Recovering Utility in LDP Schemes by Training with Noise^2

by

Kexin Li

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

Graduate Department of Electrical & Computer Engineering
University of Toronto

Recovering Utility in LDP Schemes by Training with Noiseˆ2

Kexin Li
Master of Applied Science
Graduate Department of Electrical & Computer Engineering
University of Toronto
2024

Abstract

The adoption of large cloud-based models for inference has been hampered by concerns about the privacy leakage of end-user data. One method to mitigate this leakage is to add local differentially private noise to queries before sending them to the cloud, but this degrades utility as a side effect. Our key insight is that knowledge available in the noisy labels returned from performing inference on noisy inputs can be aggregated and used to recover the correct labels. We implement this insight in LDPKiT, which stands for *Local Differentially-Private and Utility-Preserving Inference via Knowledge Transfer*. LDPKiT uses the *noisy* labels returned from querying a set of *noised* inputs to train a local model (*noiseˆ2*), which is then used to perform inference on the original set of inputs. Our experiments on CIFAR-10, Fashion-MNIST, SVHN, and CARER NLP datasets demonstrate that LDPKiT can improve utility without compromising privacy. For instance, on CIFAR-10, compared to a standard ϵ-LDP scheme applying the Laplacian noise on each data sample with $\epsilon = 15$, which provides a weak privacy guarantee, LDPKiT can achieve similar accuracy with $\epsilon = 7$, offering an enhanced privacy guarantee. Moreover, the benefits of using LDPKiT increase at higher, more privacy-protective noise levels. For Fashion-MNIST and CARER, LDPKiT's accuracy on the sensitive dataset with $\epsilon = 7$ not only exceeds the average accuracy of the standard ϵ-LDP scheme with $\epsilon = 7$ by roughly 20% and 9% but also outperforms the standard ϵ-LDP scheme with $\epsilon = 15$, a scenario with less noise and minimal privacy protection. We also perform *Zest* distance measurements to demonstrate that the type of distillation performed by LDPKiT is different from a model extraction attack.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. David Lie, for his guidance and support. I am more than grateful to have learned about the knowledge of security and privacy, and research skills from you. Thank you for your advice and support in all aspects during the past two years of research and life. It has been an incredible journey.

I would like to thank my oral examination committee members, Prof. Frank Kschischang, Prof. Nicolas Papernot and Prof. Shurui Zhou, for all their valuable time and feedback.

I would like to thank Prof. Aastha Mehta and Yang Xi for all their technical support and advice. It has been my honor to collaborate with you.

I would like to express my gratitude to all of my TossLab group members, especially Wenjun (Wendy) Qiu, for all the valuable feedback on my research and warm-hearted company. Thank you, Yuqin, Xiangyu, Guozhen, Nathan, Shirley, Stanley, Joe, John, Wei, Shawn, and Eric, for your suggestions, feedback, and support in my research and life. Especially thank you for showing up in person for my dry-run presentation, even though Toronto had a pouring rain and thunderstorm warning on that day. I felt so flattered and loved.

I would also like to thank my forever chillest yet warmest friends at my favorite Muay Thai gym with coaches Erica, Kaite, Nasib, Burak, Tom, and Eric, and my dearest friends, Lara, Natália, Andriea, *etc.,* as well as all my "choking" friends at my Brazilian Jiu-Jitsu dojo. Thank you for helping me become a physically and mentally strong girl and always supporting me whenever needed.

I am incredibly grateful to my parents for their unconditional and detailed support and caring throughout my entire childhood and my journey as a student and researcher.

Last but not least, I would like to thank my eternal love, Qingnan Yu, for always being my supportive and dependable partner. Thank you for cheering me up when I was at my lowest, for supporting my hobbies, and for unconditionally respecting my choices. Your love and support make this work possible.

Contents

List of Tables

List of Figures

List of Acronyms

AL Active Learning

AI Artificial Intelligence

ANN Artificial Neural Network

CE Cross-Entropy

CNN Convolutional Neural Network

DNN Deep Neural Network

DP Differential Privacy

FL Federated Learning

FN False Negative

FP False Positive

GAN Generative Adversarial Network

GDP Global Differential Privacy

KLDiv Kullback–Leibler Divergence

LDP Local Differential Privacy

LLM Large Language Model

RNN Recurrent Neural Network

SIDP Standard Inference with LDP

ML Machine Learning

MLaaS Machine Learning as a Service

MSE Mean Squared Error

NN Neural Network

TN True Negative

TP True Positive

Chapter 1

Introduction

1.1 Introduction

Cloud providers, such as Google, Amazon, and Microsoft, offer Machine Learning as a Service (MLaaS) [1], which enables the use of large, feature-rich Machine Learning (ML) models in several privacy-sensitive applications, such as personalized medicine and medical imaging [2], mobile healthcare apps, and surveillance [3]. At the same time, privacy concerns surrounding the MLaaS platforms arise during the widespread adoption and use of MLaaS. While users have privacy concerns regarding both training and inference stages of MLaaS platforms [4]–[7], in this work, we focus on privacy concerns with regard to the inference services.

A malicious MLaaS provider can monitor or inspect a user's queries during inference and use the information for purposes the user did not consent to. In 2016, for instance, Yahoo secretly complied with the US government's digital communication surveillance and used its custom spam and child pornography detection system to monitor users' emails [8]–[10]. More recently, the Amazon Ring Doorbell was found to have disclosed users' video and audio footage to the police without authorization [11]. Ever since Large Language Models (LLMs) have gained popularity, many organizations have banned their members from using them due to the fear of data leakage [12], [13]. Even with a trusted provider, a compromised platform could enable an adversary to infer users' queries via side-channels [14].

To tackle these privacy concerns, previous work has proposed applying homomorphic encryption schemes [15], [16] or hardware-enforced trusted execution environment [17] to protect inputs during inference. However, both schemes are primarily designed to protect against a malicious platform and implicitly trust the ML model provider—a malicious model can still leak arbitrary information about its inputs and would require some form of auditing. In addition, hardware-based schemes are prone to side-channel attacks [18]–[20], while homomorphic schemes can impose large overheads. An alternative to these schemes, which does not trust the entire model or platform and imposes virtually no performance overhead, is Local Differential Privacy (LDP), which adds random LDP-provable noise to each of the user's queries before transmitting them to the cloud for inference [21]. However, this "standard" application of LDP noise to the ML model's inputs results in noise in the model outputs, leading to loss of utility.

We propose LDPKiT, which stands for *Local Differentially-Private and Utility-Preserving In-*

ference via Knowledge Transfer, a privacy-protective framework that recovers utility by training a local model. In standard applications of LDP, each query is independent, and information from the model's prediction on one query does not help improve its predictions for others. Rather than being content with the loss of utility of each query individually, LDPKiT records both the (noised) query inputs and the (noisy and erroneous) labels returned from the cloud model (*i.e., noiseˆ2*) to form a private training set, which is used to train a local model. As a result, LDPKiT's training of a local model enables it to leverage collective knowledge from a batch of queries to improve the overall utility. However, if the local model is trained on too few points, its accuracy may not be sufficient to exceed that of using the labels directly from the standard usage of LDP. Thus, there exists a lower bound on the number of queries the user is willing to make for LDPKiT to be practical. Finally, one might wonder whether, as the number of points used to train the local model increases, it will ever become competitive with the cloud model—in other words, is this a type of model extraction attack? To study this, we also compare the local model with the cloud model and measure their model distances using the *Zest* framework [22].

Our analysis is guided by the following research questions:

RQ1. Does LDPKiT recover utility impacted by LDP noise?

RQ2. How does the number of queries impact LDPKiT?

RQ3. How does LDPKiT differ from an adversarial model extraction attack?

1.2 Motivation

As discussed above, we focus on inference data privacy protection. Here are some real-world examples that motivate the thesis.

1.2.1 Image/Video

Once the personal image is uploaded to the Internet, the action cannot be revoked, and you have no control over its future usage [23]. Demonstrated by the news about Pope Francis in 2023 [24], as generative models and AI Deep Fake technology become more sophisticated, it is difficult for a human to distinguish fake from the truth. Not to mention how threatening it becomes if the adversary intends to do something bad with the user's personal photo or video, such as fraud and blackmail [25]. For example, when the user enables FaceID on her mobile devices, she must scan her face and disclose the facial information to device manufacturers. While the intended use of this facial data is to authenticate and unlock the device, there is a risk that third-party manufacturers can misuse this information, potentially even sell user data, for profit if they have ill-founded privacy policies [26], [27]. A more serious scenario is when companies release their users' private information to the government for "political or socially beneficial uses". One infamous example is that Amazon was revealed to have shared Ring's video doorbell footage with the police without users' permission in 2022 [11]. The government may ask service providers for unauthorized information release and collect customers' biometrics without consent. In the above cases where the user needs to provide facial data to access personalized services, such as FaceID or smart door locks, our method safeguards the user's identity. Our approach protects the user's data privacy by providing an LDP privacy guarantee before data transmission to the cloud for computation. This means that the ML models

deployed on the cloud only have access to perturbed images, eliminating the necessity of sharing the user's actual photos with any third party.

1.2.2 Audio

Similarly, voice is considered sensitive personal information that can reveal one's identity. Telecommunication fraud, such as fraudulent calls that intend to record the user's voice for future abuse, can lead to severe financial loss. Traditional fraudulent calls aim to deceive the user into providing her bank account information for monetary purposes; however, more advanced telecommunication scams succeed as long as the user responds. The scammer can capture the user's unique voice signature, and just like DeepFake, the scammers can impersonate the user and authorize fraudulent charges, and even conduct criminal activities such as blackmailing the user's close contacts [28], [29]. As smart home devices gain popularity, more people use them and enjoy the customizability and convenience these services bring. Using voice-controlled virtual assistants imposes privacy risks, given that they can record a user's voice and even recognize the speaker. Many motion detection services also have complimentary audio capture capability. In benign use cases, additional audio functionalities are appreciated (*e.g.,* theft prevention and car accident analysis). However, another criticism of the Amazon Ring doorbells scandal is that the footage provided to the police did not have automatic audio recording eliminated [11]. Therefore, the government also obtained the customers' audio identity information without consent. LDPKiT has not been tested on the audio modality and we leave it as future work.

1.2.3 Text

Email spam checkers detect malicious and viral emails and block their delivery to inboxes. However, it is theoretically possible for an email service or spam checker to eavesdrop on the message content, store it without the user's awareness, and use it for purposes other than spam detection. Spam checks are generally conducted in the cloud rather than locally on a user's device for multiple reasons, *e.g.,* storage and processing power limitation, real-time synchronization, and email filtering infrastructure [30]. In 2016, multiple sources [8]–[10] reported that Yahoo's customized system, which was intended for spam and child pornography detection, secretly complied with the US government's digital communication surveillance. Yahoo spied on and scanned all the incoming emails according to the government's demands. As digital communication grows, it becomes common for government agencies like the FBI to ask tech companies for users' messages. If companies such as Yahoo decide to obey their orders, severe privacy breaches will happen. Unfortunately, data leakage had already happened by the time this action was discovered and reported. Applying our technology safeguards sensitive information without compromising the utility of the main task. In the text scenario, LDPKiT protects the privacy of target email content by querying the remote model with a sanitized email instead of the original one.

1.3 Contribution

In summary, this thesis makes the following contributions:

- The design and implementation of LDPKiT are presented, the framework that incorporates knowledge transfer techniques for privacy protection.
- LDPKiT can achieve high inference accuracy and mitigate privacy risks for sensitive queries with the (noisy) knowledge gained from the cloud model on privacy-preserving (noised) queries.
- The evaluation of LDPKiT is conducted on image and text modalities with multiple models and datasets.
- The discussion and comparison of different strategies and experimental results are presented.

1.4 Thesis Structure

The following chapters of this thesis are organized as follows. Chapter 2 introduces some related background concepts in trustworthy machine learning and specific design techniques. Chapter 3 outlines the detailed design of LDPKiT. Chapter 4 evaluates and analyzes LDPKiT on image and text classification datasets with multiple representative models and demonstrates the difference of LDPKiT from an adversarial model extraction attack. Chapter 5 discusses some limitations and future directions of the work. Lastly, Chapter 6 draws a short summary of the thesis.

Chapter 2

Background and Related Work

2.1 Machine Learning

Machine Learning (ML) provides automated data analysis methods that detect relationships in the data and solve problems under uncertainty [31]. Based on the nature of the data available, ML tasks can be categorized into different classes, such as *supervised learning* when training inputs are labelled, *unsupervised learning* when unlabeled inputs are given, and *reinforcement learning* where an intelligent agent takes actions in a certain environment and records the new states and reward values, in order to optimize the cumulative reward as a result.

2.1.1 Neural Networks

A Neural Network (NN), also known as Artificial Neural Network (ANN), is a computational model inspired by biological neural networks. It has applications in ML and Artificial Intelligence (AI) for pattern recognition and complex problem solving such as classification, regression, and clustering [32]. A *neuron* is a computational node or unit that processes input data, computes with functions, and outputs results to other neurons. Each input to a neuron can be multiplied by a *weight* and associated with an additional *bias* term to indicate the input's significance. An NN consists of layers of neurons. A simple forward neural network (forward propagation) has an *input layer*, *hidden layer(s)*, and an *output layer*. The input layer receives the data input where each neuron represents an individual *feature* in the input data. Hidden layer(s) are the layers between the input and output layers that perform the computation. The output layer produces the final result(s). An *activation function* that introduces non-linearity to the model is applied at the end of each layer. Common examples include Sigmoid, Hyperbolic Tangent (Tanh), and Rectified Linear Unit (ReLU).

A Deep Neural Network (DNN) is an ANN with multiple hidden layers and a substantial amount of neurons, enabling them to learn complex and hierarchical data representations. Examples include Convolutional Neural Networks (CNNs) for image data processing, Recurrent Neural Networks (RNNs) for sequential data and natural language processing, and Generative Adversarial Networks (GANs) for synthetic data generalization.

2.1.2 Training and Inference in Machine Learning

Training and *inference* are two fundamental stages in the lifecycle of an ML model.

Training

Training refers to the process of teaching an ML model to make predictions on pre-collected data (*i.e.,* input). The input is usually in a vector form known as *features*. Choices of ML models include NNs, Support Vector Machines, Linear Regression models, *etc.* The *hypothesis space* in an ML problem is the set of all possible functions/hypotheses that a *learning algorithm* considers as potential solutions. The space is parameterized by a vector representing the *parameters* or *weights and biases* of an ML model. By utilizing the *training data*, the goal of this learning algorithm is to solve an *optimization* problem and find the best hypothesis that approximates the actual relationship between input features and the target output. *Gradient*, a vector that represents the direction and slope of a *function*, plays a crucial role in model training. In ML, this function is usually a *loss function* that quantifies the difference between the predictions and expected outputs of the training data. Examples of common loss functions are Cross-Entropy (CE) Loss for classification tasks and Mean Squared Error (MSE) for regression tasks. In a Supervised Learning procedure, for instance, the model can iteratively adjust its parameters to minimize errors in its predictions based on the gradient information of a loss function. The gradient of a loss function with respect to each parameter is calculated by taking the partial derivatives of the loss function with respect to each parameter. We can then evaluate the model's performance using labelled *test data*, which is exclusive from the training data, to test whether the model generalizes to unseen data points during the training process. In a multi-class classification scenario, we train the ML model to classify each input into one of the problem's classes. We record model accuracy by counting the proportion of predictions that match the actual class labels in the test data.

For example, training an NN in a Supervised Learning setting involves five stages:

1. Forward Propagation: Obtain the predicted output from the received input data.
2. Loss Calculation: Compute the training loss using a loss function on a labelled dataset.
3. Backward Propagation: Calculate the gradients of loss with respect to the model parameters (weights and biases).
4. Gradient Descent: Update the model parameters with the calculated gradients, aiming at loss minimization.
5. Iteration: Repeat steps one to four until loss converges or a stopping condition is met.

Inference

Once the ML model is trained, it can be deployed to make predictions about new, unseen input data by applying its learned parameters. For example, in a multi-class classification task, the output of the ML model is the predicted class label of each input. The prediction can be in different forms. *Hard label* refers to the most likely class the model thinks the input belongs to. *Soft label* refers to a vector of probabilities representing the likelihood for the input to each class of the classification problem. *Top-k prediction* refers to the top k highest-probability predictions made by the model

based on an input. *Confidence value/score* is a presentation of top-k prediction that indicates how certain the model is about the predictions. The confidence value is often expressed as a probability. A hard-label prediction can also be referred to as a *top-1 prediction*.

2.1.3 Knowledge Distillation

Knowledge distillation [33], [34] is a subset of model compression techniques, aiming to reduce the ML model's size and computational cost while maintaining its accuracy. The goal of knowledge distillation is to save computational resources and costs by distilling a large teacher model into a relatively smaller student model with comparable performance. It can be categorized based on the knowledge form: response-based knowledge, feature-based knowledge, and relation-based knowledge [34]. *Feature-based knowledge distillation* encourages the student to learn from both the output of the final layer and the outputs of intermediate layers, such as feature maps [35], [36] and attention maps [37]. *Similarity-preserving knowledge distillation* measures the similarity between teacher and student models' activation maps and reflects it on distillation loss to boost the distillation effectiveness [38]. *Relation-based knowledge distillation*, on the other hand, studies the relationships between different feature maps among model layers [39]. However, these two common knowledge distillation techniques require white-box access to the teacher model's internals, which differs from our scenario (*i.e.,* the teacher/cloud model is proprietary). *Response-based knowledge distillation* aims to train a lightweight student model to mimic a sophisticated and complex teacher model's behavior based on the teacher's response from the last output layer (*i.e.,* soft labels). Meng, *et al.* [40] proposed a conditional knowledge distillation where students can selectively learn from ground-truth labels to boost accuracy. However, knowing the actual label is a hard assumption and differs from our problem setting. Knowledge distillation techniques can also be used adversarially, becoming model extraction, which is covered in Section 2.2.1.

2.1.4 Machine Learning as a Service

Machine Learning as a Service (MLaaS) refers to a cloud computing service provided by service providers for profit. Examples of cloud service providers include Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and IBM Watson. They offer a variety of ML tools for data processing, training, and inference/deployment phases with accessible monitoring and management interfaces, allowing users to use sophisticated ML features without the need for substantial investment in their own infrastructure development. For inference support, cloud service providers can offer query API access to pre-trained models spanning different ML tasks and charge users on a per-query basis. Depending on the amount of information they are willing to disclose, providers may opt to reveal details about the model architecture and confidence scores for interpretability or simply provide the final hard labels [41].

2.2 Trustworthy Machine Learning

In the field of information security, the CIA triad (*confidentiality*, *integrity*, and *availability*) is fundamental to creating and maintaining secure data and systems. Similar concepts also apply to ML-based systems. Confidentiality ensures that the information is only accessible to authorized

entities. Integrity ensures that the information is unaltered, accurate, and trustworthy over its entire lifecycle. Availability ensures that the information and resources are accessible to authorized entities whenever needed. In ML-based systems, the information and resources include data and ML models involved.

2.2.1 Model Confidentiality

Model Extraction

Model extraction/stealing is a realistic adversarial attack that infringes the intellectual property (IP) of ML models, where the attacker reproduces a model by stealing the model's parameters, decision boundaries, or functionalities. It can be done via a prediction query interface [42]. Model extraction attacks become a real concern with the prevalence of MLaaS systems. The motivations for the adversaries to steal a model are often cost-driven, *i.e.,* when the cost of training their own model exceeds the cost of model extraction in order to reach the same performance. The attacker aims to replicate the victim model's high performance with a minimal number of queries, ensuring the theft requires less effort than training a new model from scratch. A representative work of model extraction is proposed by Tramèr, *et al.* [42]. It aims to create a replica of a high-performance and confidential model based on input-output information via a publically accessible API. This query-based stealing attack does not require access to the original model's parameters or architecture; hence, it demonstrates the feasibility of a model extraction attack. Many prior works demonstrate successful model extraction with various levels (*i.e.,* partial or zero) of knowledge on the victim model and training data (*i.e.,* black-box, data-free, hard-label setting) [43]–[47]. Related defenses include watermarking schemes that embed information into model parameters or training datasets for future ownership claim [48]–[51], fingerprint-based methods that extract model characteristics such as decision boundaries [52], [53], and verification-based proof of ownership for IP protection [54], [55]. Analyses of model extraction [41], [56] are also in active research. Rather than model compression or extraction/stealing, we incorporate knowledge transfer for privacy preservation in a non-adversarial manner to recover the utility loss brought by LDP. We also quantitatively demonstrate that LDPKiT differs from an adversarial model extraction attack in Section 4.4.

