Differentially Private Neural Tangent Kernels for Privacy-Preserving Data Generation and Distillation

by

Yilin Yang

B.Sc., University of British Columbia, 2022

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

April 2024

© Yilin Yang, 2024
The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

**Differentially Private Neural Tangent Kernels for Privacy-Preserving Data Generation and Distillation**

submitted by **Yilin Yang** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

**Examining Committee:**

Mi Jung Park, Assistant Professor, Department of Computer Science, UBC  
*Co-supervisor*

Xiaoxiao Li, Assistant Professor, Department of Electrical and Computer Engineering, UBC  
*Co-supervisor*

Danica J. Sutherland, Assistant Professor, Department of Computer Science, UBC  
*Supervisory Committee Member*
Abstract

With the increasing interest in Deep Learning, data safety issues have become more prevalent as we rely more on Artificial Intelligence. Adversaries can easily obtain sensitive information through various attacks, this dramatically discourages patients and clients from contributing invaluable data that may be beneficial to research. This problem facilitates the need for a gold standard privacy notion. In recent years, Differential Privacy (DP) has been recognized as a gold standard notion of privacy. Among the current popular DP methods, Maximum mean discrepancy (MMD) is a particularly useful distance metric for differentially private data generation. When used with finite-dimensional features it allows us to summarize and privatize the data distribution once, which we can repeatedly use during generator training without further privacy loss. An important question in this framework is, then, what features are useful to distinguish between real and synthetic data distributions, and whether those enable us to generate quality synthetic data. This work considers using the features of neural tangent kernels (NTKs), more precisely empirical NTKs (e-NTKs). We find that, perhaps surprisingly, the expressiveness of the untrained e-NTK features is comparable to that of the features taken from pre-trained perceptual features using public data. As a result, our method improves the privacy-accuracy trade-off compared to other state-of-the-art methods, without relying on any public data, as demonstrated on several tabular and image benchmark datasets. In addition, we also extend NTK to Data Distillation (DD) in federated learning (FL) settings, where we aim to condense sensitive information into a small set of images for deep learning training in a DP manner, we show that our method obtains meaningful results even under class imbalance and spuriously correlated image datasets.
Lay Summary

In this work, we address the growing concern for data privacy in the field of Deep Learning, where the risk of sensitive information exposure is high. Differential Privacy is considered the gold standard for privacy, and this study explores a specific method called Maximum mean discrepancy for creating privacy-preserving datasets. Here we investigate the use of empirical NTK features and discover that without reliance on a public dataset, e-NTK features surprisingly match the performance of features from pre-trained models. Thus, this approach enhances the privacy-accuracy balance without needing public data, as proven through experiments on various data types. Additionally, the research extends to federated learning, demonstrating that NTK can effectively condense sensitive data for deep learning, even in adversarial conditions. The findings are validated using image datasets, showing promising results for privacy-preserving data generation.
Preface

The method DP-NTK, introduced in Section 3.1, has been presented in the 5th AAAI Workshop on Privacy-Preserving Artificial Intelligence (Yang et al., 2024) and is currently in submission to a journal. DP-NTK has been done in collaboration with Professor Danica J. Sutherland and Kamil Adamczewski and under the supervision of Professor Mi Jung Park and Professor Xiaoxiao Li. The second method introduced in Section 3.2 is work done independently under the supervision of Professor Mi Jung Park and Professor Xiaoxiao Li.

For the first method, Professor Danica J. Sutherland greatly assisted in grounding the method with solid theory in Chapter 2 and Section 3.1.1. Frederik Harder and Margarita Vinaroz provided the initial code for their respective projects that this work is based on. Kamil Adamczewski helped run the hyperparameter searches for some experiments. Professor Mi Jung Park and Professor Xiaoxiao Li provided help in guiding the project direction and defining research objectives and helped write portions of the manuscript. I developed and tested the key ideas of the paper, did thorough testing of different implementation choices and ensured it was mathematically sound. I implemented most of the code and ran most of the experiments, as well as conducted the analysis of experimental results and wrote the manuscript.

For the second method, I formulated most of the key ideas of the paper, implemented most of the code and ran all of the experiments. Margarita Vinaroz provided the initial code for her project that the work is based on. I wrote the entire manuscript and Professor Mi Jung Park and Professor Xiaoxiao Li provided feedback on the project direction and defining the research objectives.
# Table of Contents

Abstract ......................................................... iii

Lay Summary ................................................... iv

Preface .......................................................... v

Table of Contents ............................................... vi

List of Tables ................................................... viii

List of Figures ................................................... ix

Acknowledgments ................................................ x

1 Introduction .................................................... 1

2 Related Work ................................................... 5
  2.1 Differential Privacy (DP) ................................. 5
  2.2 Neural Tangent Kernel (NTK) ............................ 6
  2.3 Maximum Mean Discrepancy (MMD) .................... 7
  2.4 Data Distillation (DD) in Federated Learning (FL) with DP .......................... 8
  2.5 Kernel Inducing Points (KIP) ........................... 9

3 Models .......................................................... 11
  3.1 DP-NTK .................................................... 11
    3.1.1 DP-NTK Theoretical analysis ..................... 13
List of Tables

Table 4.1  Accuracy under different widths, with $\left(10, 10^{-5}\right)$ DP .......................... 21
Table 4.2  Performance comparison on MNIST and F-MNIST dataset averaged over five independent runs, with NTK width fixed at 800. Here $\delta$ is fixed at $10^{-5}$. .................................................. 28
Table 4.3  Performance comparison on MNIST and F-MNIST dataset for our method with NTK width fixed at 800 and DP-MEPF (Harder et al., 2022). Here $\delta$ is fixed at $10^{-5}$. ............................... 28
Table 4.4  Performance comparison on Tabular datasets averaged over five independent runs. The top six datasets contain binary labels while the bottom two datasets contain multi-class labels. The metric for the binary datasets is ROC and PRC and for the multi-class datasets F1 score. .................................................. 29
Table 4.5  KRR accuracy table for MNIST, with different DP epsilon budgets and labels are completely split between 10 clients. .................. 29
Table 4.6  KRR accuracy table for MNIST, with same DP epsilon budgets and different alpha values for the Dirichlet distribution that defines the data skew for the 10 clients. .............................. 29
Table 4.7  KRR Accuracy Table for ColorMNIST with a balanced split between 10 clients. Here all $\delta$ is fixed at $10^{-5}$ ............................... 29
Table A.1  Best Hyperparamters used at different seeds for our experiments, see our repo for details. .................................................. 39
### List of Figures

| Figure 4.1 | Generated samples of MNIST and FashionMNIST from DP-NTK with different widths $w$; all samples use the same DP noise level ($\epsilon = 10$, $\delta = 10^{-5}$). | 22 |
| Figure 4.2 | Generated samples for MNIST and FashionMNIST from DP-NTK, with the same width ($w = 800$) and different DP levels. | 23 |
| Figure 4.3 | Generated samples for MNIST and FashionMNIST from DP-NTK and comparison models. | 23 |
| Figure 4.4 | Synthetic $32 \times 32$ CelebA samples generated at different levels of privacy. Samples for DP-MERF and DP-Sinkhorn are taken from (Cao et al., 2021). Our method yields samples of higher visual quality than the comparison methods. The FID for the proposed method is 75. FID for DP-Sinkhorn is 189. FID for DP-MERF is 274. | 24 |
| Figure 4.5 | The generated and real images for the CIFAR-10 dataset. The FID scores for the proposed method are 104 ($\epsilon = \infty$) and 107 ($\epsilon = 10$), respectively. For DP-MERF, they are 127 ($\epsilon = \infty$) and 141 ($\epsilon = 10$). | 24 |
| Figure 4.6 | Row 1: ColorMNIST original images, Row 2: FL-DP-KIP initial support set for ColorMNIST, Row 3: FL-DP-KIP final support set ($\epsilon = 10$), Row 4: MNIST original images, Row 5: FL-DP-KIP initial support set for MNIST, Row 6: FL-DP-KIP final support set for disjoint ($\epsilon = 1$). Here all $\delta$ is fixed at $10^{-5}$ | 27 |
Acknowledgments

The preliminary result of this work started in the summer between finishing my undergrad and starting my master’s. Thank you to my supervisors, Professor Mi Jung Park and Professor Xiaoxiao Li, for their guidance and support. They helped me a lot, not just with my studies but in growing as a person too. Thank you to Professor Danica J. Sutherland, for providing me with immense help during the submission and rebuttal of DP-NTK as well as agreeing to be on my committee.

