them [36], sparking research to optimize training in the cloud [37] along with the emergence of products like Google Cloud TPU with high computational capabilities [38]. This trend calls for methods to protect the confidentiality of the data sent to cloud providers. Two categories of methods that are closely related to the proposed privacy preserving infrastructure will be discussed, encryption and DP.

2.2.1 Encryption

Encryption schemes can be used to encrypt data as well as the ML model architecture [39]. This thesis focuses on preserving data privacy and thus will only discuss cryptography-based methods that encrypt data for MLaaS. The most popular technique is homomorphic encryption (HE). It is a method that allows direct computation on encrypted data without decryption to protect data in non-trustworthy environments. In ML, HE was initially only applied to simple classification tasks that use models such as Naive Bayes and Decision Trees due to the intensive overhead added by the encryption process [17, 40]. Later on, the use of HE has been extended to neural networks (NNs) by transforming NNs into CryptoNets that use encrypted data as input and output during classification [12]. However, although allowing slightly faster inference compared to previous cryptography-based methods, CryptoNets still suffer high latency and limitations on model width and depth, motivating the development of low-latency CryptoNets combined with techniques from transfer learning to lower latency [41]. CryptoDL also offers a means to reduce run time and latency through by introducing new techniques to approximate activation functions [42].

More recent research continues to apply HE on more complex scenarios, including on ResNet for classification, achieving similar accuracy at the cost of non-practical run times [16]; or combining HE with federated learning [43] to slightly speed up training with HE. There is also research investigating the practical use of HE in ML for specific fields, such as medicine and bio-informatics [44] or finance [45]. As the proposed privacy preserving infrastructure using knowledge distillation and active learning is inherently different from cryptography-based methods, it does not suffer from the main challenge in HE methods: latency due to encryption and decryption calculations. Furthermore, the proposed method distinguishes itself from encryption schemes by using a replacement of instead modification of the sensitive data as input.

2.2.2 Differential Privacy

Differential privacy is widely accepted as a rigorous definition of privacy in traditional private data analysis and its use has been extended to ML [15, 46, 47]. DP-inspired methods often work by adding perturbation or noise to the model or data [39, 46] to conduct private data analysis using ML. One of the earlier methods incorporating DP into Deep Learning (DL) uses a distributed framework and selectively adds noise to the gradient within a privacy budget during stochastic gradient descent (SGD) [47]. Following this work, a simpler DPSGD scheme was developed by applying the L2 norm on gradients and then adding DP noise before gradient descent [48]. However, these initial DP-inspired frameworks have been shown to be susceptible to information leakage in certain cases. This drove research to investigate the use of DP in more complex settings such as GANs [49] and LSTMs [50]. More recent papers study how large language models can be differentially private learners [51, 52]. However, most of the above work mentioned using DP considers a different threat model, where the goal is to prevent the ML model from information leakage instead of protecting user data from abuse and exploitation.

In the context of our threat model, there has also been work done on using DP to protect data in an untrusted cloud environment [53, 54]. Wang et al. [53] used a mechanism that

consists of arbitrary data nullification and random noise addition on the sensitive input data. Furthermore, they utilized a public dataset to get the cloud model accustomed to noisy training to reduce performance degradation. Xu et al. [54] proposed a differential privacy obfuscation framework that first uses a feature distillation model to minimize the amount of data shared to the server and then modify the learned features. However, DP-inspired methods often sacrifice data utility in exchange for privacy guarantees. Both works [53, 54] saw a degradation in model accuracy with the addition of differential privacy guarantees. Specific to text, Yue et al. proposed an algorithm to incorporate DP guarantees into text sanitization [22], a type of data anonymization.

Similar to DP-inspired methods, the proposed infrastructure also has a trade off between model performance and privacy. Although the use of generated or similar data instead of the actual sensitive data may decrease the accuracy, it provides a stronger privacy guarantee compared to DP-inspired methods as the third-party adversaries have no access to the original sensitive data.

2.3 Model Extraction Techniques

As mentioned in Section 2.1, model extraction belongs to exploratory adversarial attacks that try to extract information from ML systems [27]. For MLaaS, techniques in model extraction can be used to "steal" the cloud model, where the attacker tries to replicate model characteristics through a predictive user interface with a minimal amount of queries [18, 55]. In recent years, as model owners try to further safeguard their systems through model extraction defenses [56, 57, 58], model extraction attacks have become more well developed [26, 25, 59, 60]. Depending on the exact scenario, model extraction attacks can have two goals: getting exact properties and parameters or getting approximate model behavior [61]. Two popular techniques commonly used to extract approximate model behavior are knowledge distillation and active learning. For instance, Knockoff Nets [62] utilize modifications of both techniques to extract information from black-box models. Although the proposed infrastructure takes inspiration

from model extraction due to its query-based nature, it has a drastically different goal: data privacy preservation instead of model stealing.

2.3.1 Knowledge Distillation

Knowledge distillation (KD) is a type of model compression technique. It aims to distill knowledge from a larger and more complex teacher model into a smaller and more compact student model, often to save computational resources [20, 21]. Depending on the knowledge form, different KD techniques require varying amounts of access to the teacher model architecture, weights, and parameters [63, 64]. In particular, response-based KD aims to train a student model that mimics the teacher's prediction by only using the teacher's last layer output [20] and is most feasible for our scenario with a black-box cloud model. Building on traditional KD, the student model in Knockoff Nets learns from data with a distribution different from the teacher training data [62]. Similarly, Black-Box Ripper also uses proxy data to train the student model [65]. Both of these works assume no access to training data and thus use what is called "data-free" KD [66]. The proposed privacy preserving infrastructure also tries to train a local student model in a "data-free" environment. However, the student model does not necessarily want to gain all the teacher functionality, differentiating our method from other uses of data-free KD where the goal is to extract a fully functioning teacher model. This would help minimize our query budget and increase data utility.

2.3.2 Active Learning

Oftentimes in ML applications, while there is an abundance of unlabelled data, the labeling task can be tedious and time-consuming. Active learning (AL) [19, 67] is a technique to help achieve high model accuracy with less labelled training data by carefully choosing what data to train with. The data is typically chosen based on uncertainty or information-density based measurements such as entropy [68]. To get labels, an active learner would query an oracle with unlabelled data. With the advancement of MLaaS, the oracle is no longer limited to human

annotators, but can also be high-performing ML models, thus making AL desirable for model extraction attacks [69]. In Knockoff Nets [62], a student trained using active learning sampling outperforms random sampling. Furthermore, ActiveThief shows the viability of using active learning with uncertainty sampling and public datasets to extract deep classifiers on image and text [26]. The success of AL in iterative query-based model extraction attacks makes it an ideal technique to use for the proposed privacy preserving infrastructure with limited data and a small query budget.

2.4 Architectures for Sequential Data

As this thesis focuses on evaluating the privacy-preserving framework in the NLP domain, we need to use architectures that can process long sequential text messages. The development of sequence models started with Recurrent Neural Networks (RNNs), which use the output of the previous step as the input to the current step, capturing order and historical dependencies [70]. However, RNNs suffer from vanishing gradients and are poor at retaining long-range information, giving rise to the development of Long Short-Term Memory (LSTM) models. An LSTM can better sustain long-term dependencies as it solves the vanishing gradient problem by using gates to regulate and direct the flow of information [70]. More recently, the development of attention and transformer models [71] provide a powerful architecture to solve sequential problems without recurrance. Many SOTA language models, such as BERT and GPT, use transformers. In this section, we will further describe and discuss LSTMs and transformers as they are used in our evaluations for different NLP tasks.

2.4.1 LSTM

LSTMs are a type of RNNs, they solve the vanishing gradient problem by incorporating nonlinear, data-dependent controls into RNNs to make sure that the gradient does not vanish [70]. These controls are called gates and serve several different purposes. Figure. 2.1 depicts

the internals of an RNN unit as well as an LSTM unit. The RNN feeds the hidden state of the previous time step and the input of the current time step through sigmoid functions to get the next hidden state and output. This simple structure is prone to gradients vanishing or exploding when the weights get continuously multiplied in each step. In contrast, the LSTM unit is more complex, consisting of different gates: the input gate, the output gate, and the forget gate. The combination of gates prevents the weights from getting too small or large and solves the vanishing gradient problem. In this thesis, we use bi-directional LSTMs, which learns information flow from both forward and backward directions. This allows the model to better recognize phrases and patterns in a sequence.