2.2.2 Data Confidentiality and Privacy

Privacy denotes an individual (or organization)'s ability to choose whether, when, and to whom the personal (or organizational) information is to be disclosed [57], [58]. It is complementary to the CIA triad, especially confidentiality.

A privacy breach happens when data existence or data content is exposed to an unauthorized party. Similar to ML model confidentiality, the occurrence of privacy-related attacks and defenses on data can also be classified into *training* and *inference* phases.

Training Data Privacy

A membership inference attack is a type of adversarial attack on ML models where the attacker aims to determine whether a specific data point exists in the (proprietary) training dataset used to train the model [7]. To exploit data content, a model inversion attack attempts to infer and reconstruct

the input data from the model's outputs [59]. Gradient-based data reconstruction attacks extract input data from the model's gradient information [60]–[62]. This type of attack requires some knowledge of the ML model's internals, so it is typically applied in a Federated Learning (FL) setting. As for countermeasures, differentially-private models (*e.g.,* with DP-SGD) can mitigate model inference attacks by their nature [7], [63]–[65]. Confidence masking [66], [67] and regularization [68]–[70] are also common defenses against membership inference attacks. A model inversion attack can be mitigated by adding noise or perturbing and rounding to confidence scores before making them available as outputs and minimizing the input-output dependency [71]. Gradient-based data reconstruction attacks in FL can be mitigated by secure aggregation [72], [73] with communication-efficient learning algorithms such as FedSGD and FedAVG [62], [74].

Inference Data Privacy

Privacy breaches during inference time (*i.e.,* ML models are trained and deployed) are more straightforward. The concerns come from an unauthorized third party having access to the inference inputs that contain sensitive information. The third party can be an honest-but-curious model owner, a malicious platform that deploys the ML model, or a compromised platform that may unintentionally leak private information to other adversarial attackers. As mentioned, this thesis focuses on the privacy concerns of inference data and aims to provide a privacy protection solution on inference data when using an ML model deployed on an honest-but-curious platform.

LDPKiT provides an inference data privacy protection solution with LDP noise injection to the sensitive data before data transmission. Other privacy protection techniques also exist during inference time. One class of privacy protection methods is data encryption with homomorphic algorithms, which suffers from high computational overheads [15], [16]. In contrast, LDPKiT is more efficient, as it does not require complicated computation to be performed for each query. Hardware-assisted inference in Trusted Execution Environments (TEEs) is another approach [17]. A TEE is a secure area within a processor that provides a safe environment for sensitive code execution and prevents unauthorized access. For instance, Slalom puts the computation in a TEE to address inference privacy on remote services [17]. However, Slalom does not protect against the risks of side-channel attacks. Since TEEs have access to the original data, privacy breaches can still happen if the attackers compromise the TEEs [19], [20], [75]. Side-channels are not a threat for LDPKiT since it does not transmit the original data, and any privacy leakage is bounded by the LDP noise.

2.2.3 Integrity and Availability

An ML model's integrity and availability can be compromised by attacking its training dataset. The goal is to induce undesired model behavior, especially model outputs (*e.g.,* low quality/accuracy and constrained access). The adversary can poison the training data by data injection (inject adversarial samples) or data manipulation (alter training data/label) to corrupt model training or slow down the computation speed [76], [77]. The adversary may also modify the model by logic corruption (tampering with the ML algorithm) [78]. Similarly, during inference time, the adversary can breach model integrity through evasion attacks by manipulating the input data, leading to incorrect predictions [79]. Additionally, energy-latency attacks can be employed to harm the model's

availability, which significantly increases the computational energy consumption, potentially causing a Denial of Service (DoS) for the deployed models [80]. Related defenses and analyses are in active research [81]–[84].

2.3 Implementation Background

2.3.1 Differential Privacy

According to Dwork [85], Differential Privacy (DP) refers to the insensitivity of model output on one data point, *i.e.,* addition or removal of one data point should not affect model performance. It is a rigorous mathematical notion of privacy that helps prove the existence of privacy in ML mechanisms.

DP can be used either locally [86] or globally [63]–[65], and both methods provide provable privacy guarantees.

Global Differential Privacy

Global Differential Privacy (GDP) is a privacy guarantee applied to the outputs of queries from a computation instead of the data samples themselves. Hence, a trusted curator or aggregator who has access to raw data is required in the setting to apply this privacy-preserving mechanism before publishing the sensitive data.

Definition 2.3.1. ϵ-Differential Privacy *A randomized algorithm A satisfies ϵ-Differential Privacy if, for any two input datasets, D and D' that differ at most one element and for any subset of output S, the following equation holds:*

$$\Pr[\mathrm{A}(\mathrm{D}) \in S] \leq e^{\epsilon} \cdot \Pr[\mathrm{A}(\mathrm{D}') \in S] \tag{2.1}$$

There also exists different DP variants [87]–[90]. For example, (ϵ, δ)-Differential Privacy is defined as follows [90].

Definition 2.3.2. (ϵ, δ)-Differential Privacy *A randomized algorithm A satisfies (ϵ, δ)-Differential Privacy if, for any two input datasets, D and D' that differ at most one element and for any subset of output S, the following equation holds:*

$$\Pr[\mathrm{A}(\mathrm{D}) \in S] \leq e^{\epsilon} \cdot \Pr[\mathrm{A}(\mathrm{D}') \in S] + \delta \tag{2.2}$$

In both equations, ϵ refers to the upper bound on the privacy loss. The smaller ϵ, the tighter the bound, and the stronger the privacy guarantee. δ in the second equation is a bias or relaxation term indicating the probability of accidental information leak. Hence, (ϵ, δ)-DP provides a more relaxed bound than ϵ-DP. One key property of DP is that the more robust the privacy guarantee a DP algorithm provides, the less utility (prediction accuracy) the model has.

Local Differential Privacy

The above section provides definitions for GDP. GDP shares original input data with a trusted data curator, which is responsible for applying noise to the aggregated data. In this case, the curator

has access to the original sensitive data. To remove this point of trust, LDPKiT uses an LDP mechanism. LDP is a more decentralized DP mechanism. It was first introduced by Warner [91] as a survey technique to deal with randomized responses. Now, it is widely used in a local setting where the user perturbs her private data before transmitting them to an untrusted server. This privacy-preserving mechanism does not require a trustworthy central curator, since the data points are anonymized at the source as part of the data collection and pre-processing.

Definition 2.3.3. ϵ-Local Differential Privacy. We define ϵ-LDP as follows [85]: A randomized algorithm $\mathcal{A}$ satisfies ϵ-LDP if for all pairs of values and all sets $\mathcal{S}$ of possible outputs, where $\mathcal{S} \subseteq \text{Range}(\mathcal{A})$,

$$\Pr[\mathcal{A}(v_1) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{A}(v_2) \in \mathcal{S}] \tag{2.3}$$

A lower ϵ value indicates a tighter bound of the equation and a stronger privacy guarantee.

GDP and LDP have different application scenarios. GDP is more suitable for a controlled environment with a trusted curator, whereas LDP is preferred when local data sources do not trust the curator and wish to control their own data privacy, or when data aggregation is not required. Generally, GDP leads to higher utility and prediction accuracy for the same level of privacy protection (*i.e.,* noise level) compared to LDP because noise is added to aggregated results rather than to each individual data point, which amplifies the perturbation in LDP. Nevertheless, a common challenge of DP schemes is to find a balance between utility and privacy.

LDP noise can be injected into different phases during inference. While we add ϵ-LDP noise to original inputs before offloading inference to the cloud, similar to [21], noise can also be injected into inference frameworks deployed on a split computation setting [92]–[95], where the DNN is partitioned between the cloud and edge devices. These schemes involve a white-box model, so noise can be added to intermediate representations, which is different from our setting.

2.3.2 Active Learning

Active Learning (AL) [96] is an ML strategy in which the learning algorithm interactively queries an *oracle* (*e.g.,* a human annotator) to generate a labelled training dataset. In our problem setting, the oracle is the untrusted cloud model. The goal is to train the model with fewer data points while maintaining performance; hence, the key is to select informative or representative instances to query and train the model. This learning technique is particularly useful when labelled data is scarce or expensive.

In general, AL can be categorized into three scenarios based on where the queried instances come from, *i.e.,* pool-based, stream-based selective sampling, and query synthesis. *Pool-based AL* selects query candidates from an existing data pool based on an information measurement score associated with each instance in the data pool. *Stream-based AL* can handle stream data, as the learner can immediately decide whether the upcoming data point should be labelled or discarded. Since the decision is on the fly, the setting is also called *online AL*, and decisions are made independently without comparison among different instances [97]. In the *query synthesis AL* scenario, the learner can generate its own data samples to label [98]. However, because the model needs to create samples based on the knowledge it has, it can introduce human-unreadable samples that are useless in training. Hence, this scenario is not as well-developed as the other two.

Among all scenarios, pool-based selective sampling is the most common case and aligns with our problem setting. The instances to be labelled are selected iteratively using a *querying strategy*. Different querying strategies calculate the information measurement score of instances (*i.e.,* informativeness and representativeness) differently. To rank samples based on their representativeness, querying strategies include *cluster-based sampling* [99] that utilizes cluster structure, *density-based sampling* [100] that considers the distribution and information density of instances and the expected loss on unlabeled data [101]. On the other hand, *uncertainty-based sampling* is the most studied sampling method for ranking instances based on their informativeness [102], [103]. Intuitively, the samples with higher classification uncertainty provide more information to the model. Under this category, there are three different strategies: *uncertainty sampling*, which selects the data point close to the decision boundary; *margin sampling*, which selects the data point whose top two prediction probabilities are close; and *entropy sampling*, which selects the data point with the largest output entropy value.

Definition 2.3.4. Entropy Sampling. *In our case, we use entropy sampling, since we have a multi-class problem. Mathematically, entropy can be represented by a function U such that:*

$$U_{\text{entropy}}(x) = -\sum_{i=1}^{N} p(x_i) \log p(x_i) \tag{2.4}$$

In this context, high entropy indicates that the predicted probabilities are distributed more evenly across different classes, suggesting higher uncertainty and an increased likelihood of being selected for querying. Training the model with the most uncertain instances allows it to learn more effectively with fewer samples compared to using *random sampling* (*i.e.,* random order of training data).

2.4 Loss Functions

As discussed in Section 2.1.2 above, the choice of loss function is important in model training. In this section, we introduce the loss functions used in our design. Since we assume a supervised multi-class classification task with a hard-label setting in our scenario, we use CE Loss in our implementation. We also extend our experiments to training with softmax values provided, so we incorporate Kullback–Leibler Divergence (KLDiv) Loss in the loss function as a variant.

2.4.1 Cross-Entropy Loss

Cross-entropy measures the difference between two probability distributions for a given random variable or set of events. In ML, it can be used to measure the difference between the ground-truth label and the model's predicted probability distribution. For a batch of N data samples and C classes, CE Loss is defined as follows:

Definition 2.4.1. CE Loss.

$$\text{CE}(y, \hat{y}) = -\frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{C} y_{ji} \log(\hat{y}_{ji}) \tag{2.5}$$

In this equation, y_{ji} refers to the ground-truth label for the j-th example in the i-th class. $\hat{y}_{ji}$

refers to the model's predicted probability for the j-th example in the i-th class, and typically, it is the softmax function output of the model under training. The goal is to minimize the difference between the predicted probabilities $\hat{y}$ and the ground truth labels y.

2.4.2 Kullback–Leibler Divergence Loss

Kullback-Leibler Divergence (KLDiv) loss is commonly used in knowledge distillation, which also measures the difference between two probability distributions. Specifically, it is used to measure the information loss by approximating one distribution with another. In LDPKiT, we use KLDiv loss when we study the effect of the remote model's softmax values on utility recovery.

For a batch of N data samples and C classes, KLDiv Loss is defined as follows:

Definition 2.4.2. KLDiv Loss with Temperature Scaling.

$$\mathrm{KL}(P_T \| Q_T) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} P_{T,j}(z_i) \log \left(\frac{P_{T,j}(z_i)}{Q_{T,j}(z_i)} \right) \tag{2.6}$$

where

$$P_{T,j}(z_i) = \frac{e^{z_j/T}}{\sum_{k=1}^{C} e^{z_k/T}} \tag{2.7}$$

and

$$Q_{T,j}(z_i) = \frac{e^{\hat{z}_j/T}}{\sum_{k=1}^{C} e^{\hat{z}_k/T}} \tag{2.8}$$

In a teacher-student knowledge distillation scenario, z_i refers to the teacher model's logits for i-th class, and $\hat{z}_i$ refers to the student model's logits for i-th class. The *temperature* parameter, denoted as T, is a scaling factor that softens the probabilities. The probability distributions P_T and Q_T are the temperature-scaled softmax outputs from the teacher and the student model, respectively. The goal of KLDiv loss is to make the student model's softened prediction probability distribution, Q_T, as close to the teacher model's softened probability distribution, P_T, as possible.

LDPKiT supports image and text modalities. The above loss functions are applied to image modality only. We omit the loss function choice for text modality as the details are discussed in [104].

2.5 Evaluation Metrics

A *Confusion Matrix* is particularly useful in the performance evaluation of a classification algorithm. It contains four components:

- True Positive (TP): The number of data samples where the classification model correctly predicts as the positive class.
- True Negative (TN): The number of data samples where the classification model correctly predicts as the negative class.
- False Positive (FP): The number of data samples where the classification model incorrectly predicts as the positive class

- False Negative (FN): The number of data samples where the classification model incorrectly predicts as the negative class

To evaluate the classification model performance with LDPKiT, we use the *accuracy* metric, which is the ratio of correct predictions over the total number of predictions. The accuracy metric indicates how often the classification model makes correct predictions, which is one of the performance metrics derived from the Confusion Matrix. Accuracy can be calculated using the formula: $\text{Accuracy} = \frac{\text{TP}}{\text{TP+TN+FP+FN}}$.

Additionally, as mentioned in Section 1, we use *Zest* distance [22] to demonstrate how LDPKiT differs from an adversarial model extraction attack. Zest utilizes Local Interpretable Model-Agnostic Explanations (LIME) [105] to create local linear approximations of models at each reference data point. By calculating the *Cosine Distance* between the weights of these linear approximations, Zest can compare the global behaviors of different models and identify the existence of model extraction. The detailed model extraction attack detection procedure will be discussed in Section 4.4 of Chapter 4.

Chapter 3

Design and Implementation

3.1 Threat Model

Our goals are to protect sensitive user queries when using an ML cloud service during inference time and to recover some accuracy loss due to privacy-protective LDP noise added to the queries. Traditionally, DP seeks to bind the probability of an adversary being able to distinguish whether or not a user's contribution is present in the results of a query, providing a level of privacy protection for the user. In our scenario, the user is a customer of the cloud service, so the user's identity is already known to the cloud service provider, and so is the source of all the data the user submits. In addition, the user's data is not used for training and will not be made part of any dataset, so there is no threat of a membership inference attack on the user by an adversary querying the cloud model. Instead, our goal is to bound the probability that an attacker, *e.g.,* the cloud provider, can infer whether the content of a user's query (*i.e.,* one data point in the user's sensitive dataset) matches that of a particular value. This provides a form of *plausible deniability* as an adversary cannot say with certainty what the user's original query input was. However, LDPKiT may leak information about the distribution of the queries. We assume that the user's queries share the same distribution as the cloud model's training data, meaning the cloud model is familiar with the distribution of the user's queries. This assumption is reasonable because ML tasks are typically specialized, and the user is more likely to select cloud services that match her specific needs. We also assume the cloud model provider is honest but curious. It honestly answers the user's queries but may record both the queries and their results to infer information about the user. We also restrict access to the cloud model by assuming that the cloud model returns hard labels only, which makes knowledge transfer more difficult [42], [106].

3.2 Preliminaries

We use the notations in Table 3.1 throughout the thesis.

Note that $\mathcal{D}_{\text{priv}}$ can be a predefined or dynamically generated set of *i.i.d.* data points. We denote the number of queries (*i.e.,* size of $\mathcal{D}_{\text{priv}}$) as $|\mathcal{D}_{\text{priv}}|$. Similarly, the size of $\mathcal{D}_{\text{val}}$ is denoted as $|\mathcal{D}_{\text{val}}|$.

As described above, we use SIDP to denote the standard privacy-preserving inference scheme. Our definition of *standard* LDP-provable noise injection mechanism, or SIDP, has two steps: adding

Table 3.1: Notations used throughout the thesis.

Notation	Type	Description
$\mathcal{M}_{\mathrm{R}}$	Model	Remote model hosted by the cloud provider
$\mathcal{M}_{\mathrm{L}}$	Model	Local model hosted by the user
$\mathcal{D}_{\mathrm{priv}}$	Dataset	Sensitive dataset for which the user wishes to query and attain labels from $\mathcal{M}_{\mathrm{R}}$; Each data sample (query) in the dataset is *i.i.d.*
$\mathcal{D}_{\mathrm{val}}$	Dataset	Validation dataset for model evaluation on unseen data, separate from the training process; Each data sample in the dataset is *i.i.d.*
SIDP	Methodology	Standard privacy-preserving inference scheme with LDP
LDPKiT	Methodology	Our privacy- and utility-preserving inference scheme with LDP
LDPKiT-AL	Methodology	LDPKiT with AL query selection strategy

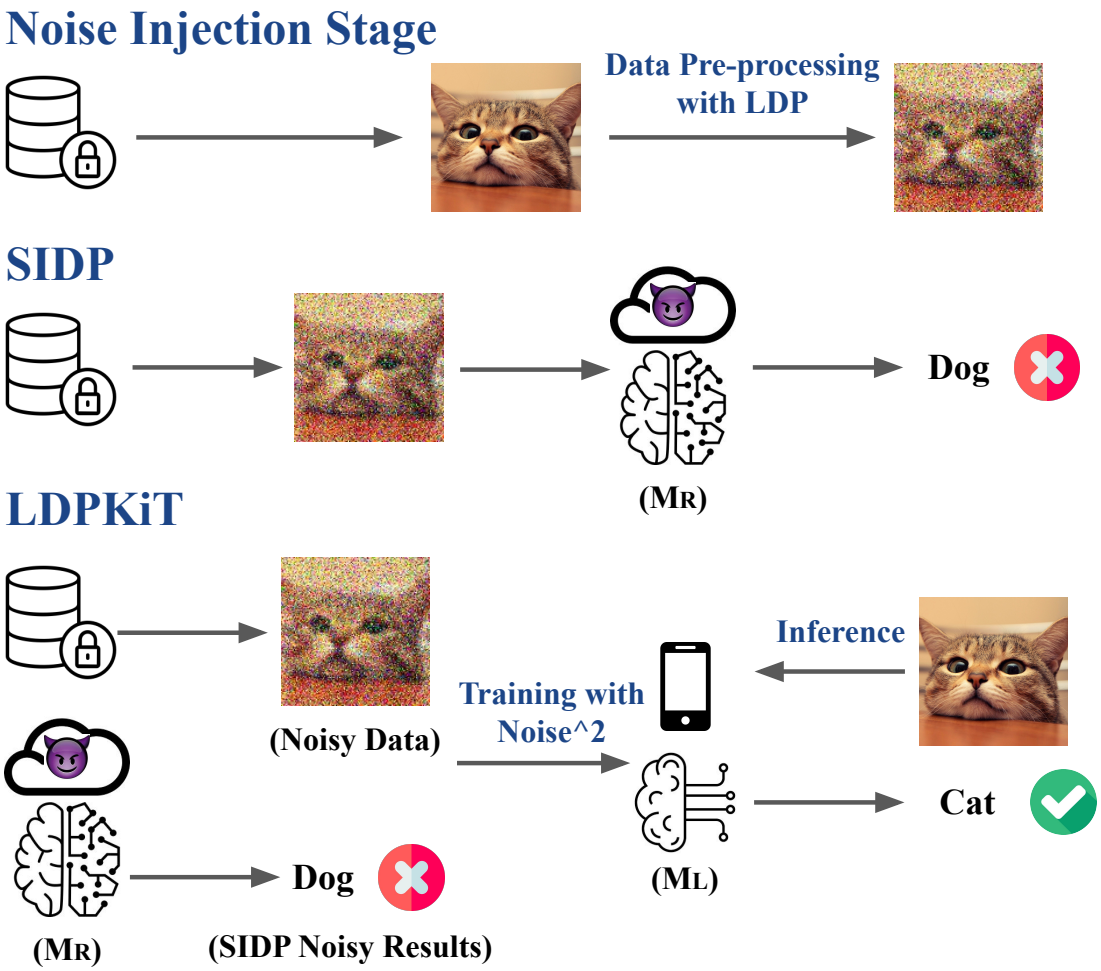


Figure 3.1: LDPKiT system overview.