As the last section I wrote, this acknowledgement has made me see that getting a degree is about more than just studying. It’s about all the great people I met along the way. So, thank you to everyone who was part of this journey with me. You made me realize that research has to come from a source of passion and love.

A huge thanks to my family. Mom and Dad and my little sister, you’ve always been there for me, supporting and believing in me. To my friends—Derren, Beidi, Wenlong, and Dong—thank you for being there for me. You’ve all been a big part of my happiness and kept me going. I’m grateful for having you in my life.

To Marga and Xenxo, thank you for your suggestions on the research directions and for being a good brother and sister when I was just beginning research. Without your guidance, I would’ve never been able to transition from my undergrad to my master’s quickly.

Most importantly, a huge thank you to my girlfriend, Christina. You’re like sunshine during Vancouver’s tough winter, always brightening my days. You inspire me to reach higher and give me strength. I hope to support you as much as you’ve supported me.

To everyone else who has been a part of my journey but I haven’t named, thank you. Your support means the world.
Chapter 1

Introduction

The initial work on Differentially Private (DP) data generation mainly focuses on a theoretical understanding of the utility of generated data, under strong constraints on the data type and its intended use (Hardt et al., 2012; Mohammed et al., 2011; Xiao et al., 2010; Zhu et al., 2017). However, recent advances in deep generative modelling provide an appealing, more general path towards private machine learning: if trained on synthetic data samples from a sufficiently faithful, differentially private generative model, any learning algorithm becomes automatically private.

One attempt towards this goal is to privatize standard deep generative models by adding appropriate noise to each step of their training process with DP-SGD (Abadi et al., 2016b). The majority of the work under this category is based on the Generative Adversarial Network (GAN) framework. In these approaches, the gradients of the GAN discriminator are privatized with DP-SGD (Chen et al., 2020; Torkzadehmahani et al., 2019; Xie et al., 2018; Yoon et al., 2019), which limits the size of the discriminator for a good privacy-accuracy trade-off of DP-SGD.

Another line of work builds off kernel mean embeddings (Berlinet and Thomas-Agnan, 2004, Chapter 4; Muandet et al., 2017), which summarizes a data distribution by taking the mean of its feature embedding into a reproducing kernel Hilbert space. The distance between these mean embeddings, known as the Maximum Mean Discrepancy (MMD) (Gretton et al., 2012), gives a distance between distributions; one way to train a generative model is to directly minimize (an estimate of) this distance (as in Dziugaite et al., 2015; Li et al., 2015).
By choosing a different kernel, the MMD “looks at” data distributions in different ways; for instance, the MMD with a short-lengthscale Gaussian kernel will focus on finely localized behaviour and mostly ignore long-range interactions, while the MMD with a linear kernel is based only on the difference between distributions’ means. Choosing an appropriate kernel, then, has a strong influence on the final behaviour of a generative model. In non-private models, there has been much exploration of adversarial optimization such as GANs (Goodfellow et al., 2014, and thousands of follow-ups), which corresponds to learning a kernel to use for the MMD (Arbel et al., 2018; Bińkowski et al., 2018; Li et al., 2017).

In private settings, however, fixing the kernel for a mean embedding has a compelling advantage: if we pick a fixed kernel with a finite-dimensional feature embedding, we can privatize the kernel mean embedding of a target dataset once, then repeatedly use the privatized embedding during generator training without incurring any further loss of privacy. This insight was first used in generative modeling in DP-MERF (Harder et al., 2021), which uses random Fourier features (Rahimi and Recht, 2007) for Gaussian kernels; Hermite polynomial features can provide a better trade-off in DP-HP (Vinaroz et al., 2022). Kernels based on perceptual features extracted from a deep network can also give far better results, if a related but public dataset is available to train that network, via DP-MEPF (Harder et al., 2022).

When no such public data is available, we would still like to use better kernels than Gaussians, which tend to have poor performance on complex datasets like natural images. In this work, we turn to a powerful class of kernels known as NTKs; (Arora et al., 2019; Chizat et al., 2019; Jacot et al., 2018; Lee et al., 2019). The “empirical NTK” (e-NTK) describes the local training behaviour of a particular neural network, giving a first-order understanding of “how the network sees data” (see, e.g., Ren et al., 2022, Proposition 1, or Chizat et al., 2019). As the width of the network grows (with appropriate parameterization), this first-order behaviour dominates and the e-NTK converges to a function that depends on the architecture but not the particular weights, often called “the NTK.” (Chapter 2 has more details.) Standard results show that in this infinite limit, training deep networks with stochastic gradient descent corresponds to kernel regression with the NTK. More relevantly for our purposes, common variants of GANs with infinitely-wide
discriminator networks in fact reduce exactly to minimizing the MMD based on an NTK (Franceschi et al., 2022), providing significant motivation for our choice of kernel.

In addition to their theoretical motivations, NTKs have proved to be powerful general-purpose kernels for small-data classification tasks (Arora et al., 2020), perhaps because typical neural network architectures provide “good-enough” prior biases toward useful types of functions for such problems. They have similarly been shown to work well for the problem of statistically testing whether the MMD between two distributions is nonzero (and hence the distributions are different) based on samples (Cheng and Xie, 2021; Jia et al., 2021), where it can be competitive with learning a problem-specific deep kernel (Liu et al., 2020; Sutherland et al., 2017). The task of generative modelling is intimately related to this latter problem, since we are aiming for a model indistinguishable from the target data (see further discussion by, e.g., Arbel et al., 2018).

In this work, we use mean embeddings based on e-NTKs as targets for our private generative model; we use e-NTKs rather than infinite NTKs for their finite-dimensional embeddings and computational efficiency. We show how to use these kernels within a private generative modeling framework, in (to the best of our knowledge) the first practical usage of NTK methods in privacy settings. Doing so yields a high-quality private generative model for several domains, outperforming models like DP-MERF and DP-HP as well as models based on DP-SGD.

In addition to image generation based on NTKs, a novel data creation called Data Distillation (DD) has become increasingly popular in recent years. First proposed by Wang et al. (2018), DD at its core aims to distill a small synthetic sample set that achieves similar performance for downstream tasks compared to the original dataset. Since its introduction, there have been various methods proposing similarity metrics that compare the distilled set and the original set. The gradient matching algorithm proposed by Zhao et al. (2020) defines this similarity metric as a gradient comparison of the neural network trained on the distilled data and the distilled dataset. Here, we focus on the Kernel Inducing Points (KIP) method proposed by Nguyen et al. (2020), as they instead formulate this metric as a kernel ridge regression problem between the two sets using the infinite width NTK. Although DD has been known to be empirically resilient against attacks aimed at the
original dataset, prior works such as Dong et al. (2022) attempt to directly connect DD with DP, claiming to achieve DP without any adversarial noise has failed as pointed out by Carlini et al. (2022). As DD is motivated by this need to preserve privacy for distilling sensitive data, Vinaroz and Park (2023) has shown that KIP is quite powerful in DP settings, having good performance even in very strict privacy budgets. In this work, we extend the work of Vinaroz and Park (2023) to the Federated Learning (FL) setting, as FL settings benefit immensely from knowledge condensation to reduce computation costs such as (Zhou et al., 2020). For this reason, there have been many works exploring this direction, including (Hu et al., 2022) which uses cross-entropy loss to find the best synthetic data and only needs to communicate the norm and the local synthetic data. However, as DP adds adversarial noise, how to add noise in FL that does not affect DD remains an open question, there have been attempts to solve this problem, such as FedLAP-DP (Wang et al., 2023), which uses local synthetic samples as loss-surrrogates and can achieve record-level DP. We call our method FL-DP-KIP and aim to show good privacy guarantees as well as good downstream performance.

The contributions of this work can be summarized as the following:

- Created a novel method called DP-NTK for Differentially Private Image Generation. We comprehensively test its performance in image and tabular datasets and compare it with similar models.

- Created a novel method called FL-DP-KIP that performs data-distillation tasks for differentially private federated learning systems. We consider the non-IID setting in FL and evaluate our models on datasets with class imbalance.

In Chapter 2, we will introduce the necessary background information, followed by a detailed explanation of our methods in Chapter 3, we then display our experimental results in Chapter 4. Finally, We will discuss our findings and future work in Chapter 5.
Chapter 2

Related Work

We begin by providing a brief background on DP, NTKs, MMD, DD in FL, and KIP.