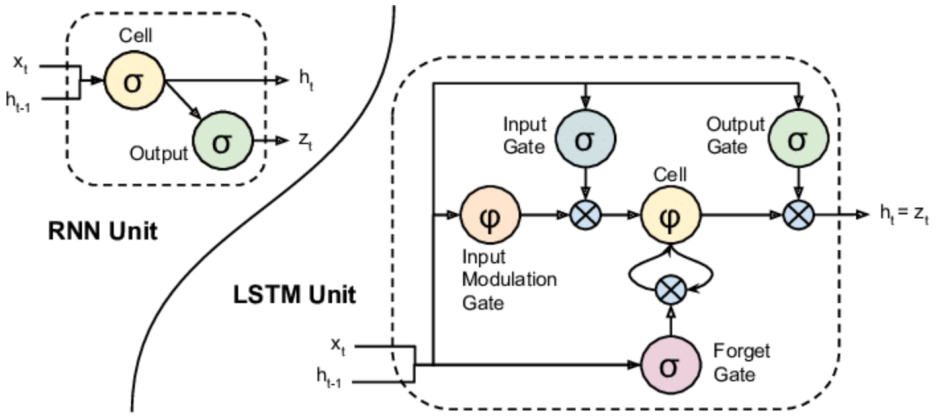

Figure 2.1: RNN vs LSTM Architecture

2.4.2 Transformer

Nowadays, transformers have become the standard for NLP applications, both in research and in the industry. Many commercially used NLP applications, such ChatGPT or BERT, use the transformer architecture. Furthermore, platforms such as HuggingFace [72] provide developers and researchers with easy access to transformer models, aiding to their widespread popularity. A transformer consists of two main parts, the encoder and the decoder. The original transformer architecture was presented in the paper 'Attention is all you need' [71], as shown in Figure. 2.2.

The encoder takes input embedding and outputs representations. The decoder then uses

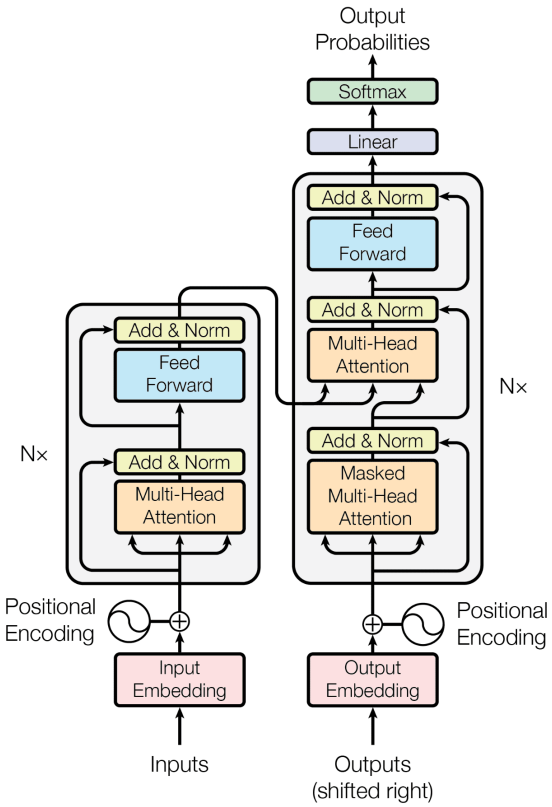

Figure 2.2: Original Transformer Architecture

the output embeddings, which are the input embeddings shifted right so that the current time step token is lined up with the next time step token, and encoder representations to output probabilities for the desired task. Depending on the task the language models are trained for, they can be encoder or decoder based models. Encoder-based models are more commonly used to learn embeddings for tasks such as predictions and classification. A popular encoder-based model is BERT [73]. In contrast, decoder-based models are typically designed to generate new sequences and used for tasks such as question answering and text summarizing. The GPT series models [74] are decoder-based. The combination of the encoder and decoder can be used for tasks such as machine translation. In this thesis, as the task we want to perform is classification, we use an encoder-based transformers model.

Chapter 3

Methods

This chapter will discuss the experimental setup, methodology, and results. It will first go over the specific dataset and tasks as well as the models used in the experiments. Then it will go through the details of the privacy preserving inference infrastructure. Finally, it will describe the different experiments performed to test the framework and provide the relevant results. Note that in this section, 'sensitive' and 'private' will be used interchangeably to describe the data we want to protect.

3.1 Experimental Setup

The privacy preserving framework can be used on different domains of data. In this thesis, we explore it's effectiveness on NLP MLaaS systems. We assume that there is a high-performing NLP classification model in the cloud that only outputs hard labels. Furthermore, we do not have any knowledge on the architecture of the cloud teacher model, hence working with a 'black-box'. We then assume that the user has the resources and compute to train a local student model that mimics a partial functionality of the teacher model.

3.1.1 Dataset and Tasks

In particular, we test our framework on two tasks: a simpler binary spam classification task and a harder multi-class emotions classification task. Specifically, the UCI SMS Spam Dataset [75] is used for binary spam classification and contains around 5 k text messages. For the emotions classification, the Contextualized Affect Representations for Emotion Recognition (CARER) dataset [76] is used and has around 417 k text messages, categorized into 6 classes of emotions. Both datasets are publicly available on the internet. Most of the experiments will only use the emotions dataset as the spam dataset is a relatively much easier task and the results may not be representative.

In our experiments, we make the assumption that there exist an insensitive dataset that overlaps with the teacher model's input space. This assumption is made as high-performing language models, such as GPT-4, use "both publicly available data (such as internet data) and data licensed from third-party providers" [9] for training. Therefore, it is likely that the cloud model has trained with the insensitive data and can make accurate predictions with it. Furthermore, we assume that the sensitive dataset has an overlapping distribution with the teacher's input state. We can make this assumption as due to the vast amount of data cloud models have been trained on, it is likely that they have been trained on data with a similar distribution to our sensitive set. Further experiments exploring the misalignment between the teacher input space and the student input space will be explored in Section. 3.3.

3.1.2 Models

Different models are used for each task. An LSTM-based model is used for binary spam detection while a transformer-based model is used for emotions classification. The LSTM model consists of 5 bidirectional LSTM layers with residual connections and a linear layer for classification. The transformer model consists of encoder blocks and a classification block. Both architectures are shown in Figure. 3.1. These models are chosen as they are commonly used for NLP tasks as described in Section 2.4. We use the LSTM architecture for the binary

spam detection problem as it is a simpler task and we have a small amount of data. Using a complex model on limited data makes it more likely to overfit, degrading its ability to generalize on test data.

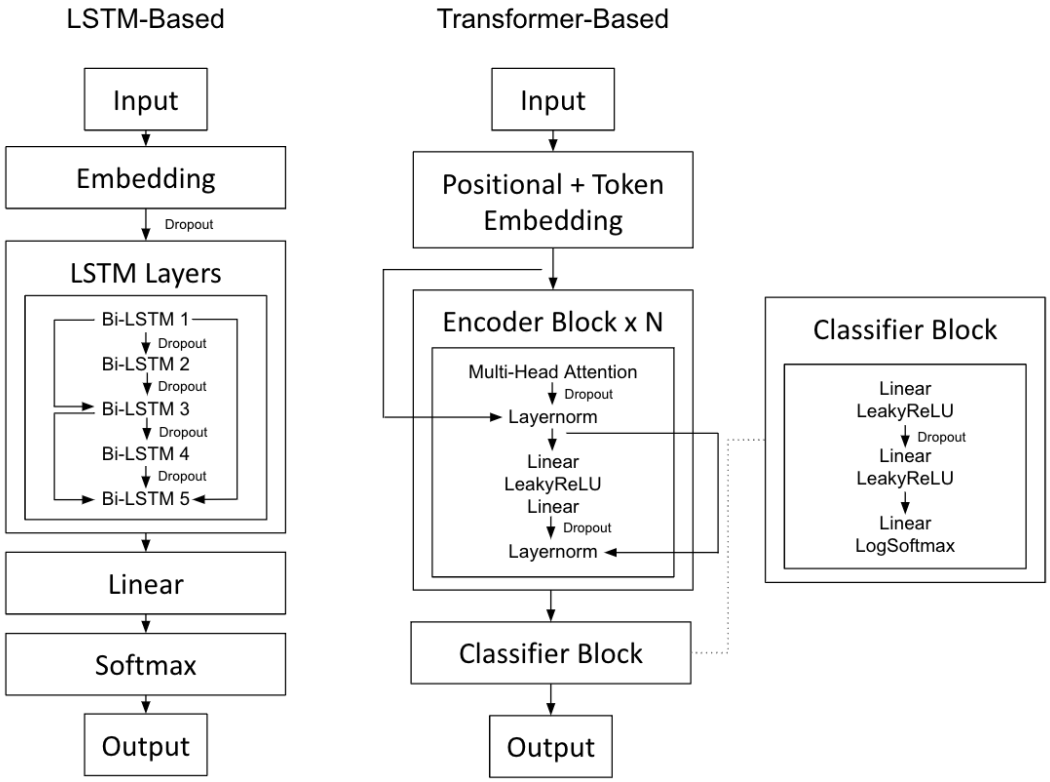

Figure 3.1: Model Architectures

Our framework does not make any assumptions on the architecture of the teacher model. To start off, for simplicity, we use similar student and teacher architecture. This means that for spam detection, the student and teacher both have the LSTM architecture. For the emotions classification, the student has 1 encoder block while the teacher has 2 encoder blocks within the transformer architecture.