LDP noise to queries before sending them to $\mathcal{M}_{\mathrm{R}}$, and $\mathcal{M}_{\mathrm{R}}$ returning noisy prediction results (with some errors) on the noisy queries.

Figure 3.1 presents an overview of LDPKiT, which has three stages: noise injection, remote inference/SIDP, and local training. Prior to querying $\mathcal{M}_{\mathrm{R}}$ for inference, LDPKiT adds ϵ-LDP [85]

or ϵ-Utility-optimized Metric LDP (UMLDP) noise to the sensitive data points in $\mathcal{D}_{\text{priv}}$, depending on the modality. Once $\mathcal{M}_{\text{R}}$ returns inference results, LDPKiT enters the local training stage. We elaborate on our design in the following sections.

3.3 Noise Injection

LDPKiT adds Laplacian noise to the sensitive queries in $\mathcal{D}_{\text{priv}}$ to obtain the ϵ-LDP guarantee [85] for image data and conducts text sanitization with ϵ-UMLDP privacy guarantee [107] for textual data. We define ϵ in the context of LDP; hence, the same amount of noise is added to each data sample in $\mathcal{D}_{\text{priv}}$ based on the ϵ value. Specifically, since we focus on providing plausible deniability for each query and we make the assumption that each query from $\mathcal{D}_{\text{priv}}$ is *i.i.d.*, the formulation of ϵ-LDP holds for each data point in $\mathcal{D}_{\text{priv}}$. In other words, the privacy leakage is not cumulative and is bounded by ϵ per query.

As a recap, ϵ-LDP is defined in Section 2.3.1 with the following equation for all pairs of input values ($v1$, $v2$) and all sets of outputs S:

$$\Pr[\mathcal{A}(v_1) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{A}(v_2) \in \mathcal{S}]$$

Definition 3.3.1. Laplacian Mechanism. The Laplacian mechanism of LDP adds noise drawn from the Laplacian distribution, with the probability density function (PDF) defined as follows for a variable z and a scaling factor λ:

$$L(z, \lambda) = \frac{1}{2\lambda} \exp\left(-\frac{|z|}{\lambda}\right) \tag{3.1}$$

We prove that LDPKiT's noise injection scheme satisfies the definition of ϵ-LDP with the Laplacian mechanism as follows.

Theorem 3.3.2. *LDPKiT's noise injection algorithm satisfies ϵ-LDP where $\epsilon = \frac{\Delta_f}{\lambda}$*

Proof. Let v be the original data and $f(v)$ be the query and computation function performed on the data with function sensitivity, $\Delta_f = \max_{v_1, v_2} \|f(v_2) - f(v_1)\|_1$. We define the randomization algorithm $\mathcal{A}$ with Laplacian mechanism such that for any input value v, $\mathcal{A} = f(v) + \mathcal{Z}$, where $\mathcal{Z}$ is sampled from the Laplacian distribution $L(z, \lambda)$ with scaling factor λ set to $\frac{\Delta_f}{\epsilon}$.

The probability that $\mathcal{A}$ has an output $s \in \mathcal{S}$ given an input v can be expressed as:

$$\begin{aligned}
\Pr[\mathcal{A}(v) = s] &= \Pr[f(v) + \mathcal{Z} = s] \\
&= \Pr[\mathcal{Z} = s - f(v)] \\
&= \frac{1}{2\lambda} \exp\left(-\frac{|s - f(v)|}{\lambda}\right)
\end{aligned}$$

for all $s \in \mathcal{S}$.

To satisfy ϵ-LDP, we need Equation 2.3 to hold for any output $s \in \mathcal{S}$ and any input pairs v_1 and v_2.

By substituting the PDF of Laplacian distribution's new expression into the Equation 2.3, we get

$$\frac{1}{2\lambda}\exp\left(-\frac{|s-f(v_1)|}{\lambda}\right) \leq e^{\epsilon}\frac{1}{2\lambda}\exp\left(-\frac{|s-f(v_2)|}{\lambda}\right)$$

which simplifies to

$$\exp\left(\frac{|s-f(v_2)|-|s-f(v_1)|}{\lambda}\right) \leq e^{\epsilon}$$

With $\lambda = \frac{\Delta_f}{\epsilon}$, the equation becomes:

$$\exp\left(\epsilon \cdot \frac{|s-f(v_1)|-|s-f(v_2)|}{\Delta_f}\right) \leq e^{\epsilon} \tag{3.2}$$

for all pairs of v_1 and v_2.

Since $|s-f(v_1)|-|s-f(v_2)| \leq \Delta_f$ by the definition of function sensitivity, Equation 3.2 always holds.

□

Laplacian noise is suitable for certain types of data, such as images. However, it would severely damage the semantic meaning of textual data because they are context-dependent and discrete. Hence, we use ϵ-UMLDP [107], an LDP notion tailored for NLP tasks with promising model accuracy (utility). ϵ-UMLDP [107] performs text sanitization on a vocabulary V by splitting V into sensitive (V_S) and insensitive (V_N) vocabularies based on the word rareness. The intuition is that insensitive words such as "a/an/the" are more frequently used in the English context. In contrast, sensitive words that contain private information such as birthdate, address, and password are used less frequently. The definition of ϵ-UMLDP is as follows.

Definition 3.3.3. ϵ-UMLDP [107]**.** If the user has a sensitive word $x \in V_S$ and an insensitive word $y \in V_N$, The probability to replace x with y is as follows:

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

where $d(\phi(x),\phi(y)) = 4\left|\frac{1-e^{Cosine\_Sim(\phi(x),\phi(y))-1}}{e^{Cosine\_Sim(\phi(x),\phi(y))-1}}\right|$ and C is a constant normalization factor.

The complete proof of ϵ-UMLDP can be found in the original paper [107]. This normalized probability distribution provides ϵ-UMLDP privacy guarantee. It retains some semantic information for utility preservation by replacing the sensitive token x in V_S with the most semantically similar sanitized (insensitive) token y in V_N before sending it to an untrusted third party. We modify the original implementation of ϵ-UMLDP slightly to make the dataset-dependent ϵ values comparable to those used in the image analysis.

The details and analyses of these changes are discussed as follows. As described in Definition 3.3.3, we made some slight modifications to the ϵ-UMLDP formula from the original text sanitization paper [107]. In the original paper, the token sanitization probability controlled by the privacy budget, ϵ, is as follows:

$$Pr[M(x)=y] = C \cdot \frac{\exp\left(-\frac{1}{2}\epsilon \cdot d\_euc(\phi(x),\phi(y))\right)}{\sum_{y' \in V_N}\exp\left(-\frac{1}{2}\epsilon \cdot d\_euc(\phi(x),\phi(y'))\right)} \tag{3.4}$$

In our implementation, we remove the $\frac{1}{2}$ normalization factor before ϵ and more importantly, instead of *Euclidean Distance*, we use a scaled *Cosine Similarity* metric for token embedding distance

measurement. Specifically,

$$d(\phi(x), \phi(y)) = 4 \left| \frac{1 - \exp(Cosine\_Sim(\phi(x), \phi(y)) - 1)}{\exp(Cosine\_Sim(\phi(x), \phi(y)) - 1)} \right| \tag{3.5}$$

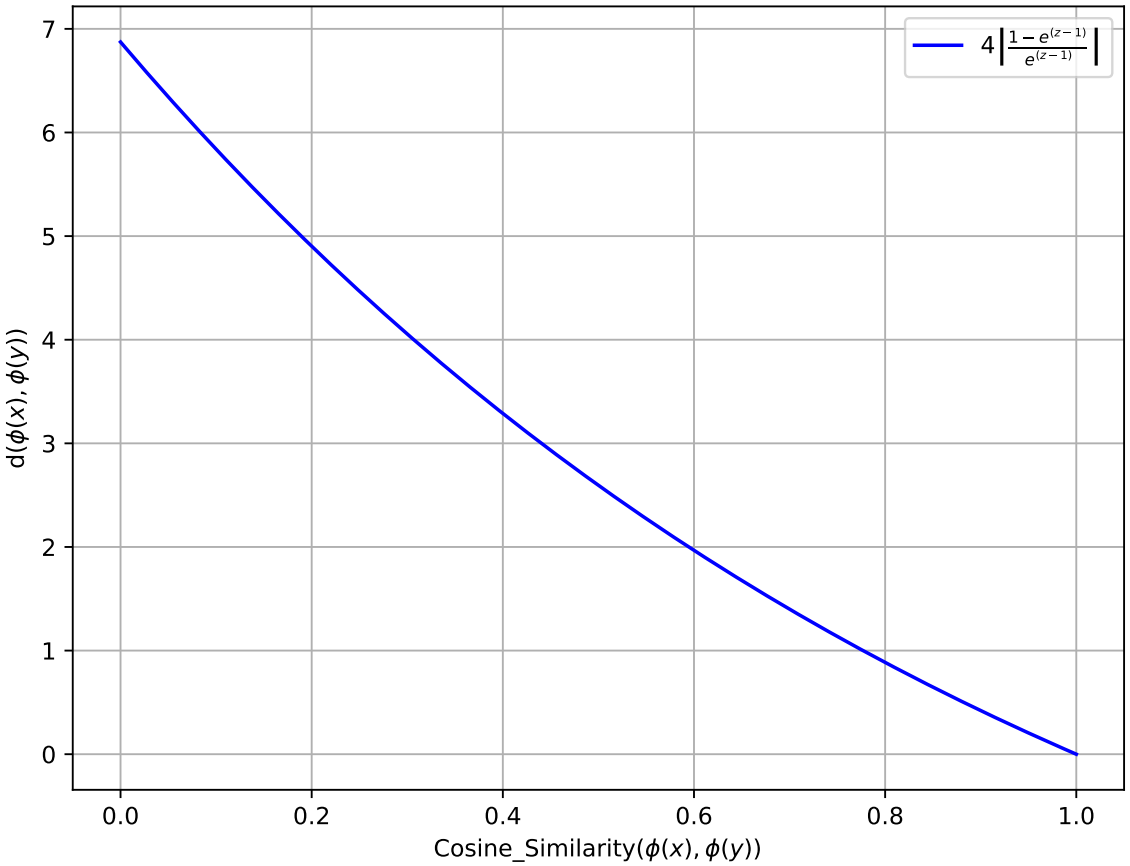


Figure 3.2: Relationship between the scaled cosine similarity and distance metrics of two token embeddings.

We replace Euclidean Distance with Cosine Similarity because it is more commonly used in measuring the differences among embeddings in NLP tasks [108]–[110]. We scale the results of Cosine Similarity between two embeddings with the function $4\left|\frac{1-\exp(Z-1)}{\exp(Z-1)}\right|$, where Z is $Cosine\_Sim(\phi(x), \phi(y))$, so that the more similar two tokens, $\phi(x)$ and $\phi(y)$, are, the smaller $d(\phi(x), \phi(y))$ becomes, and the likelihood of replacement increases more drastically. Similarly, we penalize the dissimilar tokens to preserve utility. Figure 3.2 shows the scaled *Cosine Similarilty* function. As the similarity approaches 1.0, $d(\phi(x), \phi(y))$ approaches zero. We do not need to worry about $d(\phi(x), \phi(y))$ becoming zero because the sets of sensitive and insensitive tokens are mutually exclusive. Same as the original paper, the likelihood of the replacement increases when the two tokens are more similar, *i.e.,* $d(\phi(x), \phi(y))$ is smaller. We make the above changes so that the ϵ values we present in Section 4 are comparable to the values we use for image benchmarks. These changes do not augment the effect of the original ϵ-UMLDP implementation.

More details regarding applying ϵ-UMLDP in LDPKiT are discussed in Yang's thesis report [104].

3.4 Privacy-preserving Inference and Local Model Training with Noiseˆ2

When noise is added to a model input, it will translate into noise in the predicted label of the model output. One way to remove noise is through repeated queries, which is why most DP schemes define a "privacy budget" for repeated queries. Because the noise added by the LDP mechanism is independent of the underlying inputs, repeated queries to the cloud model with different random noise will progressively leak more information [111], [112]. Our intuition is that the repeated responses that the cloud model returns to the user should similarly leak more information about the "true label" that the cloud model would have predicted in the absence of LDP noise. We test this intuition by conducting a simple experiment on 1k random data samples in CIFAR-10 [113] with ResNet-152 ($\mathcal{M}_{\mathrm{R}}$) and ResNet-18 ($\mathcal{M}_{\mathrm{L}}$) [114]. We apply LDP noise on the 1k *i.i.d.* samples, repeatedly query each sample 1k times, and train for 200 epochs. The amount of LDP-provable Laplacian noise we add to each sample satisfies ϵ-LDP where $\epsilon = 5$. To mitigate the random noise among different runs and different datasets, we evaluate it on three different sets of 1k random data points, and we run each experiment three times (9 runs in total). Since $\mathcal{M}_{\mathrm{R}}$ returns different responses each time on the same samples with a different noise, these responses can "collaborate", and the assembled knowledge can be transferred to $\mathcal{M}_{\mathrm{L}}$ throughout training. The experimental results show that $\mathcal{M}_{\mathrm{L}}$ gains enough knowledge about the original data samples to improve the prediction accuracy on them by 14.14%, from 63.89% to 78.03%, with statistical significance indicator $p = 0.0018$ (*i.e.,* $p < 0.05$ indicates that the result is statistically significant). It demonstrates that $\mathcal{M}_{\mathrm{L}}$ can learn to remove noise given noisy labels. However, repeatedly querying the same samples with noise will also leak private information about the samples to $\mathcal{M}_{\mathrm{R}}$ over time, which is undesirable. We, thus, investigate whether $\mathcal{M}_{\mathrm{L}}$ can learn about original samples when presented with noisy labels from querying *different* noisy samples. As a result, each *i.i.d.* element of $\mathcal{D}_{\mathrm{priv}}$ is queried with a single application of noise, thus preserving LDP privacy guarantees for each query.

To realize our idea, LDPKiT first sends these privacy-preserved queries to $\mathcal{M}_{\mathrm{R}}$ and stores $\mathcal{M}_{\mathrm{R}}$'s predictions for further training on $\mathcal{M}_{\mathrm{L}}$. LDPKiT trains $\mathcal{M}_{\mathrm{L}}$ with noised data from a sensitive dataset, $\mathcal{D}_{\mathrm{priv}}$, and $\mathcal{M}_{\mathrm{R}}$'s noisy predictions on those noisy data (hence noiseˆ2). We then use $\mathcal{M}_{\mathrm{L}}$ to infer the correct labels on the original (noise-free) samples in $\mathcal{D}_{\mathrm{priv}}$. LDPKiT can also be applied to an online learning setting where the user can iterate the entire process and periodically train $\mathcal{M}_{\mathrm{L}}$ using $\mathcal{M}_{\mathrm{R}}$'s predictions on new inference queries.

Advanced ML training techniques such as AL [96], and core-set strategies [100] can be used for query selection if $\mathcal{D}_{\mathrm{priv}}$ is large. Related evaluations are presented in Section 4.5.

Chapter 4

Evaluation

In this section, we answer our three research questions in Section 1.1 with empirical analysis.

RQ1. Does LDPKiT recover utility impacted by LDP noise?

RQ2. How does the number of queries impact LDPKiT?

RQ3. How does LDPKiT differ from an adversarial model extraction attack?

4.1 Experimental Setup

We evaluate LDPKiT on two modalities: image and text. The ML models we use in image classification benchmarks are ResNet-152 ($\mathcal{M}_{\text{R}}$) [114], ResNet-18 ($\mathcal{M}_{\text{L}}$) [114] and MobileNetV2 ($\mathcal{M}_{\text{L}}$) [115] . We evaluate LDPKiT on three diverse datasets, namely CIFAR-10 [113], Fashion-MNIST [116], and SVHN [117]. For the NLP benchmark, we use a customized transformer-based model [118], where $\mathcal{M}_{\text{R}}$ has two encoder blocks and $\mathcal{M}_{\text{L}}$ has one, denoted as *Transformer_EN2* and *Transformer_EN1* (See more information in [104]). We run the experiments on CARER's emotion dataset [119].

As discussed in our setting, we have a sensitive dataset ($\mathcal{D}_{\text{priv}}$) of size $|\mathcal{D}_{\text{priv}}|$. Each data sample (*i.e.,* query) from $\mathcal{D}_{\text{priv}}$ is *i.i.d.*, and the differentially private noise applied to each data sample does not depend on others; hence, each data sample has its individual privacy budget, ϵ, and privacy leakage does not accumulate. Two parties are involved: $\mathcal{M}_{\text{R}}$ deployed on a remote cloud and $\mathcal{M}_{\text{L}}$ deployed on the user's trusted local device. $\mathcal{M}_{\text{L}}$ is a model randomly initialized with random weights. We assume that $\mathcal{M}_{\text{R}}$ only returns the hard labels. The user's goal is to obtain accurate predictions on the sensitive data points in $\mathcal{D}_{\text{priv}}$ from the cloud-hosted $\mathcal{M}_{\text{R}}$ while minimizing privacy leakage.

We construct different $\mathcal{M}_{\text{R}}$ for different tasks and datasets. For image modality, we train ResNet-152 on 35k, 35k, and 48,257 data points for CIFAR-10, Fashion-MNIST, and SVHN, respectively. For text modality, we train Transformer_EN2 on 210k data points for CARER. Then, we create $\mathcal{D}_{\text{priv}}$ and a validation dataset, denoted as $\mathcal{D}_{\text{val}}$, using the left-out portion of the datasets that are unseen by $\mathcal{M}_{\text{R}}$. $\mathcal{D}_{\text{val}}$ is isolated from the training process and evaluates $\mathcal{M}_{\text{L}}$'s generalizability on unseen data. For CIFAR-10, $\mathcal{D}_{\text{priv}}$ has 15k data points, and the $\mathcal{D}_{\text{val}}$ has 10k data points. For Fashion-MNIST and SVHN, $|\mathcal{D}_{\text{priv}}|$ is 25k for both datasets, and the sizes of $\mathcal{D}_{\text{val}}$ are 10k and 26,032, respectively. Since CARER has over 410k data points, about 5× larger than image benchmarks, $\mathcal{D}_{\text{priv}}$, in this case, is larger, containing 70k data samples, and the $\mathcal{D}_{\text{val}}$ has 10k samples. As a reference, when no privacy protection exists and thus no noise is added, $\mathcal{M}_{\text{R}}$'s accuracies on $\mathcal{D}_{\text{priv}}$ of CIFAR-10,

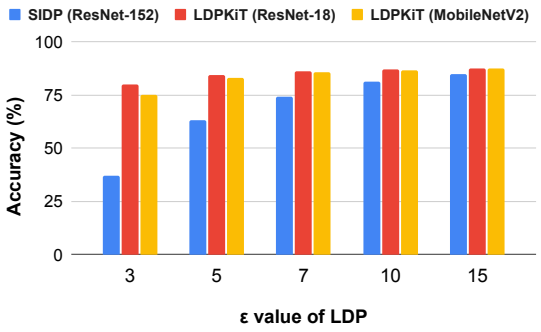
Fashion-MNIST, SVHN, and CARER are 87.85%, 93.49%, 95.20%, and 91.00%. The accuracies on $\mathcal{D}_{\text{val}}$ are 87.10%, 93.22%, 96.30%, and 90.00%, respectively. For all the experiments, we gradually expand $\mathcal{D}_{\text{priv}}$ and train $\mathcal{M}_{\text{L}}$ on noisy data and labels iteratively to study the effect of $|\mathcal{D}_{\text{priv}}|$. In each iteration, LDPKiT randomly selects a small batch of samples from $\mathcal{D}_{\text{priv}}$ (1.5k for CIFAR-10, 2.5k for Fashion-MNIST and SVHN, and 7k for CARER NLP benchmark), adds LDP noise, and then sends them to $\mathcal{M}_{\text{R}}$ to attain noisy labels. The ϵ values we present in our evaluation are 15, 10, 7, 5, and 3 across all the benchmarks, which are generally accepted by the industry standard [120], [121]. As a recap, ϵ is defined in the context of LDP, so we add the same amount of noise to each data sample in $\mathcal{D}_{\text{priv}}$ before querying. We also assume that each query in $\mathcal{D}_{\text{priv}}$ is independent, so the information inferred from one data sample does not leak information from other data samples in $\mathcal{D}_{\text{priv}}$ (*i.e.,* privacy leakage on $\mathcal{D}_{\text{priv}}$ is not cumulative). We collect and report $\mathcal{M}_{\text{L}}$'s accuracies on the original samples in $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$, where $\mathcal{M}_{\text{L}}$ is trained on noisy data from $\mathcal{D}_{\text{priv}}$ with different noise levels (noiseˆ2).