2.1 Differential Privacy (DP)

A differentially private mechanism guarantees a limit on how much information can be revealed about any single individual’s participation in the dataset: formally, a mechanism \( \mathcal{M} \) is \((\varepsilon, \delta)\)-differentially private if for all datasets \( \mathcal{D}, \mathcal{D}' \) differing by a single entry and all sets of possible outputs \( S \) for the mechanism, \( \Pr[\mathcal{M}(\mathcal{D}) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(\mathcal{D}') \in S] + \delta \). The level of information leak is quantified by a chosen value of \( \varepsilon \) and \( \delta \); smaller values correspond to stronger guarantees.

In this work, we use the Gaussian mechanism to ensure the output of our algorithm is DP. Given a function \( h(\mathcal{D}) \) with outputs in \( \mathbb{R}^d \), we will enforce privacy by adding a level of noise based on the global sensitivity \( \Delta_h \) by Dwork et al. (2006). This is defined as the maximum \( L_2 \) norm by which \( h \) can differ for neighbouring datasets \( \mathcal{D} \) and \( \mathcal{D}' \), \( \sup ||h(\mathcal{D}) - h(\mathcal{D}')||_2 \). The mechanism’s output is denoted by \( \widetilde{h}(\mathcal{D}) = h(\mathcal{D}) + n \), where \( n \sim \mathcal{N}(0, \sigma^2 \Delta_h^2 \mathbf{1}_d) \). The perturbed function \( \widetilde{h}(\mathcal{D}) \) is \((\varepsilon, \delta)\)-DP, where \( \sigma \) is a function of \( \varepsilon \) and \( \delta \). While \( \sigma = \sqrt{2\log(1.25/\delta)}/\varepsilon \) is sufficient for \( \varepsilon \leq 1 \), numerically computing \( \sigma \) given \( \varepsilon \) and \( \delta \) using the auto-dp package of Wang et al. (2019) gives a smaller \( \sigma \) for the same privacy guarantee.
2.2 Neural Tangent Kernel (NTK)

The NTK was defined by Jacot et al. (2018), emerging from a line of previous work on optimization of neural networks. The empirical NTK, which we call e-NTK, arises from Taylor expanding the predictions of a deep network $f_{\theta}$, with parameters $\theta \in \mathbb{R}^d$, and is given by

$$\text{e-NTK}(x, x') = [\nabla_{\theta} f_{\theta}(x)]^\top [\nabla_{\theta} f_{\theta}(x')] .$$

The gradient here is with respect to all the parameters in the network, treated as a vector in $\mathbb{R}^d$. The e-NTK is a kernel function, with an explicit embedding $\phi(x) = \nabla_{\theta} f_{\theta}(x)$.

For networks whose output has multiple dimensions (e.g. a multi-class classifier), we use $f_{\theta}$ above to refer to the sum of those outputs; this “sum-of-logits” scheme approximates the matrix-valued e-NTK (Mohamadi et al., 2022b), up to a constant scaling factor which does not affect our usage. In PyTorch (Paszke et al., 2019), $\nabla_{\theta} f_{\theta}(x)$ can be easily computed by simply calling `autograd.grad` on the multi-output network, which sums over the output dimension if the network’s output is multi-dimensional.

As appropriately-initialized networks of a fixed depth become wider, the e-NTK (a) converges from initialization to a fixed kernel, often called “the NTK,” independent of $\theta$; and (b) remains essentially constant over the course of gradient descent optimization (Arora et al., 2019; Chizat et al., 2019; Jacot et al., 2018; Lee et al., 2019; Yang, 2019; Yang and Littwin, 2021). Algorithms are available to compute this limiting function exactly (Arora et al., 2019; Novak et al., 2020), but they no longer have an exact finite-dimensional embedding and can be significantly more computationally expensive to compute than e-NTKs for “reasonable-width” networks.

GANs with discriminators in an appropriate infinite limit theoretically become MMD minimizers using the NTK of the discriminator architecture (Franceschi et al., 2022). In concurrent work, Zhang et al. (2022) train generative models of a similar form directly, using the infinite NTKs; unfortunately, the particular model they use – which is, although they do not quite say so, the infinite-width limit of a least-squares GAN (Mao et al., 2017) – appears much more difficult to priva-
tize. The MMD with infinite NTK has also proved useful in statistically testing whether the MMD between two datasets is positive, i.e. whether the distributions differ (Cheng and Xie, 2021; Jia et al., 2021).

Other applications of the e-NTK include predicting the generalization ability of a network architecture or pre-trained network for fine-tuning (Bachmann et al., 2022; Malladi et al., 2022; Wei et al., 2022), studying the evolution of networks over time (Fort et al., 2020), and in approximating selection criteria active learning (Mohamadi et al., 2022a). The infinite NTK has seen more widespread adoption in a variety of application areas; the software page of Novak et al. (2020) maintains an extensive list.

2.3 Maximum Mean Discrepancy (MMD)

For any positive definite function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \rightarrow \mathcal{H}$ such that $k(x,y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ for all $x, y \in \mathcal{X}$: the kernel $k$ is the inner product between feature maps (Aronszajn, 1950). We can also “lift” this definition of a feature map to distributions, defining the (kernel) mean embedding of $P$ as

$$\mu_P = \mathbb{E}_{x \sim P}[\phi(x)] \in \mathcal{H};$$

(2.1)

it exists as long as $\mathbb{E}_{x \sim P}\sqrt{k(x,x)} < \infty$ (Smola et al., 2007). One definition of the MMD is (Gretton et al., 2012)

$$\text{MMD}(P, Q) = \|\mu_P - \mu_Q\|_{\mathcal{H}}.$$  
(2.2)

If $k$ is characteristic Sriperumbudur et al. (2011), then $P \mapsto \mu_P$ is injective, and so the MMD is a distance metric on probability measures. If $k$ is not characteristic, it still satisfies all the requirements of a distance except that we might have $\text{MMD}(P, Q) = 0$ for some $P \neq Q$.

Typically, we do not observe distributions $P$ and $Q$ directly, but rather estimate (2.2) based on i.i.d. samples $\{x_1, \ldots, x_m\} \sim P$ and $\{x'_1, \ldots, x'_n\} \sim Q$. Although other estimators exist, it is often convenient to use a simple “plug-in estimator” for the
mean embeddings:

\[ \hat{\mu}_P = \frac{1}{m} \sum_{i=1}^{m} \phi(x_i); \quad \text{MMD}(P, Q) = \| \hat{\mu}_P - \hat{\mu}_Q \|_H. \] (2.3)

For many kernels, an explicit finite-dimensional feature map \( \phi : \mathcal{X} \to \mathbb{R}^d \) is available; for a simple example, the linear kernel \( k(x, y) = x^T y \) has the feature map \( \phi(x) = x \) and mean embedding \( \mu_P = E_{x \sim P}[x] \). The MMD then becomes simply the Euclidean distance between mean embeddings in \( \mathbb{R}^d \). This setting is particularly amenable to differential privacy: we can apply the Gaussian mechanism to the estimate of the relevant mean embedding in \( \mathbb{R}^d \).

The commonly-used Gaussian kernel has an infinite-dimensional embedding; DP-MERF (Harder et al., 2021) therefore approximates it with a kernel corresponding to finite-dimensional random Fourier features (Rahimi and Recht, 2007), while DP-HP (Vinaroz et al., 2022) uses a different approximation based on truncating Hermite polynomial features (Mehler, 1866). DP-MEPF (Harder et al., 2022) chooses a different kernel, one which is defined in the first place from finite-dimensional features extracted from a deep network. Our e-NTK features are also finite-dimensional, with a particular choice of neural network defining the kernel.

### 2.4 Data Distillation (DD) in Federated Learning (FL) with DP

In general, the goal of FL is to collaborate with different actors to create a central model, without having to share each client’s private dataset. To preserve each data point for each client, which is called “record-level DP”, how to collaborate between clients while ensuring learning efficiency and privacy becomes a needed question. Under this setting, the de facto standard for federated learning is a method called FedAvg by Sun et al. (2021), the method sharing gradient information and Naïvely averages all the client gradients, as different clients have different setups, such as different hardware configurations, image size and resolution, client’s data may not satisfy the IID assumptions as highlighted by Kairouz et al. (2019), these challenges are analogous to distributed learning environments. To this end, DD has proved to be a novel lightweight approach for FL. As the goal for DD is to con-
dense a small dataset Although there has been much work on how to solve this problem without DP, such as Kwan and Song (2023) which uses ensembling to aggregate a small set of central models for knowledge distillation. Considering DP adds complexity to this distributed learning problem, however standard strategies such as limiting each epoch update to a certain range, such as FedLAP-DP (Wang et al., 2023), which limits each client update to a certain radius. In this work, we aim to use a novel approach to use NTK features to emulate a pre-trained NN in place of the client and central model, and we hope to show its expressiveness with more adversarial environments such as class imbalance and spuriously correlated data.