3.2 Framework Implementation

Overall, the framework consists of three parts:

1. Knowledge distillation from the teacher to the student.

2. Privacy preserving techniques to protect the sensitive private data.

3. Querying strategies to maintain model performance utility against privacy costs.

The overall infrastructure is shown in Diagram. 3.2. The student is first pre-trained using a small labelled dataset. Then, through an iterative querying process, the student learns the functionality of the teacher. Starting off, the student is bootstrapped with the pre-training data.

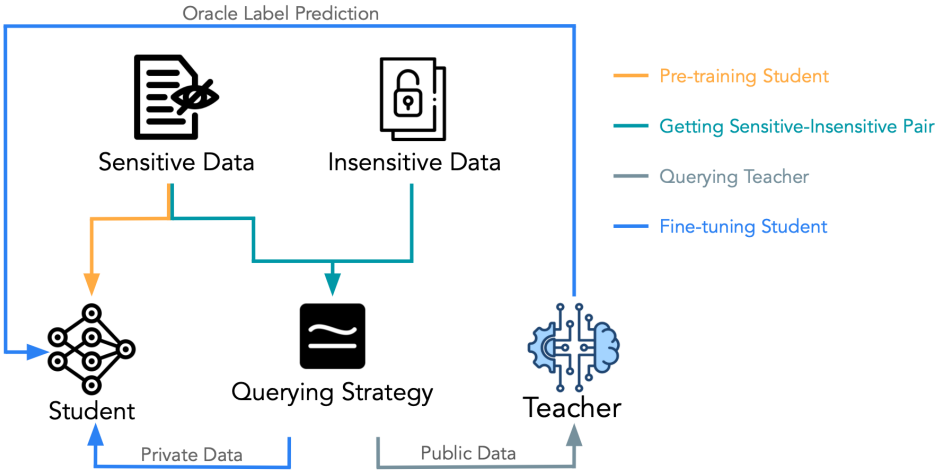

Figure 3.2: Overview of Privacy Preserving Inference infrastructure

In each iteration of training, we add new queries to the teacher by a fixed amount and perform mini-batch training with knowledge distillation. During each epoch of the mini-batch training, we have a batch of private data and a batch of public data. Using a specific querying strategy, the public data is fed into the teacher and the private data is fed into the student model.

The insensitive dataset can be constructed using various methods. In this thesis we investigate 3 ways to do so: using cosine similarity to find sensitive-insensitive pairs, differential privacy to noise sensitive data, and using a public dataset that has a similar distribution to the private dataset to preserve privacy. Other methods such as differentially private data generation [77, 78] can also be applied. However, those are not within the scope of this thesis as they require more computational resources and data, adding an extra layer of complexity.

Different querying strategies can also be used during the iterative training process. The most naive strategy is random sampling. To enhance the fine-tuning process, we can use active learning with entropy sampling to select which data points the student is most uncertain about and fine-tune with these points. A list of specific hyperparameters used for the experiments can be found in Appendix A.

3.3 Experiments and Results

3.3.1 Pre-training

We train the teacher models fully until they reach a high accuracy with insensitive data. On the other hand, the student is pre-trained with a small set of labelled private data, which we assume that the teacher has never seen but has a similar distribution to the teacher's inputs. Specifically, the student is only trained for 3 epochs and uses around 800 points for spam detection and around 70k points for emotions classification. The starting accuracies after pre-training are shown in Table. 3.1. We train the LSTM and transformer model on both the spam detection and the emotions classification task.

Model	Task	Teacher Acc, time (minutes)	Student Acc
LSTM	Spam Detection	0.96, 1.3	0.46
LSTM	Emotions Classification	0.84, 65.7	-
Transformer	Spam Detection	0.92, 0.8	-
Transformer	Emotions Classification	0.88, 43.5	0.43

Table 3.1: Starting Accuracy of Teacher and Student

The teachers are all trained with 20 epochs. Due to its recurrent nature, the LSTM model taking $\sim$ 50% more time than the transformer model. We find that the LSTM has a higher teacher accuracy for the spam detection task compared to the transformer. Therefore, although it takes longer for the LSTM to train, we chose to use this model for the spam task as the absolute training time is short due to the small dataset size. The lower accuracy achieved by

the transformer for the spam detection task is likely due to the transformer having too many parameters and overfitting on a small training set.

For emotions classification, we see that the transformer model achieves a higher accuracy. Furthermore, it is also more time efficient. Therefore, we use the transformer architecture for the emotions task. The starting student accuracies for each task are also shown in Table. 3.1.

3.3.2 Using Cosine Similarity for Sensitive-Insensitive Pairs

Cosine similarity is a metric commonly used to measure the distance between two vector representations [23], as shown in Eq. 3.1. With each sensitive data point we feed into the student, we can use cosine similarity to find a similar insensitive data point to feed into the teacher. The larger $cos(\theta)$ is, the more similar the two vectors are.

$$cos(\theta) = \frac{x \cdot y}{|x||y|} \tag{3.1}$$

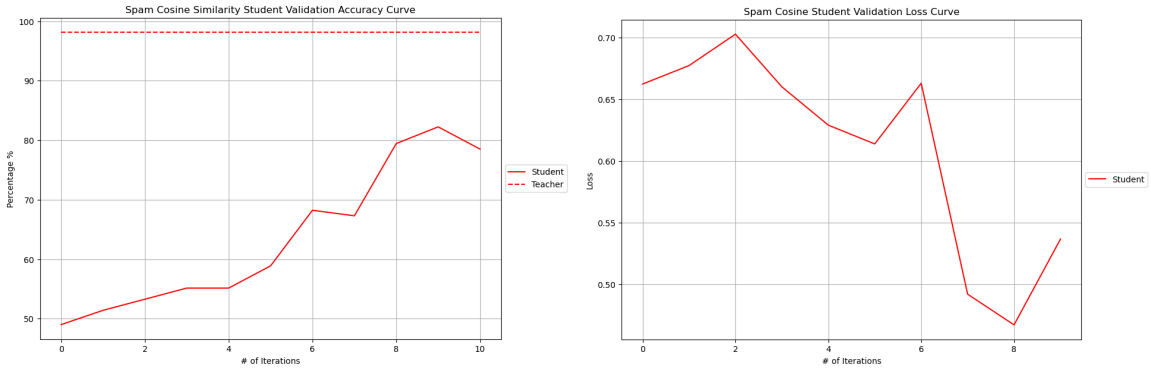

Figure 3.3: Spam Cosine Similarity Student Validation Accuracy and Loss

As shown in Figure. 3.3, we fine-tune the student with the teacher for 10 iterations. In each iteration, we do mini-batch fine-tuning for 8 epochs with 200 additional data points. After 10 iterations, the student is fine-tuned with 2000 sensitive data points. From the increase in accuracy, we can say that the model is able to gradually generate better softmax representations such that we can find more accurate sensitive-insensitive pairs with cosine similarity. The

decreasing loss curve also shows that the model is learning. However, we do note that there is still a gap between student and teacher performance, with the student being 20% lower in accuracy.

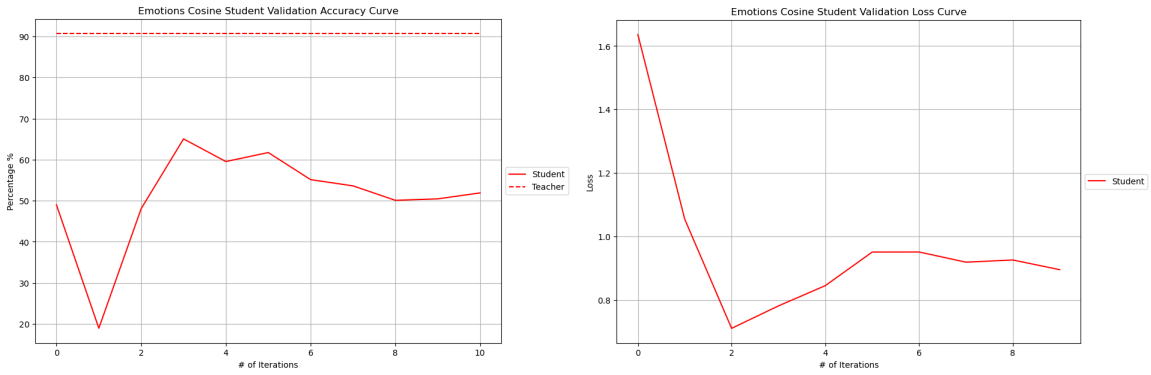

Figure 3.4: Emotions Cosine Similarity Student Validation Accuracy and Loss

We also tested cosine similarity on the larger emotions dataset, which contains 417k data points. As shown in Figure. 3.4, we fine-tune the student with the teacher for 10 iterations. In each iteration, we fine-tune the student with 2000 more points for 8 epochs, seeing 20k sensitive data points in total. We observe that the student is not able to learn well for this task. The loss drops initially but then remains at similar levels. This may be because the multi-class task is harder and our model is not able to produce good enough softmax representations for the cosine similarity to work well. As shown in Figure. 3.4, the final accuracy for the student is similar to its starting accuracy. Since we see cosine similarity failing, we experiment with different ways of constructing our insensitive dataset to preserve privacy and focus on the emotions task as it is harder.