We run our experiments on two machines. One has two GPUs with models NVIDIA GeForce RTX 3090 and 4090 with 24GB of dedicated memory and an Intel 12th Gen i7-12700 CPU with 12 cores and 64GB of RAM. The other has two NVIDIA GeForce RTX 4090 GPUs and an AMD Ryzen Threadripper PRO 5955WX CPU with 16 cores and 64GB of RAM. The underlying OS are 64-bit Ubuntu 22.04.3 LTS and Ubuntu 24.04 LTS, respectively. We use Python 3.9.7 and PyTorch v2.1.2 with CUDA 12.1. All experiments are repeated over three random seeds to determine the statistical significance of our findings. We conduct the dependent two-sample t-test on our results and collect the p values. We find that most of the improvements are statistically significant (*i.e.,* $p < 0.05$) with few outliers. We mark the accuracies that have $p > 0.05$ with an asterisk (*) in Tables 4.1 and 4.2.

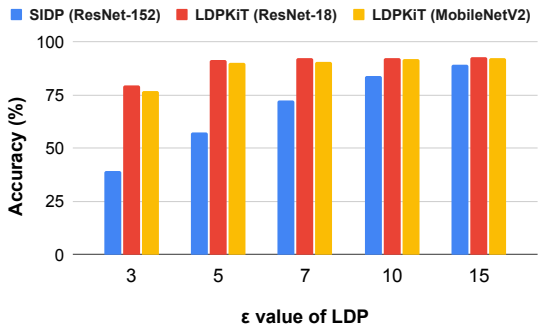
For image modality, we train $\mathcal{M}_{\text{R}}$ with a learning rate of 0.1 for 200 epochs on each dataset. $\mathcal{M}_{\text{R}}$ is trained on 35k data points for CIFAR-10 and Fashion-MNIST, 48,257 data points for SVHN and 210k data points for CARER. We use CE Loss and SGD optimizer [122] to train $\mathcal{M}_{\text{R}}$ for each image dataset. $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ are split from the remaining data points unseen by $\mathcal{M}_{\text{R}}$, where $\mathcal{D}_{\text{priv}}$ is used to train and evaluate $\mathcal{M}_{\text{L}}$, and $\mathcal{D}_{\text{val}}$ is used for pure model generalizability evaluation. Specifically, CIFAR-10 has 15k data points in $\mathcal{D}_{\text{priv}}$, Fashion-MNIST and SVHN have 25k, and CARER has 70k. As for $\mathcal{D}_{\text{val}}$, its size is 10k for both CIFAR-10 and Fashion-MNIST, 26,032 for SVHN, and 10k for CARER. We train $\mathcal{M}_{\text{L}}$ in an iterative learning process instead of one-shot learning to study the effect of $|\mathcal{D}_{\text{priv}}|$. In each iteration, $\mathcal{M}_{\text{L}}$ randomly selects 1.5k, 2.5k, and 2.5k data samples from $\mathcal{D}_{\text{priv}}$ for CIFAR-10, Fashion-MNIST, and SVHN, respectively. Thus, the total number of iterations is 10 for all benchmarks to finish querying the entire $\mathcal{D}_{\text{priv}}$. The epoch number in each iteration is set to 100 for CIFAR-10 and Fashion-MNIST and 50 for SVHN. $\mathcal{M}_{\text{L}}$'s learning rate is set to 0.1 for all image benchmarks. We use CE loss as our loss function. We use Adam optimizer [123] for CIFAR-10 and Fashion-MNIST and SGD optimizer for SVHN throughout the experiments presented in Sections 4.2 to 4.5.1. We provide auxiliary experimental results on image benchmarks in Section 4.5 where LDPKiT is applied with AL or with $\mathcal{M}_{\text{R}}$'s softmax values returned. We use Entropy Sampling as AL's query strategy and incorporate KLDiv loss (See definition 2.4.2) in the loss calculation when $\mathcal{M}_{\text{R}}$ returns softmax information. For KLDiv loss, we set the hyperparameters α to 1.0 and *temperature*, T, to 20. We keep all the other parameters and hyperparameters the same as above.

For text modality, we train $\mathcal{M}_{\text{R}}$ and $\mathcal{M}_{\text{L}}$ with a learning rate of 0.0001 for 20 epochs on CARER. $\mathcal{M}_{\text{L}}$ randomly picks 7k data points from $\mathcal{D}_{\text{priv}}$ in each iteration and trains for 20 epochs per iteration,

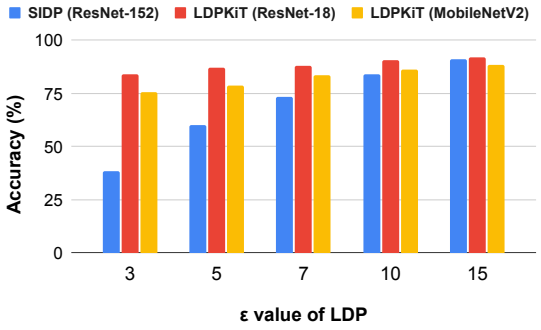
10 iterations in total, to query the entire 70k dataset split of $\mathcal{D}_{\text{priv}}$. If $\mathcal{M}_{\text{L}}$'s training dataset is the entire noised $\mathcal{D}_{\text{priv}}$, eventually, $\mathcal{M}_{\text{L}}$ is trained on 15k data points for CIFAR-10, 25k data points for Fashion-MINST and SVHN, and 70k data points for CARER. The final accuracies in Figures 4.6 to 4.12, Tables 4.1 and 4.2, as well as all the Zest distances, are reported assuming $\mathcal{M}_{\text{L}}$ is trained on the entire split of $\mathcal{D}_{\text{priv}}$.



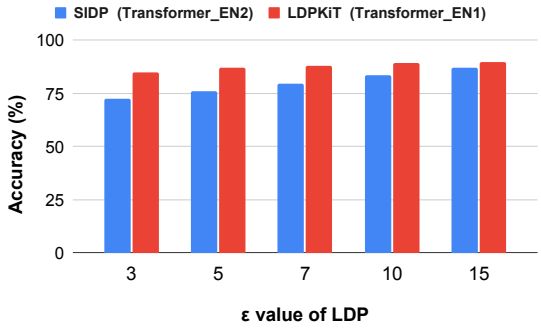
(a) CIFAR-10



(b) Fashion-MNIST



(c) SVHN



(d) CARER

Figure 4.1: Comparison of accuracies on $\mathcal{D}_{\text{priv}}$ between SIDP and LDPKiT with different ϵ values.

4.2 RQ1: Utility Recovery

We record the final accuracies on the entire $\mathcal{D}_{\text{priv}}$ at the last epoch of training (*i.e.,* 15k for CIFAR-10, 25k for Fashion-MNIST and SVHN, and 70k for CARER) and show the difference of final accuracies on $\mathcal{D}_{\text{priv}}$ between SIDP and LDPKiT in Figure 4.1. Our results show that LDPKiT can almost always achieve higher inference accuracy than SIDP, except in a few cases when the least noise is added (*i.e.,* $\epsilon = 15$). However, there is barely privacy protection in the case of $\epsilon = 15$ (See Figures 4.2 to 4.5 for samples with different noise levels). As a recap, the ϵ value represents the amount of Laplacian noise added to each data sample in $\mathcal{D}_{\text{priv}}$, and its value corresponds to the value of ϵ in the LDP scheme. As more LDP noise is added to $\mathcal{D}_{\text{priv}}$ to preserve privacy, the gap in utility that LDPKiT provides over SIDP also increases. Therefore, LDPKiT offers greater benefits in regimes with stronger privacy protection and correspondingly more noise.

In contrast, when privacy protection is weak, and the ϵ value is high (*e.g.,* $\epsilon = 15$), LDPKiT's improvement becomes minimal. We tabulate the numerical accuracies on $\mathcal{D}_{\text{priv}}$ obtained in our ex-

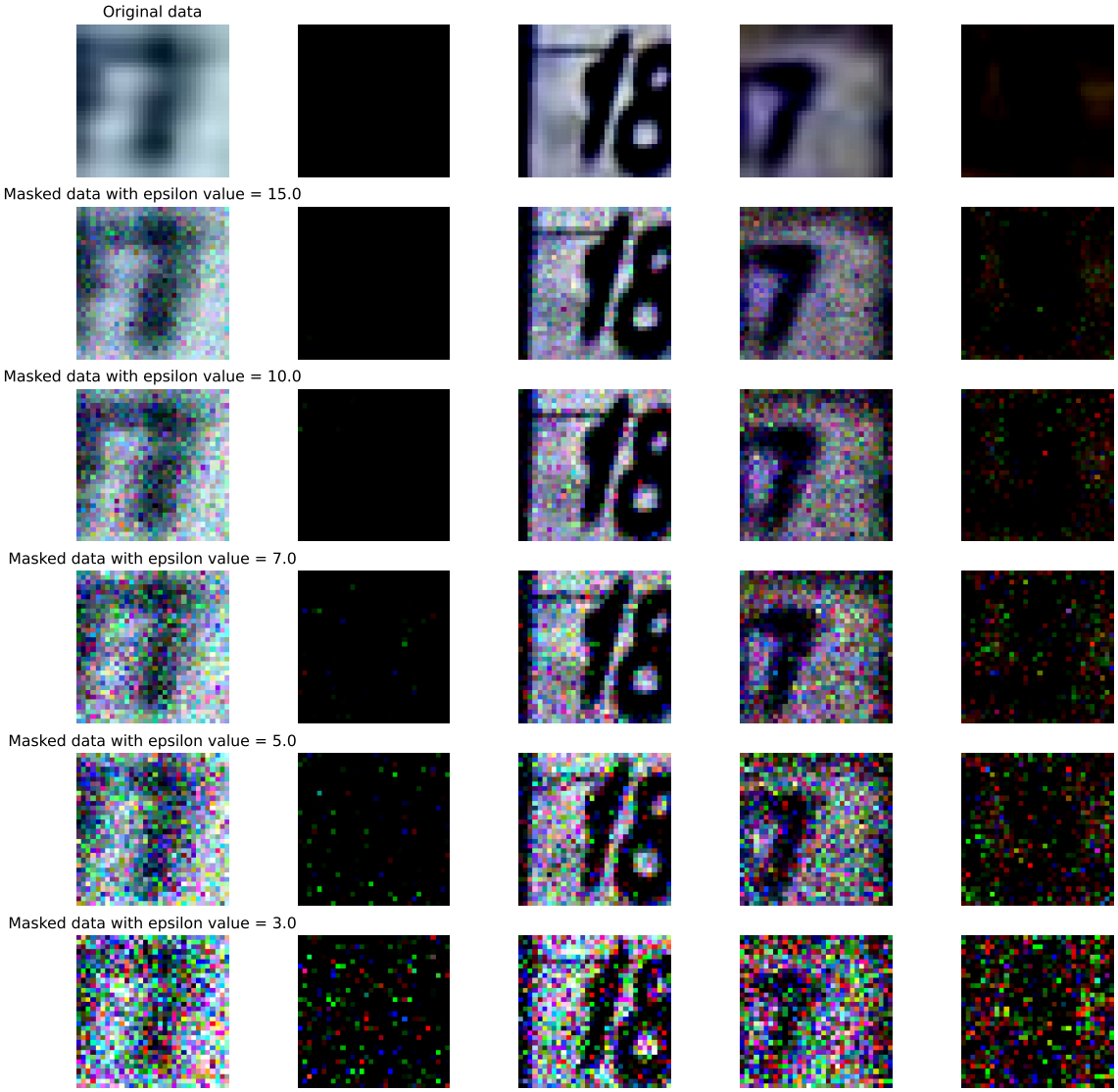


Figure 4.2: SVHN data samples with different noise levels.

Original: [‘[CLS]’, ‘i’, ‘feel’, **‘sir’**, **‘alex’**, **‘ferguson’**, ‘is’, ‘a’, ‘keen’, **‘admire’**, ‘##r’, ‘and’, ‘would’, ‘love’, ‘to’, ‘have’, ‘him’, ‘back’, ‘but’, ‘only’, ‘time’, ‘will’, ‘tell’, ‘[SEP]‘]

Substituted (ε=15): [‘[CLS]’, ‘i’, ‘feel’, **‘##cia’**, **‘##py’**, **‘side’**, ‘is’, ‘a’, ‘keen’, **‘close’**, ‘##r’, ‘and’, ‘would’, ‘love’, ‘to’, ‘have’, ‘him’, ‘back’, ‘but’, ‘only’, ‘time’, ‘will’, ‘tell’, ‘[SEP]‘]

Figure 4.3: CARER text data sample with ϵ-UMLDP noise ($\epsilon = 15$).

periments in Table 4.1. Our experiments show that LDPKiT provides better privacy with essentially no loss of accuracy compared to SIDP. For instance, on CIFAR-10 with ResNet-18, SIDP provides an average accuracy of 84.98% at $\epsilon = 15$, while LDPKiT is able to provide a roughly equivalent accuracy of 84.39% for a much stronger level of privacy at $\epsilon = 5$. In contrast, at the same level of

Original: [‘[CLS]’, ‘i’, ‘feel’, ‘as’, ‘if’, ‘that’, ‘by’, ‘itself’, ‘would’, ‘ve’, ‘given’, ‘more’, **‘lyrical’**, ‘##ly’, ‘talented’, ‘r’, ‘amp’, ‘b’, ‘artists’, ‘like’, **‘jill’**, **‘scott’**, ‘a’, **‘wider’**, **‘platform’**, ‘without’, **‘selling’**, ‘out’, ‘on’, ‘what’, ‘they’, ‘believe’, ‘[SEP]‘]

Substituted (ε=5): [‘[CLS]’, ‘i’, ‘feel’, ‘as’, ‘if’, ‘that’, ‘by’, ‘itself’, ‘would’, ‘ve’, ‘given’, ‘more’, **‘won’**, ‘##ly’, ‘talented’, ‘r’, ‘amp’, ‘b’, ‘artists’, ‘like’, **‘travel’**, **‘rid’**, ‘a’, **‘increasingly’**, **‘ask’**, ‘without’, **‘current’**, ‘out’, ‘on’, ‘what’, ‘they’, ‘believe’, ‘[SEP]‘]

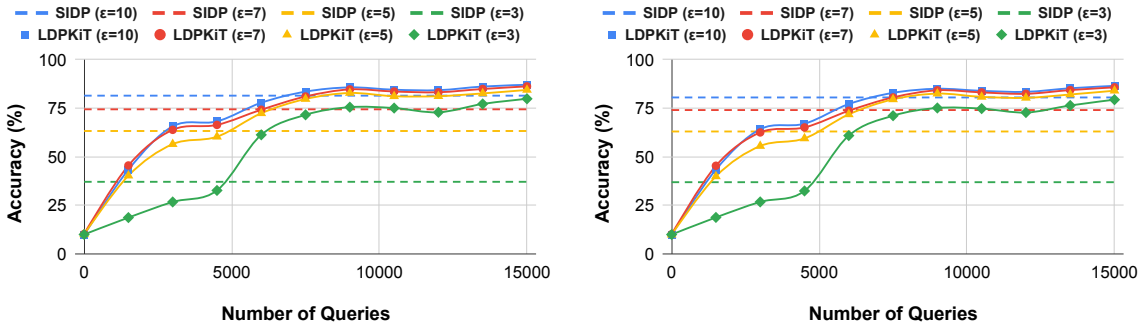
Figure 4.4: CARER text data sample with ϵ-UMLDP noise ($\epsilon = 5$).

Original: [‘[CLS]’, ‘i’, ‘feel’, ‘cause’, ‘all’, ‘of’, ‘the’, ‘most’, ‘amazing’, **‘poets’**, ‘that’, ‘iv’, ‘##e’, ‘ever’, ‘and’, ‘when’, ‘i’, ‘use’, ‘the’, ‘word’, **‘poet’**, ‘i’, ‘mean’, **‘ben’**, **‘webster’**, ‘or’, **‘billie’**, **‘holiday’**, **‘or’**, **‘maya’**, **‘pe’**, **‘##lis’**, **‘##ets’**, **‘##kaya’**, ‘or’, ‘the’, **‘incredible’**, **‘carmen’**, **‘ama’**, **‘##ya’**, ‘[SEP]‘]

Substituted (ε=3): [‘[CLS]’, ‘i’, ‘feel’, ‘cause’, ‘all’, ‘of’, ‘the’, ‘most’, ‘amazing’, **‘same’**, ‘that’, ‘iv’, ‘##e’, ‘ever’, ‘and’, ‘when’, ‘i’, ‘use’, ‘the’, ‘word’, **‘some’**, ‘i’, ‘mean’, **‘saying’**, **‘iv’**, ‘or’, **‘does’**, **‘lou’**, ‘or’, **‘##ant’**, **‘warm’**, **‘having’**, **‘more’**, **‘very’**, ‘or’, ‘the’, **‘who’**, **‘doomed’**, **‘valuable’**, **‘decided’**, ‘[SEP]‘]

Figure 4.5: CARER text data sample with ϵ-UMLDP noise ($\epsilon = 3$).

privacy of $\epsilon = 5$, SIDP only achieves an average accuracy of 63.21% by comparison. Similarly, for CARER, SIDP provides an accuracy of 87.12% at $\epsilon = 15$, while LDPKiT achieves an accuracy of 84.75% at $\epsilon = 3$. This trend is also present in Fashion-MNIST and SVHN. The experimental results demonstrate that LDPKiT can recover a majority of the utility loss due to the addition of LDP noise.



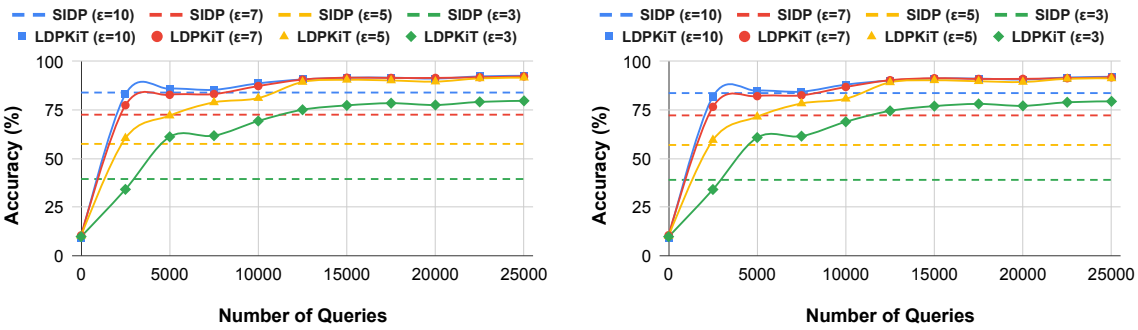
(a) CIFAR-10 ($\mathcal{D}_{\text{priv}}$) (b) CIFAR-10 ($\mathcal{D}_{\text{val}}$)

SIDP lines in the plots give the average accuracy across all queries at a given ϵ.