2.5 Kernel Inducing Points (KIP)

Current methods for a provable privacy guarantee utilize DP-SGD (Abadi et al., 2016a), a DP variation of SGD, to create DP variants of DD algorithms. Chen et al. (2022) utilized DP-SGD to privatize gradient matching, while the recently proposed DP-KIP (Vinaroz and Park, 2023) uses DP-SGD to privatize KIP. Here, we briefly introduce the KIP algorithm and how to privatize it. Following the specifications of Algorithm 1, the initial starting point is from a randomly initialized synthetic dataset denoted as $\mathcal{D}_s$, called the “support” set. In each iteration, KIP improves and encodes more information about the original dataset, called the “target” set and is denoted as $\mathcal{D}_t$, into this support set. Each iteration achieves this by minimizing the Kernel Ridge-Regression (KRR) loss:

$$
\mathcal{L}(\mathcal{D}_s) = \sum_{i=1}^{n} (y_i - \mathbf{k}_{i:s}^\top (K_{ss} + \lambda I)^{-1} \mathbf{y}_s)^2.
$$

(2.4)

As DP-SGD requires computing the per-sample gradients, thus for each sample in the target dataset batch, we can write the following:

$$
g(x_i, y_i) := \nabla_{\mathcal{D}_s} \mathcal{L}(\mathcal{D}_s) = \nabla_{\mathcal{D}_s} (y_i - \mathbf{k}_{i:s}^\top (K_{ss} + \lambda I)^{-1} \mathbf{y}_s)^2.
$$

(2.5)
Following Vinaroz and Park, 2023, Proposition 1, for the detailed proof that DP-KIP achieves \((\varepsilon, \delta)\) – DP for the distilled dataset. We have the following algorithm:

Algorithm 1: DP-KIP

\[
\begin{align*}
D_t & \leftarrow \text{Training dataset } \{ (x_{t,i}, y_{t,i}) \}_{i=1}^n \\
m & \leftarrow \text{Support set size} \\
n_{\text{iter}} & \leftarrow \text{Number of iterations} \\
B & \leftarrow \text{mini-batch size} \\
C & \leftarrow \text{Clipping norm} \\
\varepsilon, \delta & \leftarrow \text{Privacy Parameters} \\
\text{Initialization: } \text{Class-balanced Support set } D_s = \{ (x_{s,j}, y_{s,j}) \}_{j=1}^m \text{ where } x_{s,j} \sim \mathcal{N}(0, I_{D_s}) \\
\text{Compute the privacy parameter } \sigma \text{ using auto-dp package.} \\
\text{for } i = 1 \text{ to } n_{\text{iter}} \text{ do} \\
\quad \text{Randomly sample a mini-batch of the training dataset } D_B = \{ (X_{t_B}, y_{t_B}) \} \\
\quad \text{Compute KRR loss from Equation (2.4)} \\
\quad \text{for } b \text{ in } B \text{ do} \\
\quad\quad \text{Compute the per-sample gradients in Equation (2.5).} \\
\quad\quad \text{Gradient clipping } \hat{g}(x_b) = g(x_b) / \max(1, \|g(x_b)\|_2 / C) \\
\quad\quad \text{DP noise addition } \tilde{g} = \sum_{b=1}^B \hat{g}(x_b) + \mathcal{N}(0, \sigma^2 C^2 I) \\
\quad\quad \text{Update support (distilled) dataset } D_s \text{ with SGD} \\
\quad \text{end for} \\
\text{end for} \\
\text{return } \text{Differentially Private support dataset } D_s
\end{align*}
\]

In the next section, we will explain our method FL-DP-KIP, which extends DP-KIP to federated learning settings using FedAvg.
Chapter 3

Models

Here we present our method, which we call differentially private neural tangent kernels (DP-NTK) for privacy-preserving data generation. As well as our federated learning method federated learning differentially private kernel inducing points (FL-DP-KIP).

3.1 DP-NTK

We describe our method in the class-conditional setting; it is straightforward to translate to unconditional generation by, e.g., assigning all data points to the same class.

Following Harder et al. (2021, 2022); Vinaroz et al. (2022), we encode class information in the MMD objective by defining a kernel for the joint distribution over the input and label pairs in the following way. Specifically, we define

$$k((x,y), (x',y')) = k_x(x,x')k_y(y,y'),$$

where $k_y(y,y') = y^\top y'$ for one-hot encoded labels $y, y'$. For the kernel on the inputs, we use the “normalized” e-NTK of a network $f_\theta : \mathcal{X} \to \mathbb{R}$:

$$k_x(x,x') = \phi(x)\phi(x)$$

$$\phi(x) = \nabla_\theta f_\theta(x) / \|\nabla_\theta f_\theta(x)\|.$$
The normalization is necessary to bound the sensitivity of the mean embedding. Recall that if a network outputs vectors, e.g. logits for a multiclass classification problem, we use $f_\theta$ to denote the sum of those outputs (Mohamadi et al., 2022b).

Given a labelled dataset $\{(x_i, y_i)\}_{i=1}^m$, we represent the mean embedding of the data distribution as

$$\hat{\mu}_P = \frac{1}{m} \sum_{i=1}^m \phi(x_i)y_i^\top; \quad (3.1)$$

this is a $d \times c$ matrix, where $d$ is the dimensionality of the NTK features and $c$ the number of classes.\(^1\)

**Proposition 3.1.1.** The global sensitivity of the mean embedding (3.1) is $\Delta \mu_P = 2/m$.

**Proof.** Using the definition of the global sensitivity, which bounds for $D, D'$ which differ in only one entry, and the fact that $y$ is a one-hot vector and $\phi$ is normalized, we have that

$$\Delta \hat{\mu}_P = \sup_{D, D'} \left\| \frac{1}{m} \sum_{i=1}^m \phi(x_i)y_i^\top - \frac{1}{m} \sum_{j=1}^m \phi(x'_j)y'_j^\top \right\|_F$$

$$= \sup_{(x, y), (x', y')} \left\| \frac{1}{m} \phi(x)y^\top - \frac{1}{m} \phi(x')y'^\top \right\|_F$$

$$\leq \frac{2}{m} \sup_{(x, y)} \| \phi(x)y^\top \|_F$$

$$= \frac{2}{m} \sup_{x} \| \phi(x) \|$$

$$= \frac{2}{m}. \quad \square$$

We use the Gaussian mechanism to privatize the mean embedding of the data

\(^1\)This corresponds to a feature space of $d \times c$ matrices, where we use the Frobenius norm and inner product; though this is itself a Hilbert space, this exactly agrees with “flattening” these matrices into $\mathbb{R}^{dc}$ and then operating in that Euclidean space.
The privacy parameter $\sigma$ is a function of $\epsilon$ and $\delta$, which we numerically compute using the auto-dp package of Wang et al. (2019).

Our generator $G$ produces an input conditioned on a generated label $y'_i$, i.e. $G(z_i, y'_i) \mapsto x'_i$, where $y'_i$ is drawn from a uniform distribution over the $c$ classes and each entry of $z_i \in \mathbb{R}^d$ is drawn from a standard Gaussian distribution. Similar to (3.1), the mean embedding of the synthetic data distribution is given by

$$\hat{\mu}_Q = \frac{1}{n} \sum_{i=1}^{n} \phi(x'_i) y_i^T.$$  \hfill (3.2)

Our privatized MMD loss is given by

$$\text{MMD}^2(P, Q) = \|\tilde{\mu}_P - \hat{\mu}_Q\|_F^2,$$  \hfill (3.3)

where $F$ is the Frobenius norm. This loss is differentiable (as was proved in the NTK case by Franceschi et al. 2022; we compute the derivative with standard automatic differentiation systems). We minimize (3.3) with respect to the parameters of the generator $G$ with standard variations on stochastic gradient descent, where $\tilde{\mu}_P$ remains a constant vector but $\hat{\mu}_Q$ is computed based on a new batch of generator outputs at each step.

For class-imbalanced and heterogeneous datasets, such as the tabular datasets we consider, we use the modified mean embeddings of Harder et al. (2021, Sections 4.2-4.3), replacing their random Fourier features with our e-NTK features.