3.3.3 DP-Noise vs. Public Dataset for Privacy Preservation

Instead of finding sensitive-insensitive pairs, we could fine-tune the student with only the sensitive data or only the insensitive data. For the former case, we construct our insensitive dataset by adding DP-noise to the sensitive dataset to query the teacher, getting DP-noised hard label predictions. Then, we can use these predictions and the non-noised private data to

fine-tune the student. For the latter case, we can construct our insensitive dataset with public data and also fine-tune the student with public data. In both cases, to make training more efficient, we apply entropy sampling during active learning to choose which points the student is most uncertain about in each iteration.

As mentioned in Section 2.2.2, while DP is typically applied in the gradient descent process, there are also methods that focus on adding DP-noise on inputs. Since we assume no access to the architecture of the teacher model, we cannot use gradient-based DP methods. Instead, we take inspiration from text sanitization [22] proposed by Yue et al., a method that uses the concept of differential privacy during data anonymization.

A defined in [15], we can say that an algorithm or model, M, with domain D is ϵ-differentially private if $\forall S \subseteq Range(M)$ and $\forall x, y \in D$ s.t $||x - y||_1 < 1$:

$$Pr[M(x) \in S] \leq exp(\epsilon) Pr[M(y) \in S] \tag{3.2}$$

where $||x||_1 = \sum_{i=1}^{|X|} |x_i|$ is defined as the L1 norm of the x dataset and ϵ is a positive number. S represents a set of possible outputs. Intuitively, the smaller ϵ is, the closer $exp(\epsilon)$ is to 1, and the more similar the probabilities of $Pr[M(x) \in S]$ and $Pr[M(y) \in S]$ are. Therefore, smaller values of epsilon provide a higher differential privacy guarantee as the model is less likely to distinguish between two input data points. The smaller ϵ is, the tighter privacy budget we have, and the more noise we add to the data to make different points indistinguishable to each other. In [22], Yue et al. provided a variant of differential privacy, the Utility-Optimized Metric Local Differential Privacy (UMLDP). Using their definition, they performed text sanitization on a vocabulary V split into a sensitive vocabulary set V_S and an insensitive vocabulary set V_I. In English, insensitive words, such as 'an/a/the/that/...' are common across all text and frequently used. In contrast, rare words, which may include sensitive information such as locations, names, dates, are less often used. Therefore, Yue et al. used the most infrequent words as the sensitive set V_S and the remaining as V_I.

Taking inspiration from their method, we split our dataset vocabulary similarly. Then, we

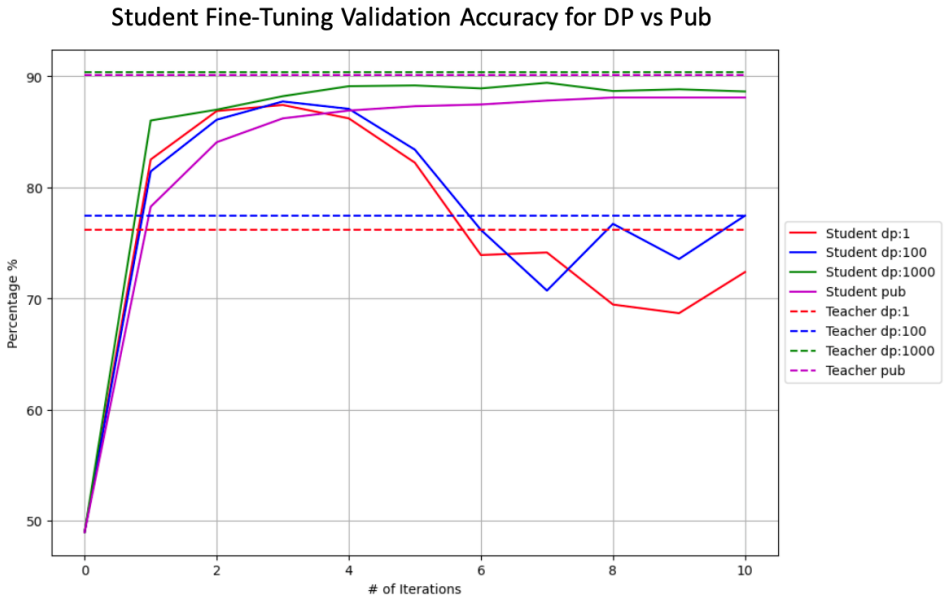

Figure 3.5: DP-noised Private vs Public Student Validation Curves

find the word embeddings $\phi(x)$ using a pre-trained BERT model for all tokens, x, in V. Finally, we iterate through each word in each sentence. If word $x \in V_S$, we sample $y \in V_I$ and replace x by y with the following probability

$$Pr[M(x) = y] = Const \cdot \frac{exp(-\epsilon \cdot d(\phi(x), \phi(y))/10)}{\sum_{y' \in V_N} exp(-\epsilon \cdot d(\phi(x), \phi(y'))/10)} \tag{3.3}$$

where $Const$ is a positive multiplying factor and $d(\phi(x), \phi(y))$ represents a distance measurement between x and y embeddings. It is calculated with the following formula:

$$d(\phi(x), \phi(y)) = |\phi(x), \phi(y)| - 1 \tag{3.4}$$

Using the above formulation, we run tests with the emotions dataset with different levels of differential privacy with ϵ = 1, 100, and 1000. Two examples of applying text sanitization with different ϵ values are shown in Appendix B. By setting the bottom 85% least frequent words as the sensitive vocabulary set, we compare the performance of the student fine-tuning using text sanitization against using the public dataset as shown in Figure. 3.5. In both cases, the student learns to predict the partial functionality of the teacher, learning to predict 3 classes

out of 6 classes.

3.3.4 Domain Shift with Public Dataset

In our previous experiment, we assume that when we construct the insensitive dataset with public data, the public dataset overlaps with the teacher input space such that during fine-tuning, the teacher is able to generate accurate labels for the data points. However, this may not always be the case. It is possible for the public dataset to have a domain shift, containing data that the teacher has not seen before and cannot accurately predict. Therefore, using the emotions classification task, we experiment what would happen when 25%, 50%, and 75% of the fine-tuning data is from the spam dataset which the teacher has not seen before, and test our fine-tuned student on the emotions task.

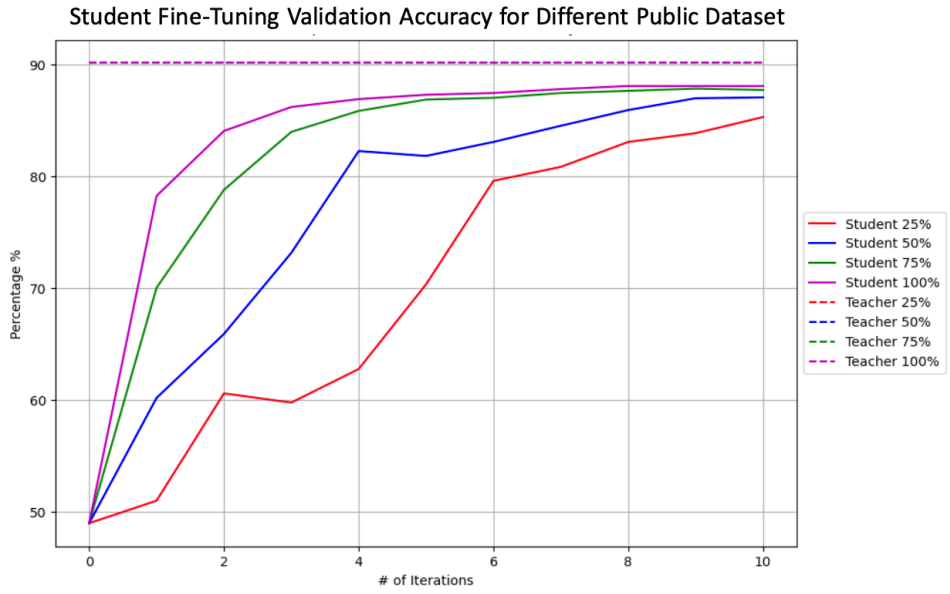

Figure 3.6: Public Domain Shift Student Validation Curves

As seen in Figure. 3.6, we can observe that when there is more domain shift, the student accuracy increases more slowly. After two iterations, the 25% dataset has an accuracy of $\sim$ 60% while the dataset with no domain shift has an accuracy of $\sim$ 85%. As the public dataset consists of more emotions data, the student performance increases more quickly and

then plateaus. With more fine-tuning, the gap between the student performance using different datasets shrinks. At the end of the last iteration, the 25% dataset has an accuracy of ~ 85% while the 100% dataset has an accuracy of ~ 88%.