Figure 4.6: ResNet-18’s accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ of CIFAR-10.

4.3 RQ2: Influence of $|\mathcal{D}_{\text{priv}}|$ on LDPKiT

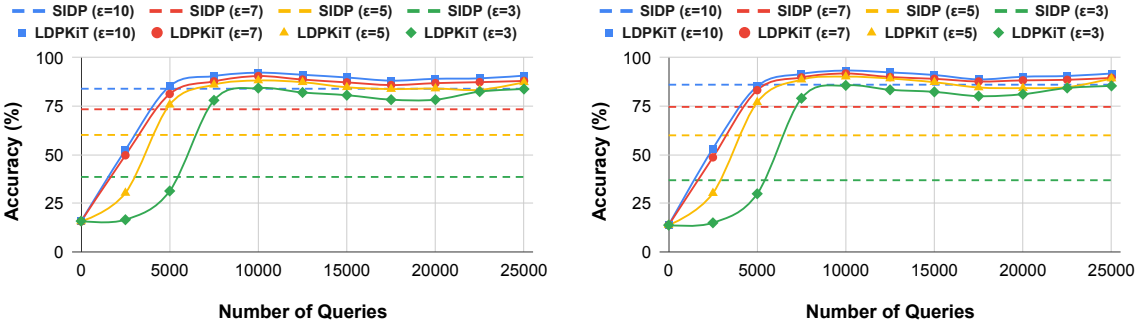
Figures 4.6 to 4.12 illustrate the accuracy of $\mathcal{M}_{\text{L}}$ on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ as a function of $|\mathcal{D}_{\text{priv}}|$ at various values of ϵ, along with the average accuracy of SIDP on $\mathcal{D}_{\text{priv}}$ at the same values of ϵ (as dotted lines) on CIFAR-10, Fashion-MNIST, SVHN and CARER, respectively. The accuracy plots of LDPKiT and SIDP on the expanding $\mathcal{D}_{\text{priv}}$ are presented in Appendix 6.1. We observe that accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ both exhibit similar and increasing accuracies as $|\mathcal{D}_{\text{priv}}|$ increases. Moreover, we



(a) Fashion-MNIST ($\mathcal{D}_{\text{priv}}$) (b) Fashion-MNIST ($\mathcal{D}_{\text{val}}$)

SIDP lines in the plots give the average accuracy across all queries at a given ϵ.

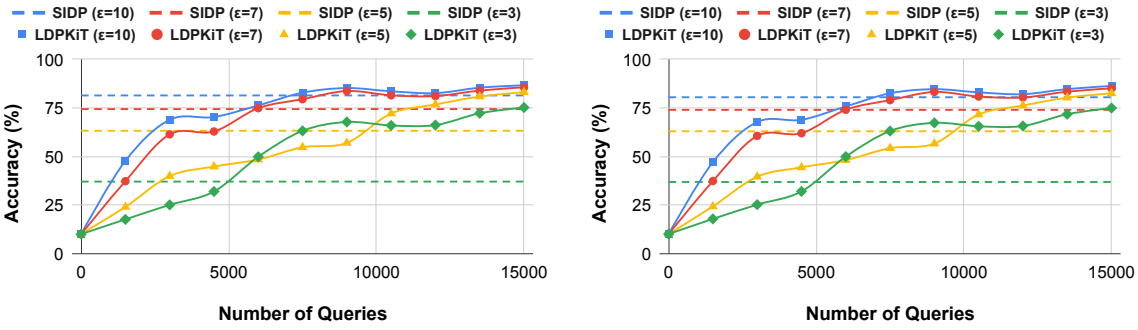
Figure 4.7: ResNet-18's accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ of Fashion-MNIST.



(a) SVHN ($\mathcal{D}_{\text{priv}}$) (b) SVHN ($\mathcal{D}_{\text{val}}$)

SIDP lines in the plots give the average accuracy across all queries at a given ϵ.

Figure 4.8: ResNet-18's accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ of SVHN.

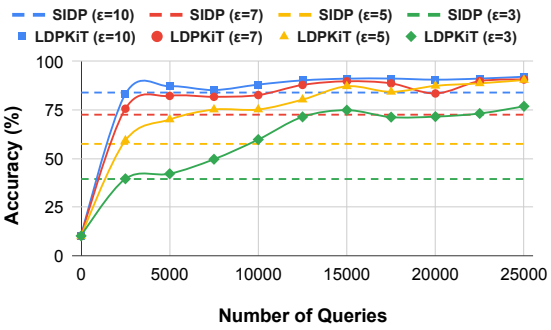


(a) CIFAR-10 ($\mathcal{D}_{\text{priv}}$) (b) CIFAR-10 ($\mathcal{D}_{\text{val}}$)

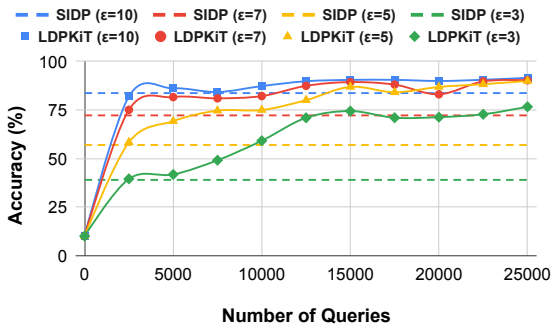
SIDP lines in the plots give the average accuracy across all queries at a given ϵ.

Figure 4.9: MobileNetV2's accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ of CIFAR-10.

observe that for each noise level, there exists a lower bound of $|\mathcal{D}_{\text{priv}}|$, *i.e.,* a *cross-over* point, where the increasing LDPKiT accuracy (solid lines) exceeds the average SIDP accuracy (dotted lines). When $|\mathcal{D}_{\text{priv}}|$ is too small, it is not sufficient to train a $\mathcal{M}_{\text{L}}$ that can outperform SIDP. Hence, $|\mathcal{D}_{\text{priv}}|$ needs to be larger than this cross-over point for LDPKiT to be beneficial. We notice that



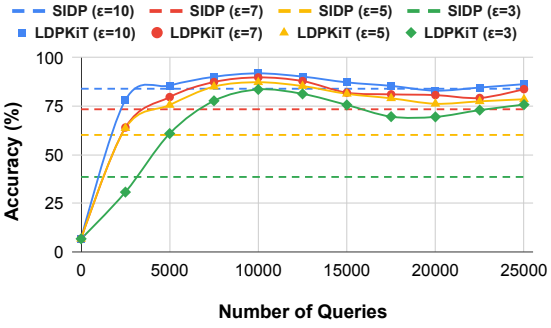
(a) Fashion-MNIST ($\mathcal{D}_{\text{priv}}$)



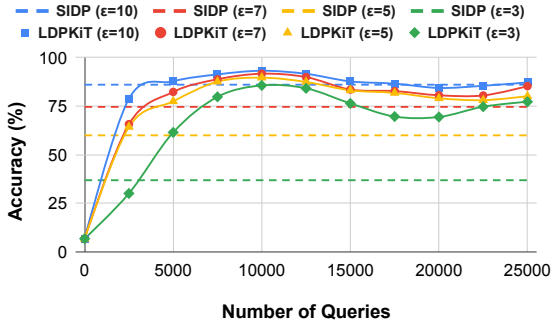
(b) Fashion-MNIST ($\mathcal{D}_{\text{val}}$)

SIDP lines in the plots give the average accuracy across all queries at a given ϵ.

Figure 4.10: MobileNetV2's accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ of Fashion-MNIST.



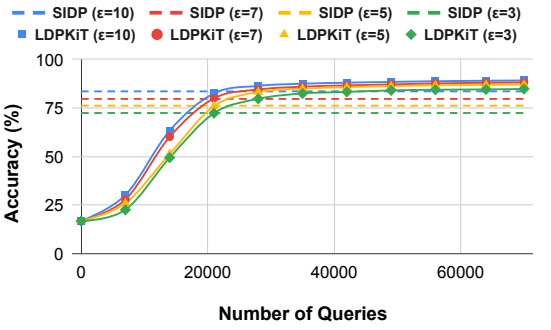
(a) SVHN ($\mathcal{D}_{\text{priv}}$)



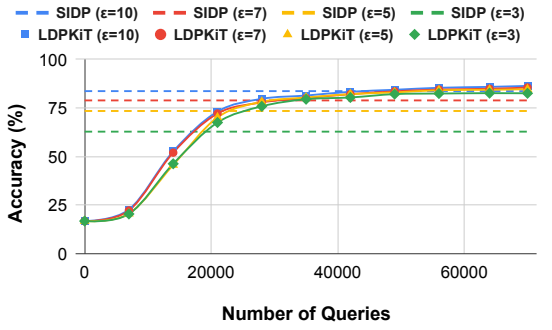
(b) SVHN ($\mathcal{D}_{\text{val}}$)

SIDP lines in the plots give the average accuracy across all queries at a given ϵ.

Figure 4.11: MobileNetV2's accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ of SVHN.



(a) CARER ($\mathcal{D}_{\text{priv}}$)



(b) CARER ($\mathcal{D}_{\text{val}}$)

SIDP lines in the plots give the average accuracy across all queries at a given ϵ.

Figure 4.12: Transformer_EN1's accuracies on $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{val}}$ of CARER emotion dataset.

the $|\mathcal{D}_{\text{priv}}|$ where the cross-over point occurs is dataset-dependent. For CIFAR-10 and CARER, the cross-over points are lower when more noise is added. In other words, fewer queries are needed for LDPKiT to outperform SIDP in a more privacy-protective setting. As shown in the Subplot 4.6a

Table 4.1: Comparison of final accuracies on $\mathcal{D}_{\text{priv}}$ between SIDP and LDPKiT.

Dataset	Model	Strategy	Accuracy on $\mathcal{D}_{\text{priv}}$ (%)				
			LDP (ϵ=15)	LDP (ϵ=10)	LDP (ϵ=7)	LDP (ϵ=5)	LDP (ϵ=3)
CIFAR-10	ResNet-152 ($\mathcal{M}_{\text{R}}$)	SIDP	84.98	81.42	74.38	63.21	37.03
	ResNet-18 ($\mathcal{M}_{\text{L}}$)	LDPKiT	87.77 ($\pm$0.28)*	87.03 ($\pm$0.11)*	86.20 ($\pm$0.16)	84.39 ($\pm$0.13)	79.81 ($\pm$0.52)
	MobileNetV2 ($\mathcal{M}_{\text{L}}$)	LDPKiT	87.74 ($\pm$0.11)	86.68 ($\pm$0.14)*	85.60 ($\pm$0.23)*	83.12 ($\pm$0.76)	75.21 ($\pm$0.58)
Fashion-MNIST	ResNet-152 ($\mathcal{M}_{\text{R}}$)	SIDP	89.28	83.96	72.55	57.50	39.41
	ResNet-18 ($\mathcal{M}_{\text{L}}$)	LDPKiT	92.73 ($\pm$0.34)	92.60 ($\pm$0.15)	92.28 ($\pm$0.30)	91.61 ($\pm$0.21)	79.69 ($\pm$0.21)
	MobileNetV2 ($\mathcal{M}_{\text{L}}$)	LDPKiT	92.63 ($\pm$0.03)	92.11 ($\pm$0.11)	90.67 ($\pm$1.15)	90.42 ($\pm$0.22)	76.88 ($\pm$0.49)
SVHN	ResNet-152 ($\mathcal{M}_{\text{R}}$)	SIDP	90.94	83.96	73.38	60.16	38.51
	ResNet-18 ($\mathcal{M}_{\text{L}}$)	LDPKiT	92.17 ($\pm$0.64)*	90.71 ($\pm$0.85)	87.98 ($\pm$0.78)	87.27 ($\pm$1.07)	83.81 ($\pm$1.16)
	MobileNetV2 ($\mathcal{M}_{\text{L}}$)	LDPKiT	88.40 ($\pm$0.63)	86.33 ($\pm$2.15)*	83.74 ($\pm$0.44)	78.54 ($\pm$4.00)	75.77 ($\pm$3.63)
CARER	Transformer_EN2 ($\mathcal{M}_{\text{R}}$)	SIDP	87.12	83.53	79.63	76.13	72.38
	Transformer_EN1 ($\mathcal{M}_{\text{L}}$)	LDPKiT	89.86 ($\pm$1E-3)	89.14 ($\pm$1E-3)	88.11 ($\pm$2E-3)	87.05 ($\pm$2E-3)	84.75 ($\pm$4E-4)

*Accuracy** indicates that the accuracy has a $p > 0.05$. The values recorded in parentheses are the standard deviations of the accuracies.

Table 4.2: Comparison of final accuracies on $\mathcal{D}_{\text{val}}$ between SIDP and LDPKiT.

Dataset	Model	Strategy	Accuracy on $\mathcal{D}_{\text{val}}$ (%)				
			LDP (ϵ=15)	LDP (ϵ=10)	LDP (ϵ=7)	LDP (ϵ=5)	LDP (ϵ=3)
CIFAR-10	ResNet-152 ($\mathcal{M}_{\text{R}}$)	SIDP	84.28	80.50	74.00	62.96	36.84
	ResNet-18 ($\mathcal{M}_{\text{L}}$)	LDPKiT	87.01 ($\pm$0.25)	86.47 ($\pm$0.19)	85.70 ($\pm$0.12)*	83.98 ($\pm$0.14)	79.35 ($\pm$0.61)
	MobileNetV2 ($\mathcal{M}_{\text{L}}$)	LDPKiT	86.99 ($\pm$0.27)	86.31 ($\pm$0.16)	85.07 ($\pm$0.31)*	82.50 ($\pm$0.70)	74.93 ($\pm$0.50)
Fashion-MNIST	ResNet-152 ($\mathcal{M}_{\text{R}}$)	SIDP	89.12	83.68	72.19	56.95	38.97
	ResNet-18 ($\mathcal{M}_{\text{L}}$)	LDPKiT	92.29 ($\pm$0.42)	92.10 ($\pm$0.31)	91.82 ($\pm$0.33)	91.26 ($\pm$0.28)	79.42 ($\pm$0.27)
	MobileNetV2 ($\mathcal{M}_{\text{L}}$)	LDPKiT	91.92 ($\pm$0.06)	91.55 ($\pm$0.21)	90.30 ($\pm$1.13)	89.93 ($\pm$0.21)	76.63 ($\pm$0.54)
SVHN	ResNet-152 ($\mathcal{M}_{\text{R}}$)	SIDP	92.63	86.04	74.60	59.95	36.82
	ResNet-18 ($\mathcal{M}_{\text{L}}$)	LDPKiT	93.12 ($\pm$0.83)*	91.81 ($\pm$0.34)	89.54 ($\pm$0.87)	89.22 ($\pm$1.59)	85.42 ($\pm$1.46)
	MobileNetV2 ($\mathcal{M}_{\text{L}}$)	LDPKiT	89.06 ($\pm$0.47)	87.21 ($\pm$2.33)*	85.29 ($\pm$0.48)	80.13 ($\pm$4.09)	77.26 ($\pm$3.31)
CARER	Transformer_EN2 ($\mathcal{M}_{\text{R}}$)	SIDP	87.21	83.65	78.86	73.41	62.77
	Transformer_EN1 ($\mathcal{M}_{\text{L}}$)	LDPKiT	87.09 ($\pm$2E-3)*	86.34 ($\pm$1E-3)	85.49 ($\pm$6E-3)	84.64 ($\pm$5E-3)	82.53 ($\pm$4E-3)

*Accuracy** indicates that the accuracy has a $p > 0.05$. The values recorded in parentheses are the standard deviations of the accuracies.

of Figure 4.6, the cross-over point for ResNet-18 on CIFAR-10 ($\mathcal{D}_{\text{priv}}$) is around 5k queries when $\epsilon = 3$. The cross-over point increases to 8k when $\epsilon = 10$. In contrast, we observe a different trend for Fashion-MNIST and SVHN. In these cases, the cross-over points are similar across all ϵ values. For instance, the cross-over points for Fashion-MNIST are around 2.5k for all ϵ, and for SVHN, the cross-over points are around 5k for ResNet-18 and 2.5k for MobileNetV2. We even notice a few instances where a cross-over point occurs at a smaller $|\mathcal{D}_{\text{priv}}|$ when less noise is added. For example, the cross-over point for SVHN occurs around 4k for $\epsilon = 5$ and over 5k for $\epsilon = 3$. Overall, fewer queries are generally needed for LDPKiT accuracy to exceed the average SIDP accuracy compared to CIFAR-10 and CARER. We suspect these differences can be attributed to Fashion-MNIST and SVHN being easier to train ML models on and, thus, having accuracies on these datasets that increase faster relative to the size of $\mathcal{D}_{\text{priv}}$. However, further investigation is needed to fully explain this dataset-dependent difference.

One may argue that LDPKiT's effectiveness is highly dependent on $|\mathcal{D}_{\text{priv}}|$, and $|\mathcal{D}_{\text{priv}}|$ should be as high as possible to make LDPKiT become beneficial. This is not true because LDP only bounds privacy leakage—it does not make it non-existent—so every additional query the user makes still leaks some amount of information. On the other hand, the gains in $\mathcal{M}_{\text{L}}$ accuracy see diminishing returns at large $|\mathcal{D}_{\text{priv}}|$. Therefore, the user may wish to set an upper bound of $|\mathcal{D}_{\text{priv}}|$, *i.e.,* a *stopping*

point, based on their desire for privacy and the decreasing benefits of querying $\mathcal{M}_{\mathrm{R}}$ for LDPKiT. We observe that more queries are required for cases with higher and more privacy-protective noise levels to reach this upper bound across all the benchmarks. Training on Fashion-MNIST and SVHN reaches such an upper bound at a relatively earlier stage compared to other datasets, *i.e.,* the accuracy remains high (saturates) after about 5k queries are made, except for the cases $\epsilon = 3$ in SVHN and $\epsilon = 3$ or 5 in Fashion-MNIST. Training MobileNetV2 on CIFAR-10, however, seems not to reach the stopping point even at the end when $\epsilon < 7$ (See Figure 4.9). Again, we believe that it is because Fashion-MNIST and SVHN are relatively easier datasets for which to train models than CIFAR-10. One way for the user to determine the stopping point to prevent further privacy leakage is to monitor the accuracy on a small labelled $\mathcal{D}_{\mathrm{val}}$. For instance, when $\mathcal{M}_{\mathrm{L}}$ has a decent accuracy on $\mathcal{D}_{\mathrm{val}}$ and increasing $|\mathcal{D}_{\mathrm{priv}}|$ brings minimal accuracy improvement on $\mathcal{D}_{\mathrm{val}}$, the user can stop querying $\mathcal{M}_{\mathrm{R}}$ and rely on $\mathcal{M}_{\mathrm{L}}$ for predictions so long as future samples fall in the same distribution as $\mathcal{D}_{\mathrm{priv}}$.

We tabulate the detailed accuracies on $\mathcal{D}_{\mathrm{priv}}$ and $\mathcal{D}_{\mathrm{val}}$ in Tables 4.1 and 4.2. The experimental results show that LDPKiT has greater benefits when more LDP noise is added (*i.e.,* stronger privacy protection). As shown in Figures 4.6 to 4.12, $\mathcal{M}_{\mathrm{L}}$'s accuracies on $\mathcal{D}_{\mathrm{priv}}$ and $\mathcal{D}_{\mathrm{val}}$ illustrate similar trends. Hence, when $|\mathcal{D}_{\mathrm{priv}}|$ is large enough, LDPKiT generates a trustworthy local model that can accurately predict unseen sensitive data points. The user can utilize the accuracy on a $\mathcal{D}_{\mathrm{val}}$ as a reference to determine the stopping point of querying $\mathcal{M}_{\mathrm{R}}$ to prevent further privacy leakage.