### 3.1.1 DP-NTK Theoretical analysis

Existing results on this class of models measure the quality of a generative model in terms of its squared MMD to the target distribution. How useful is that as a metric? The origin of the name “maximum mean discrepancy” is because we have
in general that

\[ \text{MMD}(P, Q) = \sup_{f: \|f\|_k \leq 1} \mathbb{E}_{x \sim P} f(x) - \mathbb{E}_{y \sim Q} f(y), \]

where \( \|f\|_k \) gives the norm of a function \( f : \mathcal{X} \to \mathbb{R} \) under the kernel \( k \). If the squared MMD is small, no function with small norm under the kernel \( k \) can strongly distinguish \( P \) and \( Q \). But when our kernel is the NTK of a wide neural network, the set of functions with small kernel norm is exactly the functions which can be learned by gradient descent in bounded time (also see more detailed discussion by Cheng and Xie 2021). Thus, if \( P \) is similar to \( Q \), (roughly) no neural network of the architecture used by our discriminator can distinguish the two distributions. Although our e-NTK kernel is finite-dimensional and hence not characteristic, this implies that distributions with small MMD under the NTK should model the distribution well for practical usages.

We can bound this MMD between the target distribution and the private generative model in two steps: the gap between the private and the non-private model, plus the gap between the non-private model and the truth.

For the extra loss induced by privacy, we can use the theoretical analysis of Harder et al. (2022), which is general enough to also apply to DP-NTK. Their main result shows the following in our setting.

**Proposition 3.1.2.** Fix a target distribution \( P \), and let \( \mathcal{Q} \) denote some class of probability distributions, e.g. all distributions which can be realized by setting the parameters of a generator network.

Let \( \tilde{Q} \in \arg\min_{Q \in \mathcal{Q}} \text{MMD}(P, Q) \) minimize the private loss (3.3), and let \( \hat{R} \in \arg\min_{Q \in \mathcal{Q}} \text{MMD}(P, Q) \) minimize the non-private loss (2.3).

Then the sub-optimality of \( \tilde{Q} \) relative to \( \hat{R} \) is given by\(^2\)

\[ \text{MMD}^2(P, \tilde{Q}) - \text{MMD}^2(P, \hat{R}) = O_p \left( \frac{\sigma^2 d}{m^2} + \frac{\sigma \sqrt{d}}{m} \text{MMD}(P, \hat{R}) \right). \]

The notation \( A = O_p(B) \) means roughly that with any constant probability, \( A =

\(^2\)See their Proposition A.5 for specific constants, and plug in \( \text{Tr}(\Sigma) = 4d\sigma^2/m^2 \), \( \|\Sigma\|_F = 4\sqrt{d}\sigma^2/m^2 \), \( \|\Sigma\|_{op} = 4\sigma^2/m^2 \).
Note that the MMD in the latter term is bounded by $\sqrt{2}$, so we can always achieve a $\mathcal{O}(1/m)$ rate for the squared MMD. If $\text{MMD}(P, \hat{R}) = 0$, though – the “interpolating” case – this “optimistic” rate shows that the private model is only $\mathcal{O}(1/m^2)$ worse than the non-private one, as measured on the particular sample.

Thus, we know that the private minimizer will approximately minimize the non-private loss. The sub-optimality in that minimization decays at a rate much faster than the known rates for the convergence of the non-private minimizer to the best possible model. The rates obtained by e.g. Dziugaite et al. (2015) have complex dependence on the generator architecture, but never give a rate for $\text{MMD}^2(P, \hat{R}) - \inf_{Q \in \mathcal{Q}} \text{MMD}^2(P, Q)$ faster than $\mathcal{O}(1/\sqrt{m})$; this is far slower than our $\mathcal{O}(1/m)$ or even $\mathcal{O}(1/m^2)$ rate for the extra loss due to privacy, so the private model is (asymptotically) not meaningfully worse than the non-private one.

### 3.1.2 DP-NTK implementation pseudocode

The training process of DP-NTK is split into two phases Algorithm 2, where we focus on creating the private data mean embedding with the added DP noise via the Gaussian Mechanism. Followed by Algorithm 3, the image generation step and can be completely separate from Step 1 and is done without any privacy concerns, as stated previously.
Algorithm 2: DP-NTK: Training data mean embedding

1: model ← Generative model
2: model\_NTK ← NTK model
3: Dataloader ← Dataloader for training set
4: \( d \) ← Length of flattened NTK features, i.e. size of NTK model parameters
5: \( n \) ← Size of training data obtained by count
6: \( n_c \) ← Class count
7: batch\_size ← Size of batch
8: \( n\_iter \) ← Number of iterations
9: mean\_emb1 ← \( d \times n_c \) mean embedding of training data (sensitive), initialize at 0
10: for Data, Label in Dataloader do
11: for each \( x \) in Data do
12: \( \phi(x) \) ← flattened NTK features for datapoint \( x \) calculated using model\_NTK calculated using autograd.grad
13: \( \phi(x) \leftarrow \frac{\phi(x)}{||\phi(x)||} \)
14: mean\_emb1[:,:Label[x]] += \( \phi(x) \)
15: end for each
16: end for
17: return noisy\_mean\_emb1 ← mean embedding of training data with added noise via the Gaussian mechanism
Algorithm 3: DP-NTK: Synthetic data generation

18: \textbf{for} \textit{i} in \textit{n}_\textit{iter} \textbf{do}
19: \hspace{1em} \textit{gen\_code}, \textit{gen\_label} \leftarrow \textit{code\_function}
20: \hspace{1em} \textit{gen\_data} \leftarrow \text{model(\textit{gen\_code})}
21: \hspace{1em} \textit{mean\_emb2} \leftarrow \textit{d \times n}_c \text{ mean embedding of generated data (non-sensitive)}
22: \hspace{1em} \textbf{for each} \textit{x} in \textit{gen\_data} \textbf{do}
23: \hspace{2em} \phi(x) \leftarrow \text{flattened NTK features for datapoint } x \text{ calculated using model\_NTK calculated using autograd.grad}
24: \hspace{2em} \phi(x) \leftarrow \frac{\phi(x)}{||\phi(x)||}
25: \hspace{2em} \text{mean\_emb2}[:,:\textit{gen\_label}[x]] += \phi(x)
26: \hspace{1em} \textbf{end for each}
27: \hspace{1em} \textit{loss} = \|\text{noisy\_mean\_emb1} - \text{mean\_emb2}\|_{2,1}^2
28: \hspace{1em} \text{backward using loss then step}
29: \hspace{1em} \text{per-iter evaluation}
30: \hspace{1em} \textbf{end for}
31: \textbf{return} final accuracy evaluation

3.2 FL-DP-KIP

Once again, we describe our method in the class-conditional setting; it is straightforward to translate to unconditional generation by, e.g., assigning all data points to the same class. As the distillation starting point cannot be sensitive client data, in each iteration \(m\), we perform DP-KIP on the selected random batch of client data and return the distilled images to the server to perform FedAvg. Here, FedAvg is performed by averaging the returned pictures from each client in the central server. As the returned images are already DP, we do not incur a privacy cost during communication. Each client can have the same amount of privacy budget without incurring additional privacy costs by scaling up the client count, which the formal proof will be provided in the formal definition below. This ensures the scalability of our model, which is a common bottleneck in FL settings. We repeat until a predefined amount of communication rounds. This simple method ensures that the client gradient information can be shared uniformly. We formally define
the FL-DP-KIP algorithm as follows:

Following Vinaroz and Park (2023), in the initial step the central server uses a randomly initialized synthetic dataset denoted as $\mathcal{D}_s = \{\hat{X}^0, y^0\}$, called the “support” set. The support set is balanced, meaning each class has the same amount of data points. This support set is then communicated to each client. During each communication, the client $k$ samples a random batch from its sensitive dataset $\mathcal{D}_t_m,k = \{(x^i_{k,j}, y^i_{k,j})\}_{i=1}^B$, this will be called the “Target” set for the current iteration $m$. With the support set and the target dataset, we can now perform the KRR loss described in Equation (2.4). To perform DP-SGD, we are required to individually compute each gradient, after which we clip and add controlled DP noise, finally for each sample, we update the support set directly by performing gradient decent. We return the updated support set back to the central server, and we use averaging on all client’s support sets as the global updated support set.

As mentioned, another important aspect of DP in FL is to inject the correct amount of DP noise. Thus to extend DP-KIP to FL settings, we can use the parallel composition theorem (McSherry, 2009):

**Theorem 3.2.1.** If there are $n$ mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_n$ computed on disjoint sub-sets whose privacy guarantees are $(\varepsilon_1, \delta_1), \ldots, (\varepsilon_n, \delta_n)$ respectively, then any function of $\mathcal{M}_1, \ldots, \mathcal{M}_n$ is $(\max_i \varepsilon_i, \max_i \delta_i)$–differential private.