3.3.5 Noisy Student Fine-tuning

In our privacy preserving framework using DP-noise, we use noisy predictions from the teacher and the raw private data points to fine-tune the student since we wanted the student to predict on raw private data points in the future. However, we can also feed noisy private inputs into the student. This could provide more consistency in the framework and make the distillation process easier since we are fine-tuning the student with noisy inputs and the corresponding noisy labels.

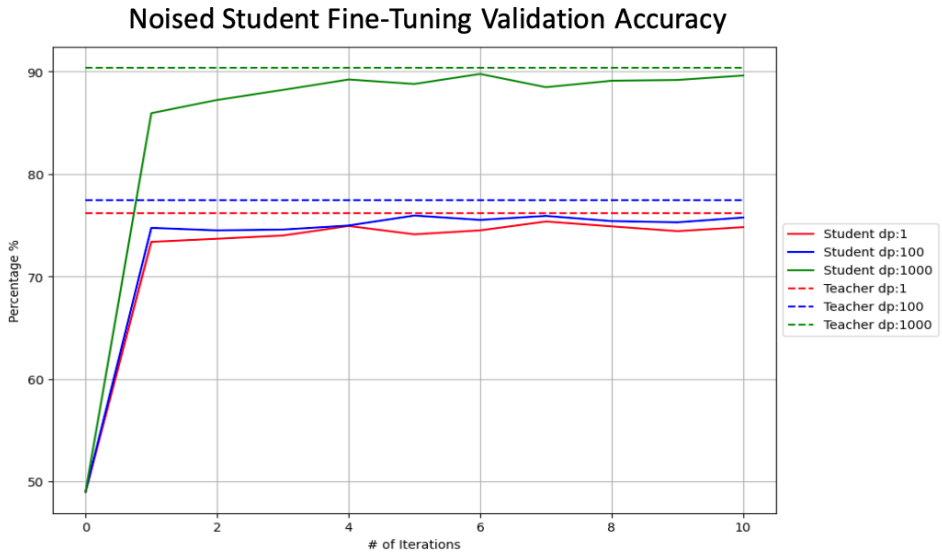

Figure 3.7: Noisy DP Private Student Validation Curves

Using noisy inputs for the student, we observe a more stable validation curve as observed in Figure. 3.7. Instead of having a high accuracy at the start like in Figure. 3.5, the student validation accuracy in Figure. 3.7 is always bounded by the teacher's performance on the noisy validation data. At the end of fine-tuning, the final accuracies of the student with noisy training and non-noisy training are similar.

3.3.6 Combining Private and Public Data

With the framework using public data outperforming the one using DP-noised private data, we investigate the use of a combination of public and private data to query the teacher. In this case, we go back to fine-tuning the student with raw data points while feeding the teacher noisy private data or public data. We construct our insensitive dataset with different ratios of private and public data. Specifically, we use datasets that are composed of 25%, 50%, and 75% private data, with the remaining being public data.

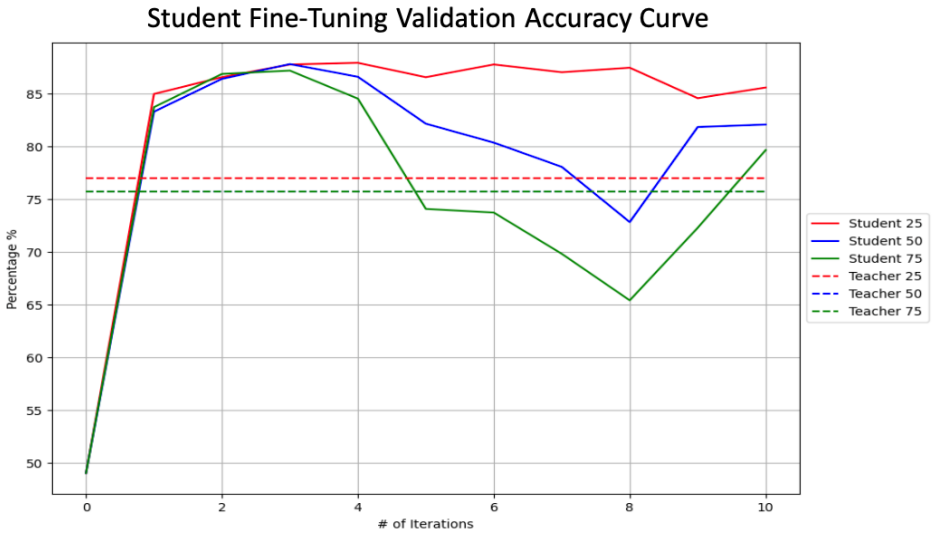

Figure 3.8: Student Validation Accuracy Using both Public and Private Data

From the results as shown in Figure. 3.8, we can observe that the more public data we have in our insensitive dataset, the less the accuracy drops after $\sim$ 4 iterations. The teacher accuracy plotted is its performance on the newly constructed dataset. In the cases where 50% and 75% of the dataset is composed of private data, the teacher has very a similar performance of $\sim$ 75.5%. When there is less private data, we see that the teacher performance is also better. Furthermore, it is worth noting that the addition of public data stops the student accuracy being bounded by the teacher performance, becoming more similar to the teacher performance on public data as seen in Figure. 3.5.

3.3.7 Changing the Student Pre-training Bootstrap Size

In the experiments for emotions, we used a bootstrap training dataset of ~ 70k data points. The original idea behind using a bootstrap is to make sure that the student can generate good enough representations to work with cosine similarity. However, we see in our cosine similarity experiments that finding sensitive-insensitive pairs fails for the emotions tasks as the student is unable to outputs good enough output representations after pre-training. Therefore, we have shifted to using DP-noised and public datasets to preserve privacy. In the later approaches, we don't necessarily need to have a bootstrap set to allow the student to have a starting vector representation. Therefore, we assess the effect of reducing the bootstrap dataset on the student fine-tuning efficiency. We run experiments using public data when the student is bootstrapped with 18k and 35k datasets and compare them with the 70k case.

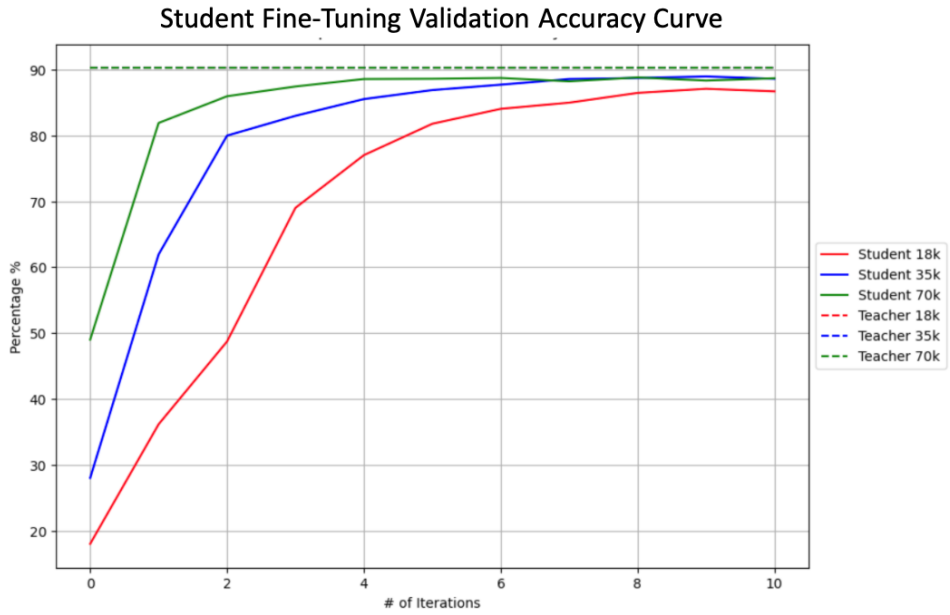

Figure 3.9: Student Validation Accuracy with Different Bootstrapping Dataset Size

A shown in Figure. 3.9, we see that the starting accuracy drastically decreases when the bootstrap set decreases. Furthermore, the student fine-tuning accuracy increases more slowly with less bootstrapping data. However, with more iterations, the final accuracies with different bootstrapping amounts are similar, with students with more bootstrapping data having slightly

higher accuracy.

3.3.8 Changing the Student Pre-training Epochs

Other than changing the boostrapping dataset size, we can also investigate how the number of epochs the student pre-trains affects the fine-tuning process. Previous experiments have all used a student that has been pre-trained with 3 epochs. We compare what happens when the student is pre-trained with 5 epochs or 8 epochs.