In LDPKiT, instead of training $\mathcal{M}_{\mathrm{L}}$ with original samples, we train it on noised samples, as described in Section 3.4. Intuitively, the additive noise will likely change some of the features in the original samples, so the noised samples will more strongly correspond to the noised labels predicted by $\mathcal{M}_{\mathrm{R}}$ for knowledge transfer. We compare local training with noisy or original data samples and observe that training $\mathcal{M}_{\mathrm{L}}$ with noisy samples yields greater recovery of accuracy, especially when there is more noise. Furthermore, since $\mathcal{M}_{\mathrm{L}}$ is trained on noisy data, LDPKiT is inherently immune to membership inference attacks if $\mathcal{M}_{\mathrm{L}}$ is ever leaked.

Table 4.3: Normalized Zest distance results with *Cosine* distance metric on $\mathcal{M}_{\mathrm{R}}$ and $\mathcal{M}_{\mathrm{L}}$.

Dataset	$\mathcal{M}_{\mathrm{R}}$	$\mathcal{M}_{\mathrm{L}}$	LDP (ϵ=15)	LDP (ϵ=10)	LDP (ϵ=7)	LDP (ϵ=5)	LDP (ϵ=3)
CIFAR-10	ResNet-152	ResNet-18	1.6158 (±0.11)	1.5996 (±0.17)	1.6625 (±0.09)	2.0300 (±0.09)	2.4117 (±0.19)
		MobileNetV2	1.6214 (±0.40)	1.8454 (±0.48)	1.9999 (±0.33)	1.9877 (±0.46)	2.4639 (±0.15)
Fashion-MNIST	ResNet-152	ResNet-18	6.7820 (±0.35)	6.9049 (±0.46)	6.3640 (±0.30)	6.6078 (±0.97)	7.2531 (±0.70)
		MobileNetV2	4.0097 (±0.43)	4.9466 (±0.79)	5.0017 (±0.88)	5.5704 (±0.02)	4.7690 (±0.52)
SVHN	ResNet-152	ResNet-18	1.2497 (±0.13)	1.2374 (±0.04)	1.2303 (±0.07)	1.3165 (±0.02)	1.4662 (±0.07)
		MobileNetV2	1.2760 (±0.03)	1.2664 (±0.06)	1.3534 (±0.13)	1.3507 (±0.06)	1.4850 (±0.01)

The values recorded in parentheses are the standard deviations of the accuracies.

4.4 RQ3: Difference from Model Extraction

Although our method requires knowledge transfer from a remote model to a local model, it is different from the adversarial model extraction attack [43]–[47]. LDPKiT has orthogonal goals. The motivations for model stealing are often cost-driven, and model stealers aim to replicate the victim model's high performance with minimal queries, ensuring the theft requires less effort. In contrast,

Table 4.4: Normalized Zest distance results with l_1 distance metric on $\mathcal{M}_{\mathrm{R}}$ and $\mathcal{M}_{\mathrm{L}}$.

Dataset	$\mathcal{M}_{\mathrm{R}}$	$\mathcal{M}_{\mathrm{L}}$	LDP (ϵ=15)	LDP (ϵ=10)	LDP (ϵ=7)	LDP (ϵ=5)	LDP (ϵ=3)
CIFAR-10	ResNet-152	ResNet-18	1.9518 (±0.12)	1.9349 (±0.11)	2.0276 (±0.24)	2.4261 (±0.20)	2.4096 (±0.26)
		MobileNetV2	1.5970 (±0.35)	1.9575 (±0.62)	1.8453 (±0.20)	1.8167 (±0.39)	2.0948 (±0.27)
Fashion-MNIST	ResNet-152	ResNet-18	13.7488 (±4.68)	12.4642 (±2.38)	10.6427 (±0.57)	11.0902 (±6.00)	8.2408 (±1.75)
		MobileNetV2	7.4141 (±0.87)	2E+04 (±2E+04)	4E+05 (±7E+05)	2E+05 (±3E+05)	23.8558 (±33)
SVHN	ResNet-152	ResNet-18	1.0320 (±0.05)	1.0314 (±0.01)	1.0535 (±0.03)	1.0829 (±0.01)	1.1469 (±0.04)
		MobileNetV2	1.0807 (±0.01)	1.0693 (±0.01)	1.1058 (±0.04)	1.0980 (±0.04)	1.1501 (±0.01)

The values recorded in parentheses are the standard deviations of the accuracies.

Table 4.5: Normalized Zest distance results with l_2 distance metric on $\mathcal{M}_{\mathrm{R}}$ and $\mathcal{M}_{\mathrm{L}}$.

Dataset	$\mathcal{M}_{\mathrm{R}}$	$\mathcal{M}_{\mathrm{L}}$	LDP (ϵ=15)	LDP (ϵ=10)	LDP (ϵ=7)	LDP (ϵ=5)	LDP (ϵ=3)
CIFAR-10	ResNet-152	ResNet-18	1.7896 (±0.11)	1.7336 (±0.17)	1.8137 (±0.07)	2.1129 (±0.21)	2.3295 (±0.26)
		MobileNetV2	1.6507 (±0.60)	1.9063 (±0.66)	2.0075 (±0.66)	1.8500 (±0.70)	2.1929 (±0.32)
Fashion-MNIST	ResNet-152	ResNet-18	24.49 (±9.24)	24.89 (±6.65)	16.68 (±2.14)	21.39 (±17.70)	19.27 (±8.33)
		MobileNetV2	11.81 (±4.02)	2E+05 (±2E+05)	5E+06 (±8E+06)	3E+06 (±5E+06)	159 (±258)
SVHN	ResNet-152	ResNet-18	1.0570 (±0.05)	1.0505 (±0.01)	1.0520 (±0.02)	1.0935 (±0.01)	1.1600 (±0.03)
		MobileNetV2	1.1115 (±1E-3)	1.1062 (±0.03)	1.1332 (±0.05)	1.1405 (±0.02)	1.1859 (±0.01)

The values recorded in parentheses are the standard deviations of the accuracies.

Table 4.6: Normalized Zest distance results with l_∞ distance metric on $\mathcal{M}_{\mathrm{R}}$ and $\mathcal{M}_{\mathrm{L}}$.

Dataset	$\mathcal{M}_{\mathrm{R}}$	$\mathcal{M}_{\mathrm{L}}$	LDP (ϵ=15)	LDP (ϵ=10)	LDP (ϵ=7)	LDP (ϵ=5)	LDP (ϵ=3)
CIFAR-10	ResNet-152	ResNet-18	1.4464 (±0.13)	1.3011 (±0.45)	1.3546 (±0.12)	1.6950 (±0.57)	3.0270 (±1.39)
		MobileNetV2	2.9533 (±3.10)	2.2410 (±1.05)	5.8475 (±4.54)	2.9826 (±0.68)	3.1735 (±0.93)
Fashion-MNIST	ResNet-152	ResNet-18	55.25 (±22.98)	64.39 (±22.42)	26.29 (±6.97)	54.08 (±53.51)	53.06 (±20.29)
		MobileNetV2	22.24 (±10.35)	6E+05 (±7E+05)	3E+07 (±5E+07)	2E+07 (±3E+07)	711 (±1188)
SVHN	ResNet-152	ResNet-18	1.1684 (±0.07)	1.1758 (±0.09)	1.0846 (±0.11)	1.1766 (±0.09)	1.2666 (±0.13)
		MobileNetV2	1.3340 (±0.28)	1.5230 (±0.48)	1.2335 (±0.20)	1.5855 (±0.14)	1.6242 (±0.19)

The values recorded in parentheses are the standard deviations of the accuracies.

our ultimate goal is to protect the privacy of inference data. The reason the user stops querying $\mathcal{M}_{\mathrm{R}}$ is not necessarily due to monetary concerns but to prevent further privacy leakage.

We answer RQ3 quantitatively using *Zest* distances [22] as our evaluation metrics, which are the distances between two models computed based on LIME's model-agnostic explanations [105]. We use Zest because of its architecture independence, the model's black-box access requirement, and its perfect accuracy in model extraction detection with *Cosine* distance metric. Zest supports l_1, l_2, l_∞ norm and Cosine distances. The authors of Zest claim that it is capable of detecting model extraction attacks with 100% accuracy when using Cosine distance [22], which is the metric we present in Table 4.3. According to the *Zest* paper [22], we detect model extraction in the following steps:

1. Calculate the Zest distance between the two models to compare, *i.e.,* $D_z(\mathcal{M}_{\mathrm{R}}, \mathcal{M}_{\mathrm{L}})$, where $\mathcal{M}_{\mathrm{L}}$ is trained on the entire data split of the noisy $\mathcal{D}_{\mathrm{priv}}$, disjunctive to $\mathcal{M}_{\mathrm{R}}$'s training dataset.

2. Calculate a reference distance by computing the average distance between five pairs of the

victim and extracted models, denoted as $\mathcal{M}_{\mathrm{V}}$ and $\mathcal{M}_{\mathrm{E}}$, where $\mathcal{M}_{\mathrm{E}}$ are generated by training on $\mathcal{M}_{\mathrm{V}}$'s labelled training dataset, *i.e.,*

$$D_{ref} = \frac{1}{5}\sum_{i=1}^{5} D_z(\mathcal{M}_{\mathrm{Vi}}, \mathcal{M}_{\mathrm{Ei}}).$$

Here, $\mathcal{M}_{\mathrm{V}}$ has the same model architecture as $\mathcal{M}_{\mathrm{R}}$, and $\mathcal{M}_{\mathrm{E}}$ has the same model architecture as $\mathcal{M}_{\mathrm{L}}$, but trained on the same dataset as $\mathcal{M}_{\mathrm{V}}$, rather than the noisy $\mathcal{D}_{\mathrm{priv}}$.

3. Calculate the normalized Zest distance, *i.e.,* $\overline{D_z} = \frac{D_z}{D_{ref}}$.

4. Determine the existence of model extraction by comparing $\overline{D_z}$ with *threshold* 1.

   $\overline{D_z} < 1$ indicates $\mathcal{M}_{\mathrm{R}}$ and $\mathcal{M}_{\mathrm{L}}$ are similar models and model extraction occurs.

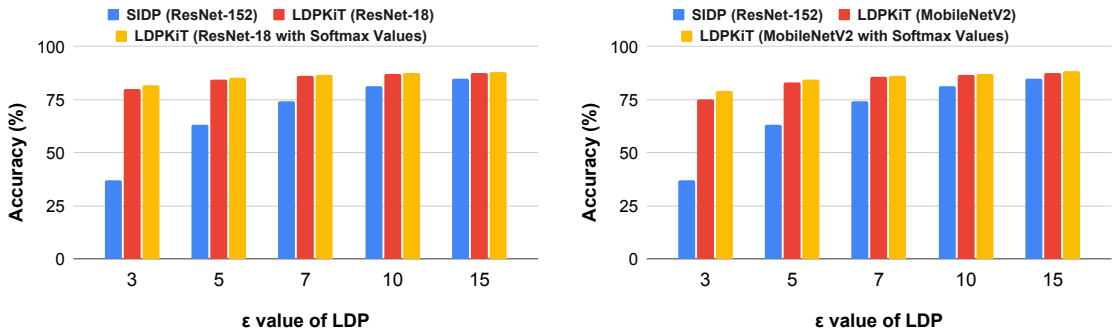
   $\overline{D_z} > 1$ indicates $\mathcal{M}_{\mathrm{R}}$ and $\mathcal{M}_{\mathrm{L}}$ are dissimilar, and thus no model extraction attack exists.

We present the normalized Zest distances, $\overline{D_z}$, with the Cosine distance metric in Table 4.3. We present the results of the rest of the distance metrics (l_1, l_2 and l_∞) in Tables 4.4, 4.5 and 4.6. According to the paper [22], Zest also has text modality support. However, that part of the code is not released to the public, so we can only report the results on image modality in our paper. As explained above, an adversarial model extraction attack occurs when $\overline{D_z} < 1$. Table 4.3 shows that LDPKiT does not contribute to model theft at any noise level since all $\overline{D_z} > 1$, and the distance increases as the noise level increases (*i.e.,* better privacy protection but farther from model extraction), which are the regimes that we expect LDPKiT to be used in.

From these results, we surmise that LDPKiT has several key differences from model stealing/extraction. First, one of the goals of a model extraction attack is to generate a model with similar performance as the victim model [42]. At fairly privacy-protective noise levels, such as $\epsilon = 5$ and $\epsilon = 3$, MobileNetV2 trained on CIFAR-10 only achieves roughly 83% and 75% accuracy, respectively, while $\mathcal{M}_{\mathrm{R}}$ has an accuracy of 88%—a significant gap in performance. Notably, since $\mathcal{M}_{\mathrm{L}}$ is not competitive with $\mathcal{M}_{\mathrm{R}}$, LDPKiT does not violate the terms of use for major commercial models [124]. Moreover, in LDPKiT, the user will stop training once accuracy gains diminish to protect privacy, so they are unlikely to achieve much higher accuracy. Second, model extraction attacks seek to maximize the accuracy of the extracted model with as few queries as possible. Efficient model extraction requires samples that are in the distribution of $\mathcal{M}_{\mathrm{R}}$'s training set [45]. Since LDPKiT queries inputs with noise added, they are not likely to be in the distribution $\mathcal{M}_{\mathrm{R}}$ is trained on and, thus, are not a query-efficient method for knowledge transfer. Finally, as demonstrated by our experiments with Zest, the extracted model does not behave very similarly to $\mathcal{M}_{\mathrm{R}}$, and its main benefit is in recovering utility for $\mathcal{D}_{\mathrm{priv}}$.

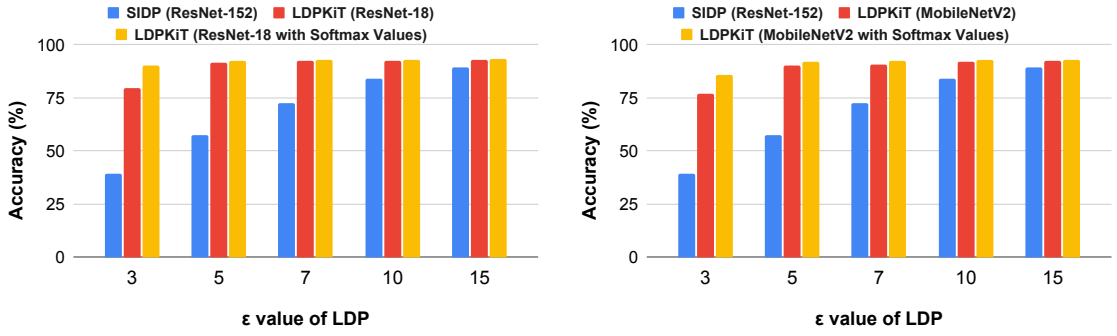
4.5 Auxiliary Experimental Results

This section shows two additional sets of experimental results on LDPKiT: (i) when $\mathcal{M}_{\mathrm{R}}$'s detailed softmax values are provided for training, and (ii) when the AL query selection strategy is applied.



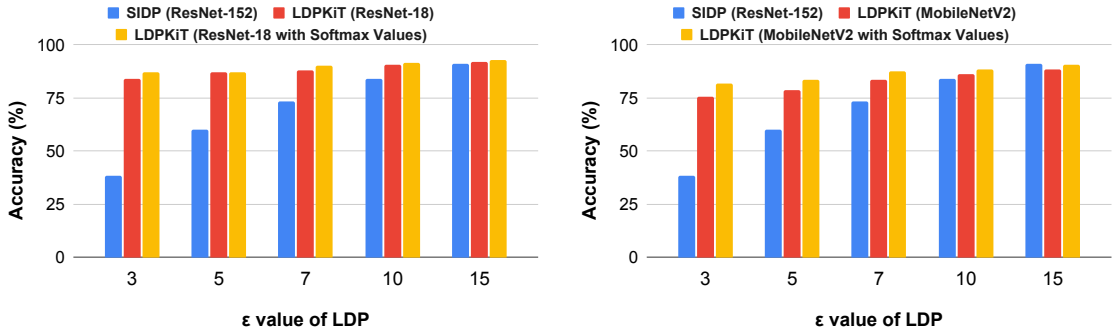
(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

Figure 4.13: Comparison of $\mathcal{M}_{\mathrm{L}}$'s prediction accuracies on $\mathcal{D}_{\mathrm{priv}}$ between SIDP and LDPKiT with/without $\mathcal{M}_{\mathrm{R}}$'s softmax values on CIFAR-10.



(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

Figure 4.14: Comparison of $\mathcal{M}_{\mathrm{L}}$'s prediction accuracies on $\mathcal{D}_{\mathrm{priv}}$ between SIDP and LDPKiT with/without $\mathcal{M}_{\mathrm{R}}$'s softmax values on Fashion-MNIST.



(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

Figure 4.15: Comparison of $\mathcal{M}_{\mathrm{L}}$'s prediction accuracies on $\mathcal{D}_{\mathrm{priv}}$ between SIDP and LDPKiT with/without $\mathcal{M}_{\mathrm{R}}$'s softmax values on SVHN.

4.5.1 LDPKiT with KLDiv Loss

As mentioned in Section 2.4 and 4.1, the loss function we use in LDPKiT (without AL) is CE loss (See Definition 2.4.1). In this section, we use a different loss function than CE. In Section 2.4, we

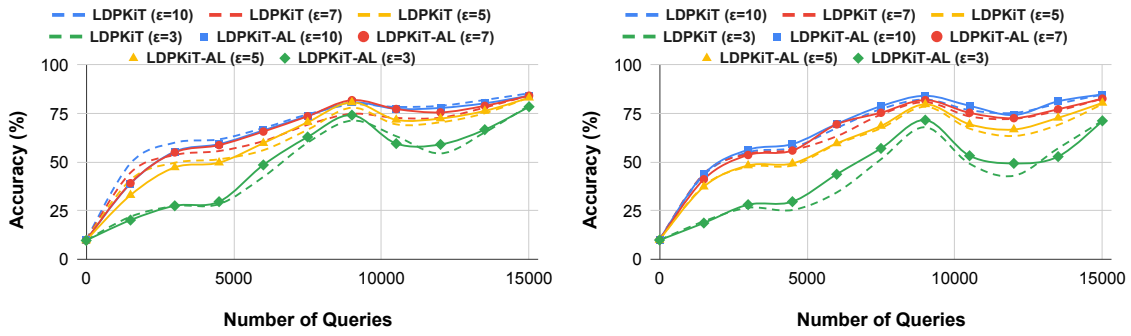
introduce a common loss function for knowledge distillation, KLDiv loss (See Definition 2.4.2). In this set of evaluations, we use a combined loss function, defined as follows:

Definition 4.5.1. Combined KLDiv + CE Loss Function for LDPKiT-AL

$$\text{Loss}_{\text{total}} = \alpha \cdot T^2 \cdot \text{KLDiv}(P_T \| Q_T) + \text{CE}(y, Q) \tag{4.1}$$

We follow the same notations as Section 2.4 in this equation, where P_T is $\mathcal{M}_{\text{R}}$'s softmax probability distribution with temperature scaling, Q is $\mathcal{M}_{\text{L}}$'s softmax probability distribution, and Q_T is the probability distribution Q softened with temperature scaling. $\alpha \cdot T^2$ is a scaling factor that determines the significance of KLDiv Loss. In our experiments, we set α to 1.0 for simplicity and T to 20.

The difference in accuracy is not significant whether or not $\mathcal{M}_{\text{R}}$'s softmax values are provided. As the bar graphs in Figures 4.13, 4.14 and 4.15 show, providing $\mathcal{M}_{\text{R}}$'s softmax probability distribution is more beneficial in a more privacy-protective noise level (*i.e.,* $\epsilon = 3$). For Fashion-MNIST, the prediction accuracy of ResNet-18 and MobileNetV2 on $\mathcal{D}_{\text{priv}}$ has increased by 10.4% and 8.7%, respectively, when trained with ResNet-152's softmax outputs with ϵ set to 3. However, when ϵ is increased to 15, softmax information brings a minimal increase in prediction accuracies. Therefore, for CIFAR-10, Fashion-MNIST, and SVHN image benchmarks, having access to $\mathcal{M}_{\text{R}}$'s softmax information does not help with utility recovery significantly, unless the ϵ value is very low. We suspect that providing softmax values would be more beneficial for a more complex dataset or when dealing with a more challenging ML task than supervised image classification.