As each client never communicates their local datasets, therefore the disjoint property is guaranteed. Thus the global Gaussian mechanism for FL-DP-KIP is $(\varepsilon, \delta)$–DP if each client is $(\varepsilon, \delta)$–DP. Hence, applying DP to $k$ clients incurs no further cost and no special procedure, FL-DP-KIP can be scaled up without any additional privacy cost. Once again, we use autograd.grad for DP noise calculation.
Algorithm 4: FL-DP-KIP: Server Execution
1: \( M \leftarrow \) number of communication rounds/iterations.
2: \( K \leftarrow \) amount of clients in the FL system.
3: \( d \leftarrow \) Length of flattened NTK features, i.e. size of NTK model parameters
4: initialize the support set \( \mathcal{D}_s = S_0 : \{\hat{X}^0, y^0\} \) where \( \hat{X}^0 \) is from Gaussian noise, \( y^0 \), size defined by hyperparameter
5: for \( m = 1, \ldots, M \) do
6:   for \( k = 1, \ldots, K \) do
7:     \( S_{m,k} \leftarrow \text{ClientExecute}(k, S_{m-1}) \)
8:   end for
9:  \( S_m \leftarrow \frac{1}{K} \sum_{k \in K} S_{m,k} \)
10: end for
11: return Distilled dataset \( S_m \).

Algorithm 5: FL-DP-KIP: Client \( k \) Execution (DP-KIP)
12: Uniformly sample random batch \( \mathcal{D}_{t,m,k} = \{(x_{k,i}^t, y_{k,i}^t)\}_{i=1}^B \) from client \( k \) dataset \( \mathcal{D}_k \)
13: compute KRR loss for \( \mathcal{D}_{t,m,k} = \{ (x_{k,i}^t, y_{k,i}^t) \}_{i=1}^B \) and \( S_{m-1} \)
14: for each \( (x_{k,i}^t, y_{k,i}^t) \) do
15:   \( g(x_{k,i}^t, y_{k,i}^t) = \nabla_{S_{m-1}} l_{\text{KRR}}(S_{m-1}, x_{k,i}^t, y_{k,i}^t) \)
16:   \( \bar{g}(x_{k,i}^t, y_{k,i}^t) = g(x_{k,i}^t, y_{k,i}^t) \cdot \min(1, \frac{C}{\|g(x_{k,i}^t)\|_2}) \)
17:   Update \( S_{m-1} \) using GD on \( \bar{g}(x_{k,i}^t, y_{k,i}^t) \)
18: end for
19: return Distilled dataset \( S_{m,k} \) for client \( k \) at time step \( m \).
Chapter 4

Experiments

In this section, we detail the experiments for our methods DP-NTK and FL-DP-KIP, we will first go through the setting, the datasets used and finally our experiments.

4.1 DP-NTK Experiments

We test our method, DP-NTK, on popular benchmark image datasets such as MNIST, FashionMNIST, CelebA, and CIFAR10, as well as on 8 benchmark tabular datasets. For MNIST/FMNIST we use a Conditional CNN as the generator. We use ResNet18 as our generator for Cifar10 and CelebA, and a fully connected network for tabular data: for homogenous data, we use 2 hidden layers + ReLU+ batch norm, or 3 hidden layers + ReLU + batch norm plus an additional sigmoid layer for the categorical features for heterogeneous data.

Our code is at https://github.com/FreddieNeverLeft/DP-NTK. The readme file of this repository and the section Appendix A.1 contain hyper-parameter settings (e.g., the architectural choices used for the model whose e-NTK we take) for reproducibility.

4.1.1 Generating MNIST and FashionMNIST images

For our MNIST and FashionMNIST experiments, we choose the e-NTK of a fully-connected 1-hidden-layer neural network architecture with a width parameter $w$, 

\[
\]
for simplicity. As the NTK features considered here are uniquely related to the width of the NTK network, for MNIST and F-MNIST data we conduct two groups of experiments, one where we test the accuracy of the model under the same Gaussian mechanism noise but with different NTK widths \( w \), and the other with the same width (\( w = 800 \)) but with varying privacy noise levels. In this setting, we will also compare results with other DP image-generating methods that do not rely on public data for pre-training, as well as DP-MEPF which uses public data to train a feature extractor model.

Following prior work such as Harder et al. (2021, 2022); Vinaroz et al. (2022), we quantitatively measure the quality of samples by training a classifier on the synthetic data and then measuring its accuracy when applied to real data. Here, we use two image classifiers: a multilayer perceptron (MLP) and logistic regression.

**Privacy-Width Trade-off** As discussed e.g. by Vinaroz et al. (2022), applying the Gaussian mechanism in feature spaces of higher dimension causes our model to become less accurate. This necessitates a trade-off in the feature dimensionality: large dimensions can lead to overwhelming amounts of added noise, while small dimensions may be inadequate to serve as a loss for image generation.

Figure 4.1 and Table 4.1 show that for both MNIST and FashionMNIST, changing the width does somewhat affect the final accuracy and image quality, but this effect is very minimal. Therefore, for subsequent experiments, we choose a width of 800 as a good compromise.

### Table 4.1: Accuracy under different widths, with \((10, 10^{-5})\) DP

<table>
<thead>
<tr>
<th></th>
<th>( w = 100 )</th>
<th>( w = 400 )</th>
<th>( w = 800 )</th>
<th>( w = 1000 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Log Reg</strong></td>
<td>0.8423</td>
<td>0.8404</td>
<td>0.84</td>
<td>0.8387</td>
</tr>
<tr>
<td><strong>MLP</strong></td>
<td>0.8824</td>
<td>0.8811</td>
<td>0.88</td>
<td>0.8805</td>
</tr>
<tr>
<td><strong>FashionMNIST</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Log Reg</strong></td>
<td>0.7640</td>
<td>0.7642</td>
<td>0.7663</td>
<td>0.7643</td>
</tr>
<tr>
<td><strong>MLP</strong></td>
<td>0.7773</td>
<td>0.7806</td>
<td>0.7838</td>
<td>0.7768</td>
</tr>
</tbody>
</table>
Figure 4.1: Generated samples of MNIST and FashionMNIST from DP-NTK with different widths \( w \); all samples use the same DP noise level \((\varepsilon = 10, \delta = 10^{-5})\).

Varying privacy levels  Table 4.2 and Figure 4.2 show our model’s performance under different levels of privacy. Other than for FashionMNIST with \( \varepsilon = 0.2 \), the performance of our model does not degrade significantly as the privacy requirement becomes more stringent.

Table 4.2 also compares with several existing models, while Figure 4.3 visually compares results with DP-GAN (Xie et al., 2018), DP-MERF (Harder et al., 2021) and DP-HP (Vinaroz et al., 2022). We see that even with simple architectures, DP-NTK broadly performs better than other high-accuracy models, and generates comprehensible images. Although there is still a distance from the current SOTA (Dockhorn et al., 2023), we show that DP-NTK still offers competitive performance under different privacy levels on MNIST and FashionMNIST. We also compare the performance of DP-MEPF (that relies on public data for pre-training a feature extractor model) and DP-NTK on MNIST and FashionMNIST datasets in Table 4.3.
Real Data

DP-NTK
\((\epsilon = 10, \delta = 10^{-5})\)

DP-NTK
\((\epsilon = 1, \delta = 10^{-5})\)

DP-NTK
\((\epsilon = 0.2, \delta = 10^{-5})\)

Figure 4.2: Generated samples for MNIST and FashionMNIST from DP-NTK, with the same width \((w = 800)\) and different DP levels.

Real Data

DP-GAN
\((\epsilon = 9.6)\)

DP-MERF
\((\epsilon = 1)\)

DP-HP
\((\epsilon = 1)\)

DP-NTK
\((\epsilon = 1)\)

Figure 4.3: Generated samples for MNIST and FashionMNIST from DP-NTK and comparison models.
Figure 4.4: Synthetic $32 \times 32$ CelebA samples generated at different levels of privacy. Samples for DP-MERF and DP-Sinkhorn are taken from (Cao et al., 2021). Our method yields samples of higher visual quality than the comparison methods. The FID for the proposed method is 75. FID for DP-Sinkhorn is 189. FID for DP-MERF is 274.

Figure 4.5: The generated and real images for the CIFAR-10 dataset. The FID scores for the proposed method are 104 ($\epsilon = \infty$) and 107 ($\epsilon = 10$), respectively. For DP-MERF, they are 127 ($\epsilon = \infty$) and 141 ($\epsilon = 10$).
4.1.2 Generating CelebA and CIFAR10 images

Image datasets beyond MNIST are notoriously challenging for differentially private data generation. Figure 4.4 on CelebA (at 32 $\times$ 32 resolution), we see that DP-MERF and DP-Sinkhorn give blurry, near-uniform samples, with DP-MERF also containing significant amounts of very obvious pixel-level noise. Our method, by contrast, generates sharper, more diverse images with some distinguishable facial features, although generation quality remains far behind that of non-private models.