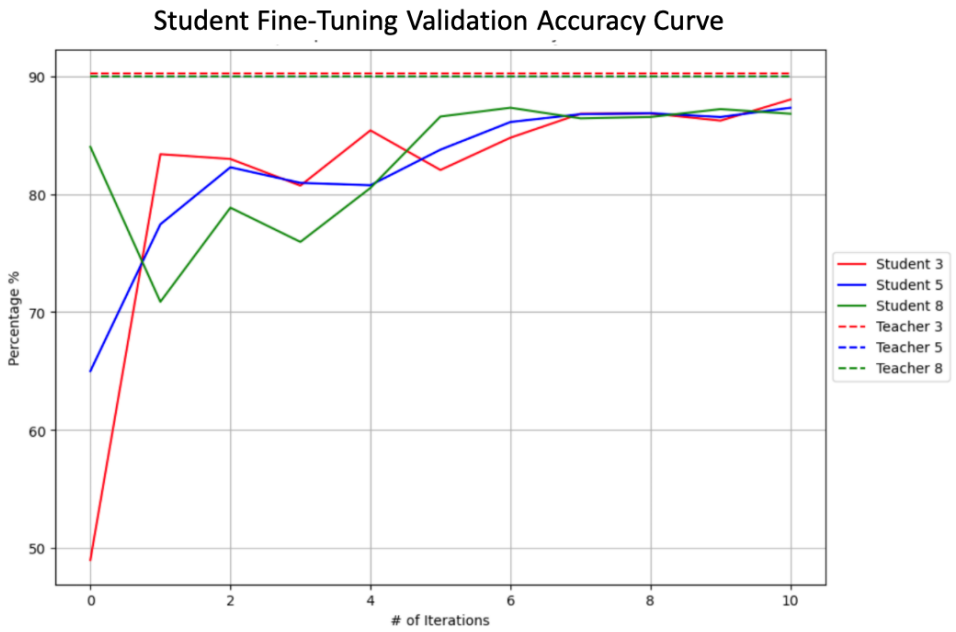

Figure 3.10: Student Validation Accuracy with Different Pre-training Epochs

As shown in Figure. 3.10, we tested using the insensitive set to observe how the trends differ. Although each student's starting accuracy is different due to having different amounts of pre-training epochs, they quickly converge to the same behavior in the fine-tuning process and have similar end accuracies.

Chapter 4

Findings

In this chapter, we will go in depth to analyze the results and provide a detailed discussion on the findings from our experiments. Then, as our framework takes inspiration from model extraction, we will also discuss the ethical and legal implications surrounding the use of the framework.

4.1 Discussion

4.1.1 Challenges with Cosine Similarity

Our experiments compared different methods to preserve privacy by various means of constructing the insensitive dataset such that it doesn't include sensitive information and is representative of the private data. The first method we explored was cosine similarity. The increase in accuracy for the student during fine-tuning shows that for the easier spam detection task, cosine similarity is able to find similar pairs using the output softmax embeddings from the LSTM-based model. However, we do observe that it is unable to reach the teacher accuracy with our fine-tuning parameters, achieving a performance that is 20% lower than the teacher. Furthermore, through the experiment with the emotions dataset, we see that the

softmax embeddings are not always good representations to use cosine similarity with.

Although the starting accuracy of the transformer-based student is similar to the LSTM-based student, the transformer-based student cannot be fine-tuned with cosine similarity. For both cases, to improve the performance of the student, we need to have better starting softmax-embeddings. This can be achieved by either pre-training the student more or fine-tuning with more data. However, as the proposed framework should work in scenarios with limited data, we are not satisfied with using either method, motivating other means of constructing the public dataset. Since we see the cosine similarity methodology fail in the emotions classification case as it is a harder task, we focus on using only this dataset in the following experiments as binary spam detection may be too trivial to solve and is less representative of a SOTA high-performing cloud teacher model.

4.1.2 Analysis of Framework using DP Noise

Next, we experiment and compare two different methods to preserve privacy, using DP-noise verses using a public dataset. As expected, the smaller we set the ϵ, the tighter our privacy budget and the more the model performance degrades. In our experiments, although not obvious from the graphs, we also find that when ϵ is under a certain value, all tokens in the private dataset get replaced and the student accuracy does not further degrade. This is because as we use text sanitization [22] to add DP-noise, we set a certain percentage of our tokens as sensitive tokens and the remaining as insensitive tokens based on how frequent they appear in our dataset. When ϵ is low enough, all sensitive tokens get replaced by insensitive tokens. In our experiments, with the 85% least frequent words as the sensitive tokens, we find that they only consists of $\sim$ 6% of the total amount of words. Therefore, when ϵ is really small, only $\sim$ 6% of the words get replaced and the accuracy doesn't degrade further. However, it is worth noting that this epsilon threshold is dataset dependent. The emotions dataset we are using consists of colloquial messages, which typically have very few rare words. In contrast, if we were to use a dataset that is related to medical diagnosis or more technical fields, we may find that 85% of the tokens make up a larger portion of the total dataset. Therefore, if one wishes

to extract a model for a different task from the cloud model with a different dataset, they would need to tune this parameter.

For the differential privacy based method, we also see that the final accuracy for the fine-tuned student is upper bounded by the teacher accuracy after all iterations in both cases, whether the student is fine-tuned with noisy data or with non-noised data. Although the student has a high accuracy at the start for the scenario using non-noised data, its performance degrades with more data points. In contrast, we see that the student fine-tuned with noisy data does not experience performance overshoot, but is always bounded by the teacher performance. This could show that knowledge distillation is more consistent when the student is fine-tuned with noisy labels and noisy data points - the exact outputs and inputs from the teacher. The consistency allows the student to closely follow the teacher behavior through the fine-tuning process such that the student is always upper-bounded by the teacher. In contrast, the use of non-noised data with noisy points for the student introduces additional uncertainties into the system. The inconsistency combined with the use of active learning may have resulted in the student sampling points that initially rapidly improved performance, potentially overfitting on these earlier data points sampled. Therefore, when the remaining data points are introduced, the student performance drops and gets closer to the teacher performance. For future experiments, we may want to used noisy labels and noisy data for fine-tuning the student to have more stability in the student performance. Additionally, sampling strategies other than entropy sampling can also be explored to see how different modes of active learning influence student performance.

4.1.3 Analysis of Framework using Public Dataset

While using DP noise to preserve privacy significantly degrades student performance when the privacy budget is tight, using a public dataset to construct the insensitive dataset allows the student to achieve similar accuracy to the teacher performance through fine-tuning, even in cases where the public dataset has a domain shift. Without a domain shift, the student performance using the public dataset is comparable to when using an ϵ of 1000, which adds

almost no noise to the private data points. In cases where we construct an insensitive dataset using both spam data and emotions data for the emotions classification task, the students fine-tuned with less emotions data increase in performance less rapidly. However, at the end of fine-tuning, the student accuracies are similar, with only $\sim 3\%$ difference between the student trained with an insensitive dataset that consists of 100% emotions data and the student trained with an insensitive dataset that consists of 25% emotions data.

This implies that we can potentially use a very large publicly available dataset that doesn't fully overlap with the teacher input space while still getting decent results comparable to when the dataset is small but a good representation of the teacher input space. However, it is worth noting that because the accuracy increases more slowly when our insensitive dataset contains less domain-specific data, if there is too little relevant data, then it would take longer and much more data to fine-tune the student. Therefore, if the user intends to minimize querying costs and time, it is better to find public datasets that are similar to the teacher input space.

However, if we want to minimize cost but cannot find a dataset that is representative enough of the teacher input space, we may consider a third method to construct the insensitive dataset, such as using generative models to create synthetic differentially private datasets. Recently, there has been work focused on generating differentially private datasets to increase the amount of training data available [79, 80]. However, more work needs to be done to optimize these generative models such that they can be implemented in a low cost manner.

4.1.4 Analysis of Framework using both Public Dataset and DP Noise

We then investigated a fourth approach to construct the insensitive dataset: combine the noisy private dataset and the public dataset. The performance of the student using a combination of datasets is more similar to the performance of its majority composition. In the case where the ratio is 1:1, the student accuracy first overshoots and then drops as more data is used. However, we do see a more obvious increasing trend in the later iterations that allow the students to achieve accuracies that are no longer bounded by the teacher. Therefore, in the case where a

user is able to find a small but representative public dataset, it may be beneficial to combine the public and noisy private dataset as the insensitive dataset such that there is enough data to fine-tune a student efficiently to the desired performance. However, one must be careful about how the data points are sampled from our insensitive dataset. If we directly use active learning on the combined dataset, it may only pick the noisy or non-noised data points first, thus introducing bias and potentially overfitting to one of them. Therefore, we sample from the public portion and the noised private portion separately and then combine the sampled data points to query the teacher, eliminating potential active learning sampling bias.

One may also wonder if there is a sampling bias when we use DP-noised samples for active learning. In our experiments, while we run tests that either fine-tune the students with non-noised data points or noisy data points, we perform entropy sampling only on the non-noised data points. This stops the DP-noise biasing which data points to be chosen from entropy sampling. If we performed entropy sampling on noisy data points, entropy sampling may try to choose data points that have many tokens swapped, ones that are more noisy. This could lead to undesirable results and introduces unwanted biases in the framework.