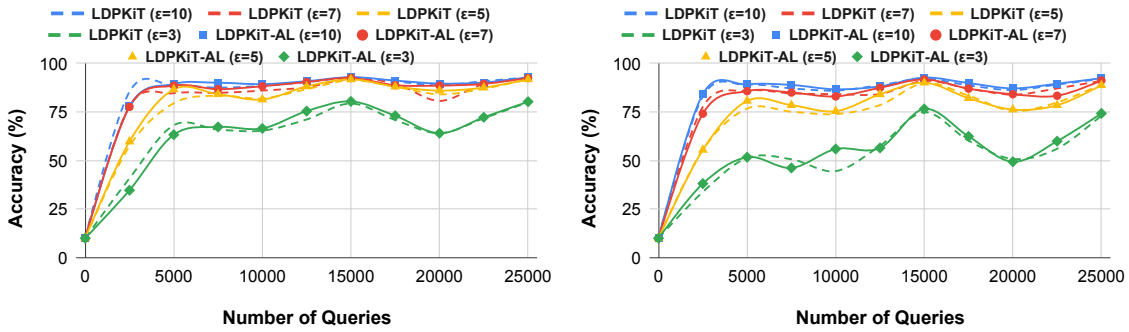


(a) $\mathcal{M}_{\text{L}}$ (ResNet-18) (b) $\mathcal{M}_{\text{L}}$ (MobileNetV2)

Figure 4.16: Effect of AL on $\mathcal{M}_{\text{L}}$'s accuracy on $\mathcal{D}_{\text{priv}}$ of CIFAR-10.

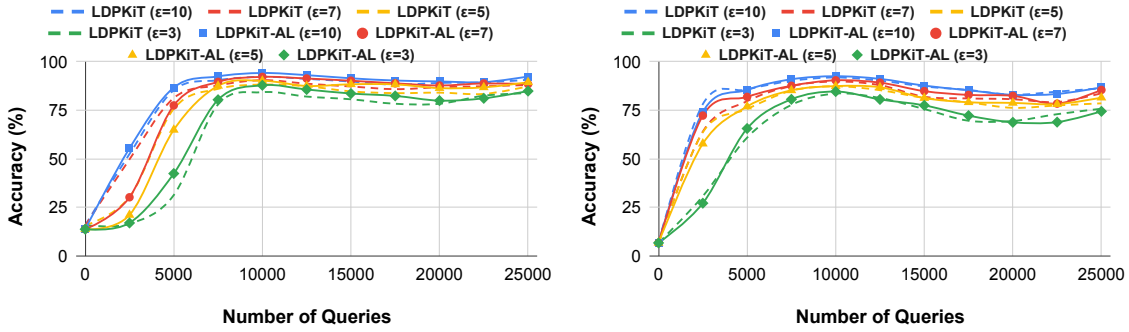
4.5.2 LDPKiT with AL

As mentioned in Chapter 3, LDPKiT is expected to work well with advanced training strategies such as AL. In this set of experiments, we apply Entropy Sampling as our query selection strategy (See Definition 2.3.4). In this section, we have a slightly different hyperparameter choice from previous sections (*i.e.,* Adam Optimizer for $\mathcal{M}_{\text{L}}$ training in LDPKiT in Sections 4.2 to 4.5.1), here we use SGD optimizer in training for both LDPKiT and LDPKiT-AL and present the accuracy comparisons in Figures 4.16 to 4.21. Note that the fluctuations in accuracy plots are not caused by the use of AL strategy, but brought by the use of SGD optimizer in training.



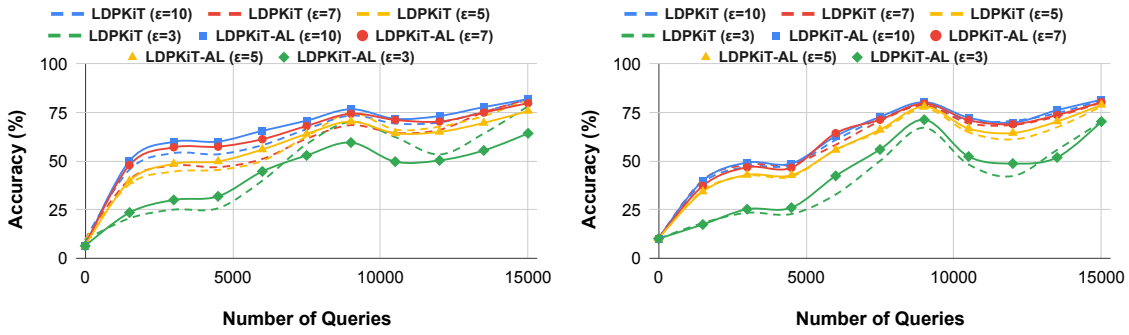
(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

Figure 4.17: Effect of AL on $\mathcal{M}_{\mathrm{L}}$'s accuracy on $\mathcal{D}_{\mathrm{priv}}$ of Fashion-MNIST.



(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

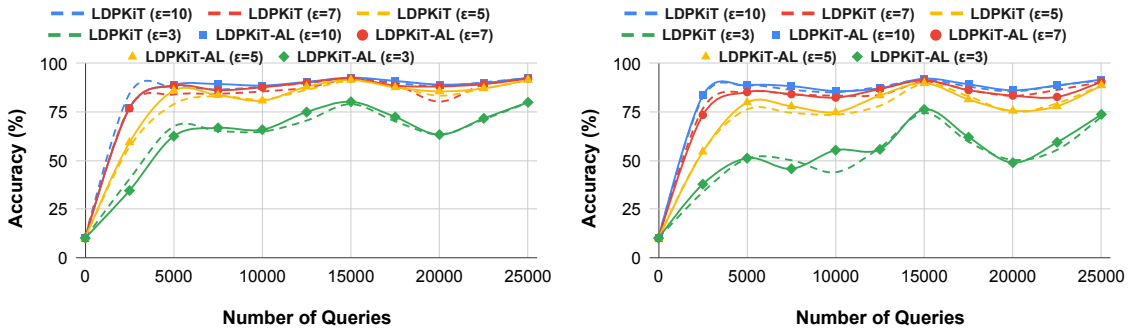
Figure 4.18: Effect of AL on $\mathcal{M}_{\mathrm{L}}$'s accuracy on $\mathcal{D}_{\mathrm{priv}}$ of SVHN.



(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

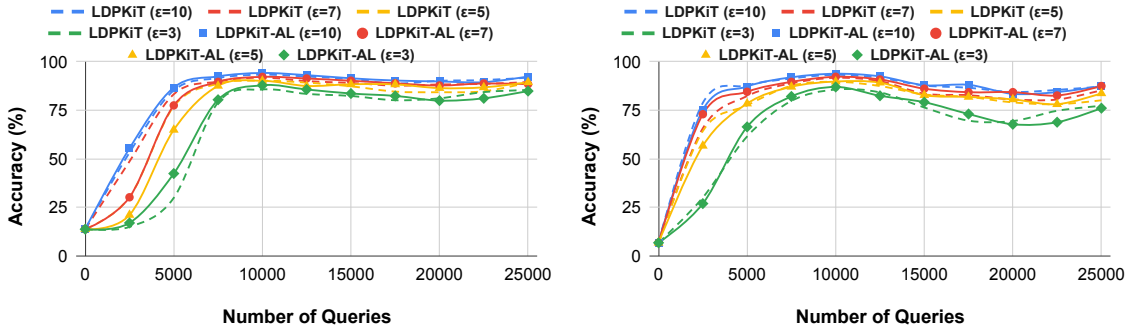
Figure 4.19: Effect of AL on $\mathcal{M}_{\mathrm{L}}$'s accuracy on $\mathcal{D}_{\mathrm{val}}$ of CIFAR-10.

The accuracies on $\mathcal{D}_{\mathrm{priv}}$ of the image benchmarks show that LDPKiT with AL (LDPKiT-AL) outperforms LDPKiT slightly after the first iteration(s) of training. This is anticipated because $\mathcal{M}_{\mathrm{L}}$ are initialized randomly, and the initial set of data points selected by AL can be considered as random sampling with a randomly initialized model. We also observe that when all the available data points in $\mathcal{D}_{\mathrm{priv}}$ have been queried, $\mathcal{M}_{\mathrm{L}}$'s accuracies on $\mathcal{D}_{\mathrm{priv}}$ are similar regardless of AL. This is also expected as, in both cases, the final training datasets will eventually be the same. As



(a) $\mathcal{M}_{\text{L}}$ (ResNet-18) (b) $\mathcal{M}_{\text{L}}$ (MobileNetV2)

Figure 4.20: Effect of AL on $\mathcal{M}_{\text{L}}$'s accuracy on $\mathcal{D}_{\text{val}}$ of Fashion-MNIST.



(a) $\mathcal{M}_{\text{L}}$ (ResNet-18) (b) $\mathcal{M}_{\text{L}}$ (MobileNetV2)

Figure 4.21: Effect of AL on $\mathcal{M}_{\text{L}}$'s accuracy on $\mathcal{D}_{\text{val}}$ of SVHN.

Figure 4.18 Subplot 4.18a shows, for example, when ϵ is set to 3 (*i.e.,* the green lines), LDPKiT-AL starts to outperform LDPKiT after the first iteration of training (after querying 2.5k data points). This advantage starts to disappear near the end of the training process (after querying 20k data points out of the total 25k data points in $\mathcal{D}_{\text{priv}}$). In the end, LDPKiT and LDPKiT-AL achieves similar prediction accuracies on $\mathcal{D}_{\text{priv}}$ of SVHN, 83.81% and 84.87%, respectively. The relationship between the accuracy gap of LDPKiT and LDPKiT-AL and the ϵ values is unclear. There also exist few instances where LDPKiT and LDPKiT-AL have similar accuracies on $\mathcal{D}_{\text{priv}}$, such as all ϵ cases in Figure 4.17 Subplot 4.17b.

The effect of AL on accuracies on $\mathcal{D}_{\text{val}}$ are shown in Figures 4.19, 4.20, and 4.21. The accuracy difference is more obvious in CIFAR-10. For instance, as the red lines in Figure 4.19 Subplot 4.19a show, the accuracy gap between LDPKiT-AL and LDPKiT can reach 10% when $\epsilon = 7$. LDPKiT-AL demonstrates similar accuracies as LDPKiT in Fashion-MNIST and SVHN benchmarks. We suspect that it is because these two datasets are less complex to train on. Similar to the observations on $\mathcal{D}_{\text{priv}}$, there is no explicit relationship between the accuracy gap and the ϵ values.

Overall, the accuracy improvement brought by AL is not significant, except for CIFAR-10. We suspect that it is because we have a small $\mathcal{D}_{\text{priv}}$ (*i.e.,* 15k, 25k, and 25k for CIFAR-10, Fashion-MNIST, and SVHN, respectively), and the improvement is dataset- and data sample-dependent. AL's effectiveness may increase when we expand $\mathcal{D}_{\text{priv}}$ or evaluate on more complicated datasets

and ML tasks. Further investigations are needed to thoroughly study the contribution of AL to utility recovery.

Chapter 5

Limitations and Future Work

5.1 Relationship between privacy guarantee and assumptions on queries in $\mathcal{D}_{\text{priv}}$

As discussed in Chapter 3, LDPKiT provides a form of plausible deniability such that an adversary cannot infer the user's original data input. The per-query ϵ-LDP privacy guarantee only holds under the assumption that each data point in $\mathcal{D}_{\text{priv}}$ is *i.i.d.*. If the data points are not *i.i.d.*, our privacy guarantee will be weakened by their mutual information.

5.2 Dataset and task extension

As for the limitations in evaluation, we only tested on supervised learning, specifically classification tasks. One future work is to study the privacy risks and protection in other learning problems and extend the evaluation to regression tasks or unsupervised clustering tasks. Moreover, our current evaluation is on relatively small datasets. We can expand our evaluation on larger and more complex datasets to study the effect of the cloud model's softmax output information and the AL training strategy on utility recovery. Furthermore, as discussed in Section 1.2, we can extend the evaluation to the audio modality.

5.3 Alternative selection/generation strategies

Also, our privacy-preserving queries in the thesis only refer to sensitive queries with LDP noise applied. As a future direction, we can design alternative query selection or generation strategies. In other words, instead of adding noise to the original queries for privacy protection, we can compose our privacy-preserving (sanitized) $\mathcal{D}_{\text{priv}}$. For instance, we can explore synthetic query generation techniques (*i.e.,* generate substitute image data with GANs and text data with LLMs). Another area for further exploration in the image modality is to study the latent space of image data. One of the query selection strategies can be using similarity metrics (*e.g.,* Cosine Similarity) to choose comparable queries from existing public datasets. These selected public queries can then replace sensitive queries in $\mathcal{D}_{\text{priv}}$ based on their latent space information.

5.3.1 Different LDP noise insertion point

In LDPKiT, we only considered the case of inserting LDP noise into the inference data inputs. With the above-mentioned synthetic query generation techniques, we can further explore the possibility of using differentially private GANs and LLMs for data generation.

5.3.2 Different points to apply similarity metrics

To study and compare the latent space of two images using similarity metrics, we typically consider comparing the feature vectors from the penultimate layer of the ML model that processes the data. However, more possibilities exist, such as comparing the vectors from preceding layers.

5.4 Deployment on real-world frameworks

As a next step, we can deploy LDPKiT in a more realistic use case scenario and evaluate its accuracy/privacy trade-off. One potential platform for this evaluation can be an edge computing network, such as StarlingX [125], an open-source, distributed cloud edge native platform.

Chapter 6

Conclusion

LDPKiT is an inference framework that preserves the privacy of data in a sensitive dataset when using malicious cloud services by injecting LDP noise. Since all the privacy protection measures are applied before transmitting to the remote server, the protection still exists even if the cloud service is deployed on a compromised platform or the cloud model is leaked. The key insight is that partial knowledge about the real data, though noisy, still exists in the noisy labels returned from the cloud model. LDPKiT put this insight into use by aggregating the partial knowledge from noisy queries and then training a local model with the knowledge. The experimental results demonstrate that LDPKiT can recover the prediction accuracy on private data throughout training while preserving privacy. LDPKiT has greater benefits when the noise level increases, which are the more privacy-protective regimes we expect LDPKiT to be used in. We also quantitatively demonstrate that the level of knowledge transfer in LDPKiT does not construct an adversarial model extraction attack. For future steps, alternative strategies for query selection and generation can be explored. Additionally, further evaluation of LDPKiT can be conducted on more complex datasets, advanced ML tasks, and diverse use case scenarios.

Bibliography

[1] M. Ribeiro, K. Grolinger, and M. A. Capretz, "MLaaS: Machine Learning as a Service," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 896–902. DOI: 10.1109/ICMLA.2015.152.

[2] A. Esteva, A. Robicquet, B. Ramsundar, *et al.*, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[3] A. Singh, D. Patil, and S. Omkar, "Eye in the Sky: Real-Time Drone Surveillance System (DSS) for Violent Individuals Identification Using ScatterNet Hybrid Deep Learning Network," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1710–17 108. DOI: 10.1109/CVPRW.2018.00214.

[4] A. Hannun, C. Guo, and L. van der Maaten, "Measuring data leakage in machine-learning models with Fisher information," in *Uncertainty in Artificial Intelligence*, PMLR, 2021, pp. 760–770.

[5] Y. A. Hamza and M. D. Omar, "Cloud computing security: abuse and nefarious use of cloud computing," *Int. J. Comput. Eng. Res*, vol. 3, no. 6, pp. 22–27, 2013.

[6] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49–58, 2019.

[7] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *2017 IEEE symposium on security and privacy (SP)*, IEEE, 2017, pp. 3–18.

[8] C. Savage and N. Perlroth, *Yahoo said to have aided U.S. email surveillance by adapting spam filter*, Oct. 2016. [Online]. Available: https://www.nytimes.com/2016/10/06/technology/yahoo-email-tech-companies-government-investigations.html.

[9] P. Dave and B. Bennet, *Yahoo helped the U.S. Government spy on emails, report says*, 2016. [Online]. Available: https://www.latimes.com/business/technology/la-fi-tn-yahoo-email-20161004-snap-story.html.

[10] D. Kaye, *Reports that Yahoo aided us e-mail surveillance draw concern of UN Human Rights Expert — UN News*, 2016. [Online]. Available: https://news.un.org/en/story/2016/10/542152.

[11] A. NG, *Amazon gave ring videos to police without owners' permission*, 2022. [Online]. Available: https://www.politico.com/news/2022/07/13/amazon-gave-ring-videos-to-police-without-owners-permission-00045513.

[12] S. Ray, *Apple joins a growing list of companies cracking down on use of chatgpt by staffers-here's why*, 2023. [Online]. Available: https://www.forbes.com/sites/siladityaray/2023/05/19/apple-joins-a-growing-list-of-companies-cracking-down-on-use-of-chatgpt-by-staffers-heres-why/.

[13] N. Gordon, *Apple restricts employee chatgpt use as companies worry about data leaks*, 2023. [Online]. Available: https://fortune.com/2023/05/19/apple-restricts-chatgpt-employee-data-leaks-iphone/.

[14] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *IEEE Symposium on Security and Privacy*, 2015.

[15] 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," Tech. Rep. MSR-TR-2016-3, 2016. [Online]. Available: https://www.microsoft.com/en-us/research/publication/cryptonets-applying-neural-networks-to-encrypted-data-with-high-throughput-and-accuracy.

[16] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, *Gazelle: A Low Latency Framework for Secure Neural Network Inference*, 2018. DOI: 10.48550/ARXIV.1801.05507. [Online]. Available: https://arxiv.org/abs/1801.05507.

[17] F. Tramèr and D. Boneh, *Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware*, 2019. arXiv: 1806.03287 [stat.ML].

[18] X. Lou, T. Zhang, J. Jiang, and Y. Zhang, *A Survey of Microarchitectural Side-channel Vulnerabilities, Attacks and Defenses in Cryptography*, 2021. arXiv: 2103.14244 [cs.CR].

[19] J. Van Bulck, M. Minkin, O. Weisse, *et al.*, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient Out-of-Order execution," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 991–1008.

[20] M. Schwarz, M. Lipp, D. Moghimi, *et al.*, "ZombieLoad: Cross-privilege-boundary data sampling," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 753–768.

[21] S. Leroux, T. Verbelen, P. Simoens, and B. Dhoedt, *Privacy Aware Offloading of Deep Neural Networks*, 2018. arXiv: 1805.12024 [cs.LG].

[22] H. Jia, H. Chen, J. Guan, A. S. Shamsabadi, and N. Papernot, "A Zest of LIME: Towards Architecture-Independent Model Distances," in *International Conference on Learning Representations*, 2021.

[23] M. Haris, H. Haddadi, and P. Hui, "Privacy leakage in mobile computing: Tools, methods, and characteristics," *arXiv preprint arXiv:1410.4978*, 2014.

[24] D. Bennett, *Pope Francis' white puffer coat AI image sparks deep fake concerns*, 2023. [Online]. Available: https://www.bloomberg.com/news/newsletters/2023-04-06/pope-francis-white-puffer-coat-ai-image-sparks-deep-fake-concerns.

[25] T. Brooks, P. G., J. Heatley, *et al.*, *Increasing Threats of DeepFake Identities*. [Online]. Available: https://www.dhs.gov/sites/default/files/publications/increasing_threats_of_deepfake_identities_0.pdf.

[26] J. C. Wong, *The Cambridge Analytica scandal changed the world – but it didn't change Facebook*, 2019. [Online]. Available: https://www.theguardian.com/technology/2019/mar/17/the-cambridge-analytica-scandal-changed-the-world-but-it-didnt-change-facebook.

[27] J. Isaak and M. J. Hanna, "User data privacy: Facebook, Cambridge Analytica, and privacy protection," *Computer*, vol. 51, no. 8, pp. 56–59, 2018.

[28] E. Francis, *FCC warns consumers about new "Yes" phone scam*, 2019. [Online]. Available: https://abcnews.go.com/Business/fcc-warns-consumers-phone-scam/story?id=46405703.