Figure 4.5 shows results on CIFAR-10, where the difficulty lies in having relatively few samples for a wide range of image objects. Our samples, while again far from the quality of non-private generators, are less blurry than DP-MERF and contain shapes reminiscent of real-world objects.

4.1.3 Generating tabular data

For DP-NTK, we explore the performance of the algorithm on eight different imbalanced tabular datasets with both numerical and categorical input features. The numerical features on those tabular datasets can be either discrete (e.g. age in years) or continuous (e.g. height), and the categorical ones may be binary (e.g. drug vs placebo group) or multi-class (e.g. nationality).

For evaluation, we use ROC (area under the receiver characteristic curve) and PRC (area under the precision-recall curve) for datasets with binary labels, and F1 score for datasets with multi-class labels. Table 4.4 shows the average over the 12 classifiers trained on the generated samples (also averaged over 5 independent seeds). For most of the datasets, DP-NTK outperforms the benchmark methods.

4.2 FL-DP-KIP Experiments

We test our method, FL-DP-KIP, on popular benchmark image datasets such as MNIST and ColorMNIST, we consider two non-IID settings in FL for our experiments: class imbalance and spuriously correlated labels.

For MNIST, we consider the case of class imbalance, where each client has an imbalanced set of training data, here we consider two ways of splitting our MNIST data, one is following a Dirichlet distribution, where the parameter alpha controls
how skewed the distribution is: the smaller the value, the more disjoint the class distribution. We also consider the completely disjoint case, where each client only has access to 1 class each.

For ColorMNIST, we consider the dataset from (Nam et al., 2020). Here, the labels are spuriously correlated with the colours of the digits, therefore the challenge in this task in FL is to learn the underlying digit information rather than the colours that are spuriously correlated with the labels.

For both experiments, we showcase our model performance with 10 clients and display the performance of various privacy levels. We will report the KRR accuracy on the test set as our metric. Our code is at https://github.com/FreddieNeverLeft/FL-DP-KIP.git. The readme file of this repository contains some example scripts for reproducibility.

To our knowledge, this work is one of the first to consider the intersection between Differential Privacy, Federated Learning and Data Distillation in non-IID settings such as Class-imbalanced and spuriously correlated labels. We leave for future work more rigorous comparison studies and only report our preliminary results.

4.2.1 MNIST experiments
For the MNIST experiments, we report the performance of FL-DP-KIP in Table 4.6 and Table 4.5. A larger privacy budget should be less adverse for learning, our results could be due to the hyperparameter search being not thorough, as this learning problem is inherently difficult. However, fixing the $\varepsilon = 1$ and changing the Dirichlet distribution showed that our model performs well when the model is not as skewed. When compared with the centralized DP-KIP model by Vinaroz and Park (2023), we found that FL-DP-KIP achieves similar performance even with class imbalance. In Figure 4.6 we also showcase the visual examples of the distilled dataset, we can see that there is some resemblance to the original dataset, and is visually very similar to DP-KIP.
4.2.2 ColorMNIST experiments

For ColorMNIST experiments, we report the performance of FL-DP-KIP at various privacy levels. Here as we perform a balanced split between clients. From Figure 4.6 we can see that FL-DP-KIP can capture the underlying digit information and can achieve good KRR accuracy. We also noticed based on Table 4.7, DP noise didn’t affect the learning as much. However, as the KRR accuracy is lower than that of the MNIST settings, the causal relation with colour and label has significantly affected the final distilled data, leaving room for potential mitigation strategies exploration in the future.

Figure 4.6: Row 1: ColorMNIST original images, Row 2: FL-DP-KIP initial support set for ColorMNIST, Row 3: FL-DP-KIP final support set ($\epsilon = 10$), Row 4: MNIST original images, Row 5: FL-DP-KIP initial support set for MNIST, Row 6: FL-DP-KIP final support set for ($\alpha = 2, \epsilon = 1$), Row 7: FL-DP-KIP final support set for disjoint ($\epsilon = 1$). Here all $\delta$ is fixed at $10^{-5}$
### Table 4.2: Performance comparison on MNIST and F-MNIST dataset averaged over five independent runs, with NTK width fixed at 800. Here $\delta$ is fixed at $10^{-5}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>DP-$\epsilon$</th>
<th>MNIST</th>
<th>Fashion-MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Log Reg</td>
<td>MLP</td>
</tr>
<tr>
<td>DP-NTK (ours)</td>
<td>0.2</td>
<td>0.804</td>
<td>0.794</td>
</tr>
<tr>
<td>DPDM Dockhorn et al. (2023)</td>
<td>0.2</td>
<td><strong>0.81</strong></td>
<td><strong>0.817</strong></td>
</tr>
<tr>
<td>PEARL Liew et al. (2021)</td>
<td>0.2</td>
<td>0.762</td>
<td>0.771</td>
</tr>
<tr>
<td>DP-MERF Harder et al. (2021)</td>
<td>0.2</td>
<td>0.772</td>
<td>0.768</td>
</tr>
<tr>
<td>DP-NTK (ours)</td>
<td>1</td>
<td><strong>0.8324</strong></td>
<td><strong>0.8620</strong></td>
</tr>
<tr>
<td>DPDM Dockhorn et al. (2023)</td>
<td>1</td>
<td><strong>0.867</strong></td>
<td><strong>0.916</strong></td>
</tr>
<tr>
<td>PEARL Liew et al. (2021)</td>
<td>1</td>
<td>0.76</td>
<td>0.796</td>
</tr>
<tr>
<td>DP-HP Vinaroz et al. (2022)</td>
<td>1</td>
<td>0.807</td>
<td>0.801</td>
</tr>
<tr>
<td>DP-MERF Harder et al. (2021)</td>
<td>1</td>
<td>0.769</td>
<td>0.807</td>
</tr>
<tr>
<td>DP-NTK (ours)</td>
<td>10</td>
<td>0.8400</td>
<td>0.8800</td>
</tr>
<tr>
<td>DPDM Dockhorn et al. (2023)</td>
<td>10</td>
<td><strong>0.908</strong></td>
<td><strong>0.948</strong></td>
</tr>
<tr>
<td>PEARL Liew et al. (2021)</td>
<td>10</td>
<td>0.765</td>
<td>0.783</td>
</tr>
<tr>
<td>DP-Sinkhorn Cao et al. (2021)</td>
<td>10</td>
<td>0.828</td>
<td>0.827</td>
</tr>
<tr>
<td>DP-CGAN Torkzadehmahani et al. (2019)</td>
<td>10</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>DP-HP Vinaroz et al. (2022)</td>
<td>10</td>
<td>0.8082</td>
<td>0.8038</td>
</tr>
<tr>
<td>DP-MERF Harder et al. (2021)</td>
<td>10</td>
<td>0.794</td>
<td>0.783</td>
</tr>
<tr>
<td>GS-WGAN Chen et al. (2020)</td>
<td>10</td>
<td>0.79</td>
<td>0.79</td>
</tr>
</tbody>
</table>

### Table 4.3: Performance comparison on MNIST and F-MNIST dataset for our method with NTK width fixed at 800 and DP-MEPF (Harder et al., 2022). Here $\delta$ is fixed at $10^{-5}$.