4.1.5 Effects of Student Changing Hyperparameters

The experiments from Section 3.3.2 to 3.3.6 all used the same hyperparameters and pre-trained student model as outlined in the experimental setup and Appendix A. We perform experiments to study how varying the pre-trained student model affects the fine-tuning performance. Two parameters were investigated: pre-training bootstrap size and the number of pre-training epochs. When we pre-train with less data, the student starts at a lower accuracy in fine-tuning. Furthermore, the student pre-trained on less data increases in accuracy more slowly. However, similar to the case when there is a domain shift in the public data, while the students are pre-trained with different amounts of data, all of them gradually plateau and reach similar accuracies at the end of fine-tuning. Therefore, if we train for enough iterations, even if the student was pre-trained with less data, it can still reach a good performance. However, if there is more data available, to minimize querying costs, we should pre-train the student with more

data such that it can use less time to reach a high accuracy.

When the amount of pre-training epochs is varied, we see no obvious trend that is dependant on the number of epochs. While there is a drastic difference in student starting performance, students pre-trained with 3, 5, and 8 epochs soon have similar accuracies and follow the same increasing trend. This could mean that it is not necessary to pre-train the student for a lot of iterations. To save compute and time, we can try to minimize the amount of epochs the student needs to pre-train. However, this test was only done with the original setup where the bootstrap size is 70k. There could be more variation in performance if we use a smaller bootstrapping size since the student may not be able to generalize well when exposed to less data. However, using our current results, we find that it is more important for the student to have seen more examples than being trained to a higher accuracy through more epochs in pre-training.

4.2 Ethical and Legal Concerns

As our privacy preserving inference framework is inspired by model extraction techniques, there are concerns surrounding whether this framework can be exploited for malicious purposes or not. One key distinguishing factor that separates our framework from model stealing is that our framework extracts a partial model. For all our experiments, we only learn to classify 3 out of 6 classes. The effectiveness of extracting all functionality has not been tested. Although with the current implementation, it is not hard to change the setup to learn to classify all classes, the results with that modification are still simplified versions of real life scenarios, providing a proof-of-concept on the framework effectiveness.

While our teacher model only performs emotions classification, an actual cloud model may be able to generalize across different tasks and even be multi-modal. For instance, ChatGPT is a model that generalizes across NLP tasks and is now also compatible with the image domain. It is true that one can potentially use our framework to extract multiple partial local models from a cloud model. However, they would need to develop new methods to merge and aggregate the functionality of all of the partial models. Furthermore, it is not clear that our method is

necessarily more cost efficient for the purpose of model extraction since it is not designed as an attack.

While there is yet a systematic set of policies and laws surrounding AI products, many AI companies such as OpenAI and Anthropic have their own terms of usage. These terms of usage outline what users are prohibited from doing. Relating back to legal concerns about model extraction, OpenAI users cannot ”automatically or programmitically extract data or output...to develop models that compete with OpenAI” [81]. Similarly, Anthropic also outlines that users may not access or use the service ”to develop any products or services that supplant or compete with our Services” [82]. Our framework is designed to extracts partial functionality. While there is a possibility that a malicious user wants to use our framework to extract full cloud model functionality, it may not be nearly as efficient as using model extraction specific attacks. Furthermore, Antrhopic also outlines that users may not ”decompile, reverse engineer, disassemble, or otherwise reduce our Services to human-readable form, except when these restrictions are prohibited by applicable law” [82]. As our framework extracts an approximate partial model, we do not attempt and cannot to get exact parameters or weights, thus not violating the outlined terms. All in all, we acknowledge that users may try to use the framework for harmful purposes, but there is a small likelihood of them adapting it since the framework is not optimized for model extraction purposes. Arguably, with enough effort, any technology can be used for malicious intents.

Chapter 5

Conclusion

In this thesis, we present a privacy preserving inference framework that takes inspiration from model extraction techniques. Specifically, it uses knowledge distillation and active learning. Firstly, our experiments show that cosine-similarity is not always able to find similar data points when the vectors representations used are not good enough. Furthermore, our experiments confirm the degradation in model performance when we add more differential privacy noise. In contrast, we find that using a public dataset with the inference framework can extract partial student models with high performance, even when the public dataset we use to query the teacher has a domain shift. In addition, we find that depending on the type and amount of data available to the user, it may be beneficial to fine-tune the student on a combination of DP-noised private and public data. Finally, during the student pre-training, to optimize fine-tuning, one should try to use as much data as possible from what is available but train for fewer epochs.

All in all, we first presented a privacy preserving inference infrastructure that utilizes model extraction techniques to protect user data sent to NLP MLaaS. Then, we performed experiments to test the framework and showed that it is able to preserve privacy while having low performance degradation. Finally, we provide analysis and discussion on when the framework is applicable and how the users should use it.

Bibliography

[1] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, "Mlaas: Machine learning as a service," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 896–902, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:206823005

[2] A. B. Nassif, M. A. Talib, Q. Nasir, H. Albadani, and F. M. Dakalbab, "Machine learning for cloud security: A systematic review," *IEEE Access*, vol. 9, pp. 20 717–20 735, 2021.

[3] R. Bianchini, M. Fontoura, E. Cortez, A. Bonde, A. Muzio, A.-M. Constantin, T. Moscibroda, G. Magalhaes, G. Bablani, and M. Russinovich, "Toward ml-centric cloud platforms," *Commun. ACM*, vol. 63, no. 2, p. 50–59, jan 2020. [Online]. Available: https://doi.org/10.1145/3364684

[4] A. Y. Hannun, C. Guo, and L. van der Maaten, "Measuring data leakage in machine-learning models with fisher information," *CoRR*, vol. abs/2102.11673, 2021. [Online]. Available: https://arxiv.org/abs/2102.11673

[5] M. Al-Rubaie and J. M. Chang, "Privacy preserving machine learning: Threats and solutions," *CoRR*, vol. abs/1804.11238, 2018. [Online]. Available: http://arxiv.org/abs/1804.11238

[6] P. Rao, S. Krishna, and A. Kumar, "Privacy preservation techniques in big data analytics: a survey," *Journal of Big Data*, vol. 5, 09 2018.

[7] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. A. DePristo, K. Chou, C. Cui, G. S. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature Medicine*, vol. 25, pp. 24 – 29, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:205572964

[8] R. A. Berk, "Artificial intelligence, predictive policing, and risk assessment for law enforcement," *Annual Review of Criminology*, vol. 4, no. 1, pp. 209–237, 2021. [Online]. Available: https://doi.org/10.1146/annurev-criminol-051520-012342

[9] OpenAI, J. Achiam, and S. A. et al, "Gpt-4 technical report," 2024.

[10] R. Anil, S. Borgeaud, Y. Wu, and et al, "Gemini: A family of highly capable multimodal models," 2023.

[11] B. C. M. Fung, K. Wang, and P. S. Yu, "Anonymizing classification data for privacy preservation," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 5, p. 711–725, may 2007. [Online]. Available: https://doi.org/10.1109/TKDE.2007.1015

[12] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 201–210.

[13] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS'16. ACM, Oct. 2016. [Online]. Available: http://dx.doi.org/10.1145/2976749.2978318

[14] N. Ponomareva, H. Hazimeh, A. Kurakin, Z. Xu, C. Denison, H. B. McMahan, S. Vassilvitskii, S. Chien, and A. G. Thakurta, "How to dp-fy ml: A practical guide to machine learning with differential privacy," *Journal of Artificial Intelligence Research*, vol. 77, p. 1113–1201, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1613/jair.1.14649

[15] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, p. 211–407, aug 2014. [Online]. Available: https://doi.org/10.1561/0400000042

[16] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No, "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," 2021.

[17] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 331, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:2259147

[18] X. Gong, Q. Wang, Y. Chen, W. Yang, and X. Jiang, "Model extraction attacks and defenses on cloud-based machine learning models," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 83–89, 2020.

[19] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A survey of deep active learning," 2021.

[20] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, p. 1789–1819, Mar. 2021. [Online]. Available: http://dx.doi.org/10.1007/s11263-021-01453-z

[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[22] X. Yue, M. Du, T. Wang, Y. Li, H. Sun, and S. S. M. Chow, "Differential privacy for text analytics via natural text sanitization," 2021.

[23] J. Wang and Y. Dong, "Measurement of text similarity: A survey," *Information*, vol. 11, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/9/421

[24] ——, "Measurement of text similarity: A survey," *Information*, vol. 11, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/9/421

[25] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," 2019.

[26] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. K. Shevade, and V. Ganapathy, "Activethief: Model extraction using active learning and unannotated public data," in *AAAI Conference on Artificial Intelligence*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:213157375

[27] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," *CoRR*, vol. abs/1810.00069, 2018. [Online]. Available: http://arxiv.org/abs/1810.00069

[28] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroSP)*, 2018, pp. 399–414.

[29] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases*, H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 387–402.

[30] G. Caruana and M. Li, "A survey of emerging approaches to spam filtering," *ACM Comput. Surv.*, vol. 44, no. 2, mar 2008. [Online]. Available: https://doi.org/10.1145/2089125.2089129

[31] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, p. e01802, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844018353404

[32] S. Thielman, "Yahoo email surveillance: Who approved the secret scanning?" 2016.