[29] M. Cerullo, *Cybercriminals are using AI Voice Cloning Tools to dupe victims*, 2023. [Online]. Available: https://www.cbsnews.com/news/ai-scam-voice-cloning-rising/.

[30] IBM, *About IBM Cloud Internet Services*, 2022. [Online]. Available: https://cloud.ibm.com/docs/cis?topic=cis-about-ibm-cloud-internet-services-cis.

[31] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012, ISBN: 0262018020.

[32] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

[33] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[34] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *Int. J. Comput. Vision*, vol. 129, no. 6, 1789–1819, 2021, ISSN: 0920-5691. DOI: 10.1007/s11263-021-01453-z. [Online]. Available: https://doi.org/10.1007/s11263-021-01453-z.

[35] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, *FitNets: Hints for Thin Deep Nets*, 2015. arXiv: 1412.6550 [cs.LG].

[36] K. Xu, L. Rui, Y. Li, and L. Gu, "Feature normalized knowledge distillation for image classification," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, Springer, 2020, pp. 664–680.

[37] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[38] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1365–1374.

[39] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational Knowledge Distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[40] Z. Meng, J. Li, Y. Zhao, and Y. Gong, "Conditional Teacher-student Learning," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019. DOI: 10.1109/icassp.2019.8683438. [Online]. Available: https://doi.org/10.1109\%2Ficassp.2019.8683438.

[41] 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, ISBN: 978-1-939133-17-5.

[42] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs.," in *USENIX security symposium*, vol. 16, 2016, pp. 601–618.

[43] 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, Abu Dhabi, United Arab Emirates: Association for Computing Machinery, 2017, 506–519, ISBN: 9781450349444. DOI: 10.1145/3052973.3053009. [Online]. Available: https://doi.org/10.1145/3052973.3053009.

[44] J. Zhang, C. Chen, and L. Lyu, "IDEAL: Query-Efficient Data-Free Learning from Black-Box Models," in *The Eleventh International Conference on Learning Representations*, 2022.

[45] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4771–4780.

[46] 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 *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 865–872. DOI: 10.1609/AAAI.V34I01.5432. [Online]. Available: https://doi.org/10.1609/aaai.v34i01.5432.

[47] T. Orekondy, B. Schiele, and M. Fritz, *Knockoff Nets: Stealing Functionality of Black-Box Models*, 2018. arXiv: 1812.02766 [cs.CV].

[48] B. D. Rouhani, H. Chen, and F. Koushanfar, *DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models*, 2018. arXiv: 1804.00750 [cs.CR].

[49] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding Watermarks into Deep Neural Networks," in *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ACM, 2017. DOI: 10.1145/3078971.3078974. [Online]. Available: https://doi.org/10.1145\%2F3078971.3078974.

[50] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, *Entangled Watermarks as a Defense against Model Extraction*, 2021. arXiv: 2002.12200 [cs.CR].

[51] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, *Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring*, 2018. arXiv: 1802.04633 [cs.LG].

[52] H. Chen, B. D. Rouhani, C. Fu, J. Zhao, and F. Koushanfar, "DeepMarks: A Secure Fingerprinting Framework for Digital Rights Management of Deep Learning Models," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, ser. ICMR '19, Ottawa ON, Canada: Association for Computing Machinery, 2019, 105–113, ISBN: 9781450367653.

DOI: 10.1145/3323873.3325042. [Online]. Available: https://doi.org/10.1145/3323873.3325042.

[53] S. Wang and C.-H. Chang, "Fingerprinting Deep Neural Networks - a DeepFool Approach," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401119.

[54] A. Dziedzic, M. A. Kaleem, Y. S. Lu, and N. Papernot, *Increasing the Cost of Model Extraction with Calibrated Proof of Work*, 2022. arXiv: 2201.09243 [cs.CR].

[55] Y. Liu, K. Li, Z. Liu, *et al.*, "Provenance of Training without Training Data: Towards Privacy-Preserving DNN Model Ownership Verification," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23, Austin, TX, USA: Association for Computing Machinery, 2023, 1980–1990, ISBN: 9781450394161. DOI: 10.1145/3543507.3583198. [Online]. Available: https://doi.org/10.1145/3543507.3583198.

[56] A. Dziedzic, N. Dhawan, M. A. Kaleem, J. Guan, and N. Papernot, *On the Difficulty of Defending Self-Supervised Learning against Model Extraction*, 2022. arXiv: 2205.07890 [cs.LG].

[57] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975. DOI: 10.1109/PROC.1975.9939.

[58] M. Abadi, U. Erlingsson, I. Goodfellow, *et al.*, "On the Protection of Private Information in Machine Learning Systems: Two Recent Approches," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, IEEE, 2017. DOI: 10.1109/csf.2017.10. [Online]. Available: https://doi.org/10.1109\%2Fcsf.2017.10.

[59] M. Fredrikson, S. Jha, and T. Ristenpart, "Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, Denver, Colorado, USA: Association for Computing Machinery, 2015, 1322–1333, ISBN: 9781450338325. DOI: 10.1145/2810103.2813677. [Online]. Available: https://doi.org/10.1145/2810103.2813677.

[60] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, *When the Curious Abandon Honesty: Federated Learning Is Not Private*, 2023. arXiv: 2112.02918 [cs.LG].

[61] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, *Inverting Gradients – How easy is it to break privacy in federated learning?* 2020. arXiv: 2003.14053.

[62] J. C. Zhao, A. Sharma, A. R. Elkordy, Y. H. Ezzeldin, S. Avestimehr, and S. Bagchi, *LOKI: Large-scale Data Reconstruction Attack against Federated Learning through Model Manipulation*, 2023. arXiv: 2303.12233 [cs.LG].

[63] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Úlfar Erlingsson, *Scalable Private Learning with PATE*, 2018. arXiv: 1802.08908 [stat.ML].

[64] M. Abadi, A. Chu, I. Goodfellow, *et al.*, "Deep Learning with Differential Privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016. DOI: 10.1145/2976749.2978318. [Online]. Available: https://doi.org/10.1145\%2F2976749.2978318.

[65] Y. Zhu, X. Yu, M. Chandraker, and Y.-X. Wang, "Private-kNN: Practical Differential Privacy for Computer Vision," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 851–11 859. DOI: 10.1109/CVPR42600.2020.01187.

[66] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, *MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples*, 2019. arXiv: 1909.10594 [cs.CR].

[67] L. Hanzlik, Y. Zhang, K. Grosse, *et al.*, "MLCapsule: Guarded Offline Deployment of Machine Learning as a Service," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 3295–3304. DOI: 10.1109/CVPRW53098.2021.00368.

[68] J. Hayes, L. Melis, G. Danezis, and E. D. Cristofaro, *LOGAN: Membership Inference Attacks Against Generative Models*, 2018. arXiv: 1705.07663 [cs.CR].

[69] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, *Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer*, 2019. arXiv: 1912.11279 [stat.ML].

[70] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in classification models," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '21, Virtual Event, USA: Association for Computing Machinery, 2021, 5–16, ISBN: 9781450381437. DOI: 10.1145/3422337.3447836. [Online]. Available: https://doi.org/10.1145/3422337.3447836.

[71] S. V. Dibbo, "Sok: Model inversion attack landscape: Taxonomy, challenges, and future roadmap," in *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, IEEE, 2023, pp. 439–456.

[72] K. Bonawitz, V. Ivanov, B. Kreuter, *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, Dallas, Texas, USA: Association for Computing Machinery, 2017, 1175–1191, ISBN: 9781450349468. DOI: 10.1145/3133956.3133982. [Online]. Available: https://doi.org/10.1145/3133956.3133982.

[73] H. Fereidooni, S. Marchal, M. Miettinen, *et al.*, "SAFELearn: Secure Aggregation for private FEderated Learning," in *2021 IEEE Security and Privacy Workshops (SPW)*, Los Alamitos, CA, USA: IEEE Computer Society, 2021, pp. 56–62. DOI: 10.1109/SPW53761.2021.00017. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SPW53761.2021.00017.

[74] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, *Communication-Efficient Learning of Deep Networks from Decentralized Data*, 2023. arXiv: 1602.05629 [cs.LG].

[75] M. Lipp, M. Schwarz, D. Gruss, *et al.*, "Meltdown: Reading kernel memory from user space," *Communications of the ACM*, vol. 63, no. 6, pp. 46–56, 2020.

[76] 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, Abu Dhabi, United Arab Emirates: Association for Computing Machinery, 2017, 506–519, ISBN: 9781450349444. DOI: 10.1145/3052973.3053009. [Online]. Available: https://doi.org/10.1145/3052973.3053009.

[77] I. Shumailov, Z. Shumaylov, D. Kazhdan, *et al.*, *Manipulating SGD with Data Ordering Attacks*, 2021. arXiv: 2104.09667 [cs.LG].

[78] A. S. Rakin, Z. He, and D. Fan, *Bit-Flip Attack: Crushing Neural Network with Progressive Bit Search*, 2019. arXiv: 1903.12269 [cs.CV].

[79] B. Biggio, I. Corona, D. Maiorca, *et al.*, "Evasion attacks against machine learning at test time," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*, Springer, 2013, pp. 387–402.

[80] I. Shumailov, Y. Zhao, D. Bates, N. Papernot, R. Mullins, and R. Anderson, *Sponge Examples: Energy-Latency Attacks on Neural Networks*, 2021. arXiv: 2006.03463 [cs.LG].

[81] E. Wong and Z. Kolter, "Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 5286–5295. [Online]. Available: https://proceedings.mlr.press/v80/wong18a.html.

[82] H. Xu, Y. Ma, H. Liu, *et al.*, *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review*, 2019. arXiv: 1909.08072 [cs.LG].

[83] B. Wu, S. Wei, M. Zhu, *et al.*, *Defenses in adversarial machine learning: A survey*, 2023. arXiv: 2312.08890 [cs.CV].

[84] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial machine learning attacks and defense methods in the cyber security domain," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[85] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, Springer, 2006, pp. 265–284.

[86] U. Erlingsson, V. Pihur, and A. Korolova, "RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14, Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, 1054–1067, ISBN: 9781450329576. DOI: 10.1145/2660267.2660348. [Online]. Available: https://doi.org/10.1145/2660267.2660348.

[87] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, IEEE, 2017. DOI: 10.1109/csf.2017.11. [Online]. Available: https://doi.org/10.1109\%2Fcsf.2017.11.

[88] C. Dwork, "Differential privacy: A survey of results," in *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings 5*, Springer, 2008, pp. 1–19.

[89] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, IEEE, 2010, pp. 51–60.

[90] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3–4, 211–407, 2014, ISSN: 1551-305X. DOI: 10.1561/0400000042. [Online]. Available: https://doi.org/10.1561/0400000042.

[91] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[92] L. Lyu, J. C. Bezdek, J. Jin, and Y. Yang, "FORESEEN: Towards Differentially Private Deep Inference for Intelligent Internet of Things," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2418–2429, 2020. DOI: 10.1109/JSAC.2020.3000374.

[93] F. Mireshghallah, M. Taram, P. Ramrakhyani, A. Jalali, D. Tullsen, and H. Esmaeilzadeh, "Shredder: Learning noise distributions to protect inference privacy," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 3–18.

[94] S. A. Osia, A. Shahin Shamsabadi, S. Sajadmanesh, *et al.*, "A Hybrid Deep Learning Architecture for Privacy-Preserving Mobile Analytics," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4505–4518, 2020. DOI: 10.1109/JIOT.2020.2967734.

[95] 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*, 2018. DOI: 10.48550/ARXIV.1809.03428. [Online]. Available: https://arxiv.org/abs/1809.03428.

[96] B. Settles, "Active learning literature survey," 2009.

[97] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active Learning from Data Streams," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 2007, pp. 757–762. DOI: 10.1109/ICDM.2007.101.

[98] C. X. Ling and J. Du, "Active learning with direct query construction," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, 480–487, ISBN: 9781605581934. DOI: 10.1145/1401890.1401950. [Online]. Available: https://doi.org/10.1145/1401890.1401950.

[99] H. T. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04, Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 79, ISBN: 1581138385. DOI: 10.1145/1015330.1015349. [Online]. Available: https://doi.org/10.1145/1015330.1015349.

[100] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[101] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active Learning by Querying Informative and Representative Examples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 10, pp. 1936–1949, 2014. DOI: 10.1109/TPAMI.2014.2307881.

[102] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds., San Francisco (CA): Morgan Kaufmann, 1994, pp. 148–156, ISBN: 978-1-55860-335-6. DOI: https://doi.org/10.1016/B978-1-55860-335-6.50026-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978155860335650026X.

[103] V.-L. Nguyen, M. Shaker, and E. Hüllermeier, "How to measure uncertainty in uncertainty sampling for active learning," *Machine Learning*, vol. 111, Jan. 2022. DOI: 10.1007/s10994-021-06003-9.

[104] Y. Xi, *Privacy Preserving Inference via Model Extraction Against Third-Party NLP Cloud Adversaries*, 2024.

[105] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?"Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[106] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, *High Accuracy and High Fidelity Extraction of Neural Networks*, 2020. arXiv: 1909.01838 [cs.LG].

[107] X. Yue, M. Du, T. Wang, Y. Li, H. Sun, and S. S. Chow, "Differential Privacy for Text Analytics via Natural Text Sanitization," *arXiv preprint arXiv:2106.01221*, 2021.

[108] S. Kim, D. Le, W. Zheng, *et al.*, *Evaluating User Perception of Speech Recognition System Quality with Semantic Distance Metric*, 2022. arXiv: 2110.05376 [cs.CL].

[109] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *BERTScore: Evaluating Text Generation with BERT*, 2020. arXiv: 1904.09675 [cs.CL].

[110] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger, "MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 563–578. DOI: 10.18653/v1/D19-1053. [Online]. Available: https://aclanthology.org/D19-1053.

[111] W. Jiang, Z. Ma, S. Li, H. Xiao, and J. Yang, "Privacy budget management and noise reusing in multichain environment," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 10 462–10 475, 2022.

[112] U. Hassan and M. S. Anwar, "Reducing noise by repetition: Introduction to signal averaging," *European Journal of Physics*, vol. 31, no. 3, p. 453, 2010.

[113] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[114] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[115] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *MobileNetV2: Inverted Residuals and Linear Bottlenecks*, 2019. arXiv: 1801.04381 [cs.CV].

[116] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[117] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[118] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention Is All You Need*, 2023. arXiv: 1706.03762 [cs.CL].

[119] 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, 2018, pp. 3687–3697. DOI: 10.18653/v1/D18-1404. [Online]. Available: https://www.aclweb.org/anthology/D18-1404.

[120] Apple, *Differential Privacy*, https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.

[121] A. Orr, *Google's differential privacy may be better than Apple's*, 2017. [Online]. Available: https://www.macobserver.com/analysis/google-apple-differential-privacy/.

[122] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[123] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[124] OpenAI, *Terms of use*, https://openai.com/policies/terms-of-use/.

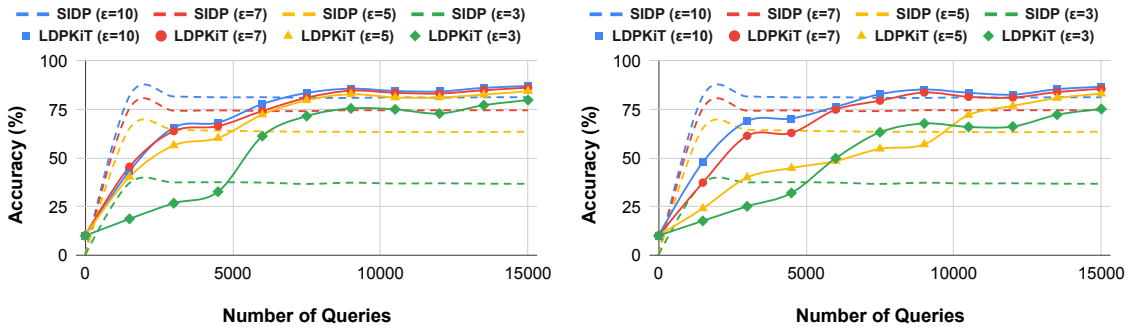
[125] OpenStack Foundation, *Starlingx: Open source cloud infrastructure for the distributed edge*, Version 5.0, 2024. [Online]. Available: https://www.starlingx.io.

Appendix

6.1 Alternative presentation of accuracy comparisons between LDPKiT and SIDP in RQ2

In Section 4.3, we compare LDPKiT's accuracy on a growing $\mathcal{D}_{\text{priv}}$ with the averaged SIDP accuracy on the entire $\mathcal{D}_{\text{priv}}$. Alternatively, we can present the accuracy plots of LDPKiT and SIDP on a varying $\mathcal{D}_{\text{priv}}$ as we increase the number of queries in the following Figures 6.1, 6.2 and 6.3.

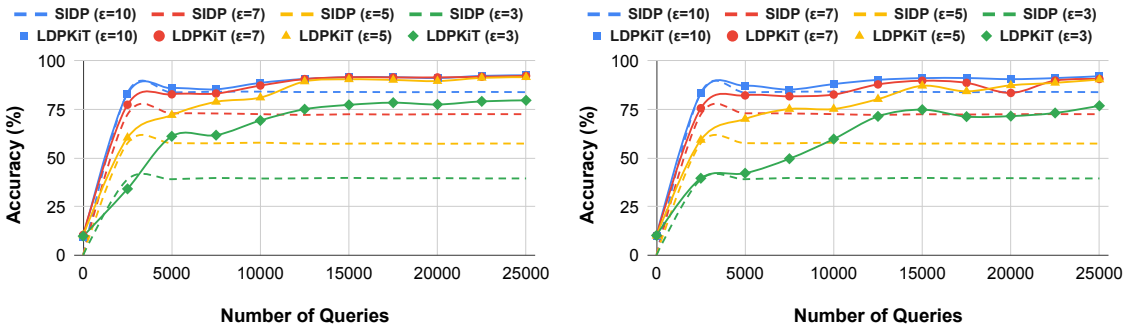
Since no training is involved in the SIDP evaluation process, $\mathcal{D}_{\text{val}}$ remains constant; therefore, this alternative presentation of accuracy plots does not apply to $\mathcal{D}_{\text{val}}$.



(a) $\mathcal{M}_{\text{L}}$ (ResNet-18)

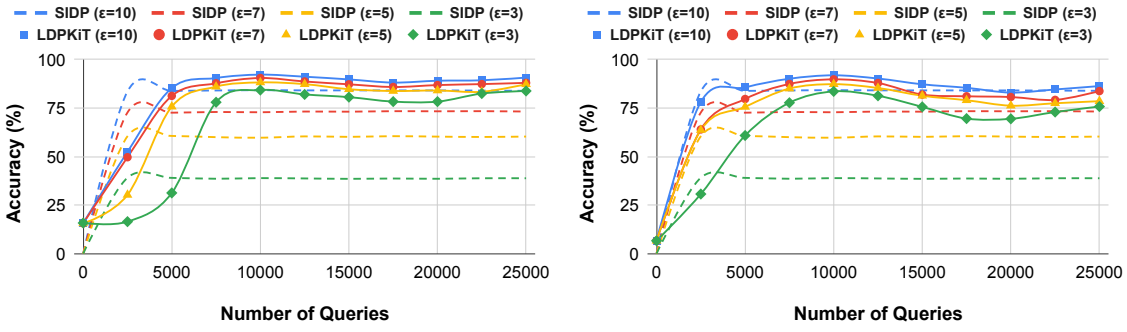
(b) $\mathcal{M}_{\text{L}}$ (MobileNetV2)

Figure 6.1: $\mathcal{M}_{\text{L}}$'s accuracies on $\mathcal{D}_{\text{priv}}$ of CIFAR-10.



(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

Figure 6.2: $\mathcal{M}_{\mathrm{L}}$'s accuracies on $\mathcal{D}_{\mathrm{priv}}$ of Fashion-MNIST.



(a) $\mathcal{M}_{\mathrm{L}}$ (ResNet-18) (b) $\mathcal{M}_{\mathrm{L}}$ (MobileNetV2)

Figure 6.3: $\mathcal{M}_{\mathrm{L}}$'s accuracies on $\mathcal{D}_{\mathrm{priv}}$ of SVHN.