<table>
<thead>
<tr>
<th>Method</th>
<th>DP-$\epsilon$</th>
<th>MNIST</th>
<th>Fashion-MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Log Reg</td>
<td>MLP</td>
</tr>
<tr>
<td>DP-NTK (ours)</td>
<td>0.2</td>
<td><strong>0.804</strong></td>
<td><strong>0.794</strong></td>
</tr>
<tr>
<td>DP-MEPF Harder et al. (2022)</td>
<td>0.2</td>
<td>0.80</td>
<td>0.76</td>
</tr>
<tr>
<td>DP-NTK (ours)</td>
<td>1</td>
<td><strong>0.8324</strong></td>
<td><strong>0.8620</strong></td>
</tr>
<tr>
<td>DP-MEPF Harder et al. (2022)</td>
<td>1</td>
<td>0.82</td>
<td><strong>0.89</strong></td>
</tr>
<tr>
<td>DP-NTK (ours)</td>
<td>10</td>
<td><strong>0.8400</strong></td>
<td><strong>0.8800</strong></td>
</tr>
<tr>
<td>DP-MEPF Harder et al. (2022)</td>
<td>10</td>
<td>0.83</td>
<td><strong>0.89</strong></td>
</tr>
</tbody>
</table>
Table 4.4: Performance comparison on Tabular datasets averaged over five independent runs. The top six datasets contain binary labels while the bottom two datasets contain multi-class labels. The metric for the binary datasets is ROC and PRC and for the multiclass datasets F1 score.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Real</th>
<th>DP-CGAN (1, 10^{-5})-DP</th>
<th>DP-GAN (1, 10^{-5})-DP</th>
<th>DP-MERF (1, 10^{-5})-DP</th>
<th>DP-HP (1, 10^{-5})-DP</th>
<th>DP-NTK (1, 10^{-5})-DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult</td>
<td>0.786</td>
<td>0.683</td>
<td>0.509</td>
<td>0.444</td>
<td>0.642</td>
<td>0.524</td>
</tr>
<tr>
<td>census</td>
<td>0.776</td>
<td>0.433</td>
<td>0.655</td>
<td>0.216</td>
<td>0.529</td>
<td>0.166</td>
</tr>
<tr>
<td>cervical</td>
<td>0.959</td>
<td>0.858</td>
<td>0.519</td>
<td>0.200</td>
<td>0.485</td>
<td>0.183</td>
</tr>
<tr>
<td>credit</td>
<td>0.924</td>
<td>0.864</td>
<td>0.664</td>
<td>0.356</td>
<td>0.435</td>
<td>0.150</td>
</tr>
<tr>
<td>epileptic</td>
<td>0.808</td>
<td>0.636</td>
<td>0.578</td>
<td>0.241</td>
<td>0.505</td>
<td>0.196</td>
</tr>
<tr>
<td>isolet</td>
<td>0.895</td>
<td>0.741</td>
<td>0.511</td>
<td>0.198</td>
<td>0.540</td>
<td>0.205</td>
</tr>
<tr>
<td>covtype</td>
<td>0.980</td>
<td>0.285</td>
<td>0.492</td>
<td>F1</td>
<td>0.467</td>
<td>F1</td>
</tr>
<tr>
<td>intrusion</td>
<td>0.971</td>
<td>0.302</td>
<td>0.251</td>
<td>F1</td>
<td>0.892</td>
<td>F1</td>
</tr>
</tbody>
</table>

Table 4.5: KRR accuracy table for MNIST, with different DP epsilon budgets and labels are completely split between 10 clients.

<table>
<thead>
<tr>
<th>ε</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>KRR Acc:</td>
<td>0.648</td>
<td>0.3213</td>
<td>0.703</td>
<td>0.534</td>
</tr>
</tbody>
</table>

Table 4.6: KRR accuracy table for MNIST, with same DP epsilon budgets and different alpha values for the Dirichlet distribution that defines the data skew for the 10 clients.

<table>
<thead>
<tr>
<th>ε = 1, α =</th>
<th>2</th>
<th>1</th>
<th>0.1</th>
<th>1e − 3 (complete split)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KRR Acc:</td>
<td>0.790</td>
<td>0.785</td>
<td>0.715</td>
<td>0.648</td>
</tr>
<tr>
<td>DP-KIP (Vinaroz and Park, 2023)</td>
<td></td>
<td></td>
<td></td>
<td>0.827</td>
</tr>
</tbody>
</table>

Table 4.7: KRR Accuracy Table for ColorMNIST with a balanced split between 10 clients. Here all δ is fixed at 10^{-5}

<table>
<thead>
<tr>
<th>ε =</th>
<th>0.2</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>KRR Acc:</td>
<td>0.2802</td>
<td>0.345</td>
<td>0.36839998</td>
</tr>
</tbody>
</table>
Chapter 5

Discussion

We introduced DP-NTK, a new model for differentially private generative modelling. NTK features are an excellent fit for this problem, providing a data-independent representation that performs well on a variety of tasks: DP-NTK greatly outperforms existing private generative models (other than those relying on the existence of similar public data for pre-training) for relatively simple image datasets and a majority of tabular datasets that we evaluated.

There are, however, certain avenues that could improve the DP-NTK model in future work. One path might be finding a way to use the infinite NTK, which tends to perform better on practical problems than e-NTKs at initialization (Arora et al., 2020).

When related public datasets are available for pretraining, it would also be helpful for DP-NTK to be able to incorporate that information, as (Harder et al., 2022) have argued is important. Indeed, by exploiting this information, their model outperforms ours on some image datasets. One strategy would be to take the e-NTK of a pre-trained model on that related dataset; our initial attempt at this did not help much, however, and we leave further investigation to future work.

We also introduce FL-DP-KIP, a novel application of KIP in differentially private federated learning settings. We show that NTK displays good performance even with the presence of data imbalance on simple datasets. In the future, we aim to explore the capabilities of FL-DP-KIP in larger datasets as well as conduct some theoretical analysis to gain more intuition on why KIP is a good fit for this task. Ad-
ditionally, there are a lot of potential directions to take as our experimental results are mostly preliminary. One direction is experimenting with replacing FedAVG with different FL methods that can mitigate data imbalance, such as SCAFFOLD (Karimireddy et al., 2021).


M. Arbel, D. J. Sutherland, M. Bińkowski, and A. Gretton. On gradient regularizers for MMD GANs. In NeurIPS, 2018. → pages 2, 3


H. M. Kwan and S. Song. Fedsdd: Scalable and diversity-enhanced distillation for model aggregation in federated learning, 2023. → page 9


F. G. Mehler. Ueber die entwicklung einer function von beliebig vielen variablen nach laplaceschen functionen höherer ordnung. Journal für die reine und angewandte Mathematik, 1866. → page 8


Y. Yang, K. Adamczewski, D. J. Sutherland, X. Li, and M. Park. Differentially private neural tangent kernels for privacy-preserving data generation, 2024. → page v

J. Yoon, J. Jordon, and M. van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *ICLR*, 2019. → page 1


## Appendix A

### Supporting Materials

#### A.1 Hyperparameters Used in DP-NTK Experiments

<table>
<thead>
<tr>
<th>dataset</th>
<th>iter</th>
<th>d_code</th>
<th>ntk width</th>
<th>batch</th>
<th>lr</th>
<th>eps</th>
<th>architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>dmnist</td>
<td>2000</td>
<td>5</td>
<td>800</td>
<td>5000</td>
<td>0.01</td>
<td>10, 0.2</td>
<td>fc_1l</td>
</tr>
<tr>
<td>fmnist</td>
<td>2000</td>
<td>5</td>
<td>800</td>
<td>5000</td>
<td>0.01</td>
<td>10, 0.2</td>
<td>fc_1l</td>
</tr>
<tr>
<td>celeba</td>
<td>20000</td>
<td>141</td>
<td>3000,200</td>
<td>1000</td>
<td>0.01</td>
<td>10</td>
<td>fc_2l</td>
</tr>
<tr>
<td>cifar10</td>
<td>40000</td>
<td>201</td>
<td>3000,200</td>
<td>1000</td>
<td>0.01</td>
<td>10, 1</td>
<td>fc_2l</td>
</tr>
<tr>
<td>cifar10</td>
<td>20000</td>
<td>21</td>
<td>800,1000</td>
<td>200</td>
<td>0.01</td>
<td>None</td>
<td>fc_2l</td>
</tr>
<tr>
<td>adult</td>
<td>50</td>
<td>11</td>
<td>30,200</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>cnn_2l</td>
</tr>
<tr>
<td>census</td>
<td>2000</td>
<td>21</td>
<td>30,20</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>cnn_2l</td>
</tr>
<tr>
<td>cervical</td>
<td>500</td>
<td>11</td>
<td>800,1000</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>cnn_2l</td>
</tr>
<tr>
<td>credit</td>
<td>500</td>
<td>11</td>
<td>1500</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>fc_1l</td>
</tr>
<tr>
<td>epileptic</td>
<td>2000</td>
<td>101</td>
<td>50,20</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>cnn_2l</td>
</tr>
<tr>
<td>isolet</td>
<td>1000</td>
<td>21</td>
<td>10,20</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>cnn_2l</td>
</tr>
<tr>
<td>covtype</td>
<td>1000</td>
<td>101</td>
<td>100,20</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>cnn_2l</td>
</tr>
<tr>
<td>intrusion</td>
<td>1000</td>
<td>21</td>
<td>30,1000</td>
<td>200</td>
<td>0.01</td>
<td>1</td>
<td>fc_2l</td>
</tr>
</tbody>
</table>

*Table A.1:* Best Hyperparameters used at different seeds for our experiments, see our repo for details.