[33] C. Savage and N. Perlroth, "ahoo said to have aided u.s. email surveillance by adapting spam filter," 2016.

[34] M. Hill, "The biggest data breach fines, penalties, and settlements so far," 2023.

[35] X. Wu, R. Duan, and J. Ni, "Unveiling security, privacy, and ethical concerns of chatgpt," 2023.

[36] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *CoRR*, vol. abs/1908.00080, 2019. [Online]. Available: http://arxiv.org/abs/1908.00080

[37] T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, and R. Buyya, "Machine learning (ml)-centric resource management in cloud computing: A review and future directions," *J. Netw. Comput. Appl.*, vol. 204, no. C, aug 2022. [Online]. Available: https://doi.org/10.1016/j.jnca.2022.103405

[38] [Online]. Available: https://cloud.google.com/tpu/docs/intro-to-tpu

[39] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin, "When machine learning meets privacy: A survey and outlook," *CoRR*, vol. abs/2011.11819, 2020. [Online]. Available: https://arxiv.org/abs/2011.11819

[40] T. Graepel, K. Lauter, and M. Naehrig, "Ml confidential: Machine learning on encrypted data," Cryptology ePrint Archive, Paper 2012/323, 2012, https://eprint.iacr.org/2012/323. [Online]. Available: https://eprint.iacr.org/2012/323

[41] A. Brutzkus, O. Elisha, and R. Gilad-Bachrach, "Low latency privacy preserving inference," *CoRR*, vol. abs/1812.10659, 2018. [Online]. Available: http://arxiv.org/abs/1812.10659

[42] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *CoRR*, vol. abs/1711.05189, 2017. [Online]. Available: http://arxiv.org/abs/1711.05189

[43] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/1999-5903/13/4/94

[44] A. Wood, K. Najarian, and D. Kahrobaei, "Homomorphic encryption for machine learning in medicine and bioinformatics," *ACM Comput. Surv.*, vol. 53, no. 4, aug 2020. [Online]. Available: https://doi.org/10.1145/3394658

[45] O. Masters, H. Hunt, E. Steffinlongo, J. Crawford, F. Bergamaschi, M. E. D. Rosa, C. C. Quini, C. T. Alves, F. de Souza, and D. G. Ferreira, "Towards a homomorphic machine learning big data pipeline for the financial services sector," Cryptology ePrint Archive, Paper 2019/1113, 2019, https://eprint.iacr.org/2019/1113. [Online]. Available: https://eprint.iacr.org/2019/1113

[46] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," *CoRR*, vol. abs/1412.7584, 2014. [Online]. Available: http://arxiv.org/abs/1412.7584

[47] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1310–1321. [Online]. Available: https://doi.org/10.1145/2810103.2813687

[48] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:207241585

[49] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the gan: Information leakage from collaborative deep learning," *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:5051282

[50] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3461939

[51] D. Yu, S. Naik, A. Backurs, S. Gopi, H. A. Inan, G. Kamath, J. Kulkarni, Y. T. Lee, A. Manoel, L. Wutschitz, S. Yekhanin, and H. Zhang, "Differentially private fine-tuning of language models," *ArXiv*, vol. abs/2110.06500, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:238743879

[52] X. Li, F. Tramèr, P. Liang, and T. B. Hashimoto, "Large language models can be strong differentially private learners," *ArXiv*, vol. abs/2110.05679, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:238634219

[53] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:50770883

[54] C. Xu, J. Ren, D. Zhang, and Y. Zhang, "Distilling at the edge: A local differential privacy obfuscation framework for iot data analytics," *IEEE Communications Magazine*, vol. 56, pp. 20–25, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52014193

[55] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16. USA: USENIX Association, 2016, p. 601–618.

[56] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," *CoRR*, vol. abs/1912.00888, 2019. [Online]. Available: http://arxiv.org/abs/1912.00888

[57] H. Jia, C. A. Choquette-Choo, and N. Papernot, "Entangled watermarks as a defense against model extraction," *ArXiv*, vol. abs/2002.12200, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211532649

[58] M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, "PRADA: protecting against DNN model stealing attacks," *CoRR*, vol. abs/1805.02628, 2018. [Online]. Available: http://arxiv.org/abs/1805.02628

[59] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 506–519. [Online]. Available: https://doi.org/10.1145/3052973.3053009

[60] J. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," *CoRR*, vol. abs/2011.14779, 2020. [Online]. Available: https://arxiv.org/abs/2011.14779

[61] D. Oliynyk, R. Mayer, and A. Rauber, "I know what you trained last summer: A survey on stealing machine learning models and defences," *ACM Computing Surveys*, vol. 55, no. 14s, p. 1–41, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1145/3595292

[62] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," *CoRR*, vol. abs/1812.02766, 2018. [Online]. Available: http://arxiv.org/abs/1812.02766

[63] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, "A comprehensive overhaul of feature distillation," *CoRR*, vol. abs/1904.01866, 2019. [Online]. Available: http://arxiv.org/abs/1904.01866

[64] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," *CoRR*, vol. abs/1907.09682, 2019. [Online]. Available: http://arxiv.org/abs/1907.09682

[65] A. Barbalau, A. Cosma, R. T. Ionescu, and M. Popescu, "Black-box ripper: Copying black-box models using generative evolutionary algorithms," 2020.

[66] R. G. Lopes, S. Fenu, and T. Starner, "Data-free knowledge distillation for deep neural networks," *ArXiv*, vol. abs/1710.07535, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:1844680

[67] B. Settles, "From theories to queries: Active learning in practice," 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:58935912

[68] J. Zhu, H. Wang, B. K. Tsou, and M. Ma, "Active learning with sampling by uncertainty and density for data annotations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1323–1331, 2010.

[69] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Exploring connections between active learning and model extraction," in *Proceedings of the 29th USENIX Conference on Security Symposium*, ser. SEC'20. USA: USENIX Association, 2020.

[70] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020. [Online]. Available: http://dx.doi.org/10.1016/j.physd.2019.132306

[71] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[72] "Huggingface."

[73] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[74] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[75] T. Almeida and J. Hidalgo, "SMS Spam Collection," UCI Machine Learning Repository, 2012, DOI: https://doi.org/10.24432/C5CC84.

[76] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen, "CARER: Contextualized affect representations for emotion recognition," in *Proceedings of the*

2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3687–3697. [Online]. Available: https://www.aclweb.org/anthology/D18-1404

[77] X. Yue, H. A. Inan, X. Li, G. Kumar, J. McAnallen, H. Shajari, H. Sun, D. Levitan, and R. Sim, "Synthetic text generation with differential privacy: A simple and practical recipe," 2023.

[78] F. Mireshghallah, Y. Su, T. Hashimoto, J. Eisner, and R. Shin, "Privacy-preserving domain adaptation of semantic parsers," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 4950–4970. [Online]. Available: https://aclanthology.org/2023.acl-long.271

[79] X. Yue, H. Inan, X. Li, G. Kumar, J. McAnallen, H. Shajari, H. Sun, D. Levitan, and R. Sim, "Synthetic text generation with differential privacy: A simple and practical recipe," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 1321–1342. [Online]. Available: https://aclanthology.org/2023.acl-long.74

[80] F. Mireshghallah, Y. Su, T. Hashimoto, J. Eisner, and R. Shin, "Privacy-preserving domain adaptation of semantic parsers," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 4950–4970. [Online]. Available: https://aclanthology.org/2023.acl-long.271

[81] "Terms of usage," 2024. [Online]. Available: https://openai.com/policies/terms-of-use

[82] "Terms of usage," 2024. [Online]. Available: https://www-cdn.anthropic.com/files/4zrzovbb/website/e2d538c84610b7cc8cb1c640767fa4ba73f30190.pdf

Appendix A

Experiment hyperparameters

We use the following hyperparameters to fine-tune our student:

- dropout: 0.1
- learning rate: 0.0005
- weight decay rate: 0.0001
- batch size: 128
- iterations: 10
- epochs per iteration: 8

Appendix B

Data Noised with Text Sanitization

Example 1

Original

i do feel **kris** allen is an incredibly talented young man and i do respect him for that adam **lambert** is a one of a kind artist and clearly head and shoulders above **kris** s john **mayer** imp ##erson ##ation

Noised with ϵ=1000

i do feel **likes** allen is an incredibly talented young man and i do respect him for that adam **complete** is a one of a kind artist and clearly head and shoulders above **naughty** s john **somehow** imp ##erson ##ation

Example 2

Original

i can see **adam thor** ##es ##by **saving david fern ##ley** from the press **gang gene ##tta turner** involved in the **jet** industry and i feel for **eliza** and john mitchell when they **inherit** an **estate** in **cornwall provided** they leave their beloved **broom ##field** manor close to

Noised with ϵ=0.1

i can see **ago ate** ##es ##by **empty wonderful progress energetic** from the press **community j activities under** involved in the **hanging** industry and i feel for **prepared** and john mitchell when they **##me** an **##uous** in **god calling** they leave their beloved **runs ta** manor close to