A Neural Tangent Kernel Perspective on Function-Space Regularization in Neural Networks

Zonghao Chen
Tsinghua University

Xupeng Shi
Northeastern University

Tim G. J. Rudner
University of Oxford

Qixuan Feng
University of Oxford

Weizhong Zhang
HKUST

Tong Zhang
HKUST

Abstract

Loss regularization can help reduce the gap between training and test error by systematically limiting model complexity. Popular regularization techniques such as ℓ_2 weight regularization act directly on the network parameters, but do not explicitly take into account how the interplay between the parameters and the network architecture may affect the induced predictive functions. To address this shortcoming, we propose a simple technique for effective function-space regularization. Drawing on the result that fully-trained wide multi-layer perceptrons are equivalent to kernel regression under the Neural Tangent Kernel (NTK), we propose to approximate the norm of neural network functions by the *reproducing kernel Hilbert space* norm under the NTK and use it as a function-space regularizer. We prove that neural networks trained using this regularizer are arbitrarily close to kernel ridge regression solutions under the NTK. Furthermore, we provide a generalization error bound under the proposed regularizer and empirically demonstrate improved generalization and state-of-the-art performances on downstream tasks where effective regularization on the induced space of functions is essential.

1 Introduction

Over-parameterized deep neural networks have been shown to succeed at solving a wide array of complex tasks when a large number of training data is available. However, in areas where data is scarce or data labeling is expensive, neural networks may overfit and require effective regularization techniques to systematically control model complexity to improve generalization (Vapnik, 1998).

The empirical phenomenon of double decent in deep learning with over-parameterized neural networks (Belkin et al., 2018) have demonstrated that notions of regularization and generalization developed for smaller models may not transfer to highly expressive over-parameterized neural networks. Furthermore, recent work has shown that weight penalization is unable to effectively regularize model complexity in some model classes (Triki et al., 2018), that neural network parameters alone are a poor proxy for the predictive functions induced by the network architecture (Benjamin et al., 2018; Rudner et al., 2021, 2022), and that minimizing the parameter norm is not necessary to obtain good generalization (Zhang et al., 2021; Kawaguchi et al., 2017). This raises the question of why parameter space regularization techniques fall short and to what extent over-parameterized neural networks may benefit from alternative regularization techniques that do not explicitly regularize the network parameters but rather regularize a principled measure of network complexity.

In this paper, we propose to directly regularize the norm of function mappings induced by the neural network to more effectively control the complexity of the learned function and improve generalization in over-parameterized models. A natural candidate for the function norm is the *reproducing kernel Hilbert space* (RKHS) norm, but unfortunately there is no direct evidence that the function space

Preprint. Under review.

$\mathcal{C}$ defined by a neural network is identical to a RKHS $\mathcal{H}_K$. To address this difficulty, we draw on a result by Arora et al. (2019) that a fully-trained wide multi-layer perceptron is equivalent to a kernel regression predictor under the Neural Tangent Kernel (NTK). Building on this result, we prove that functions f encoded by a neural network can be approximated by functions $\tilde{f}$ in an RKHS $\mathcal{H}_K$ associated with the network's NTK with arbitrary precision. Since the RKHS norm $||\tilde{f}||_{\mathcal{H}_K}$ of $\tilde{f}$ is a principled measure of the function complexity, in Section 3 we propose to compute the norm of functions f in the same way and use the function norm as the function space regularizer. We further prove in Theorem 2 that fully-trained neural network functions under this regularizer are arbitrarily close to kernel ridge regression solutions under NTK and provide a generalization error bound in Theorem 3.

Several prior works have proposed to regularize neural networks directly in the space of functions. Benjamin et al. (2018) proposed to use the L_2 norm and Bietti et al. (2019) have proposed to use the Jacobian norm as a lower bound of RKHS norm. However, both methods fall short, as the L_2 space is a trivial function space which does not contain essential topological information in $\mathcal{C}$, and regularizing the lower bound has no direct relation to controlling the function complexity.

To evaluate whether the proposed function-space regularization technique leads to improved generalization, we follow prior work (Bietti et al., 2019) and train neural networks on MNIST and CIFAR-10—in each case with only a with small number of training samples—and show that the function-space regularizer improves predictive performance, compared to alternative methods. We further show that the proposed approach can be scaled to the MLP-Mixer architecture (Tolstikhin et al., 2021) which is prone to overfitting (Yu et al., 2022). Our function-space regularizer can effectively alleviate overfitting to achieve higher accuracy as well as log-likelihood. Lastly, we evaluate the proposed regularization technique on practically-relevant protein homology detection and continual learning tasks where effective regularization is essential. We show that our regularization method leads to state-of-the-art predictive accuracy in continual learning, outperforming related parameter- and function-space regularization techniques (Kirkpatrick et al., 2017; Nguyen et al., 2018; Titsias et al., 2019; Pan et al., 2020).

Contributions. We propose a regularization technique for neural network optimization that explicitly regularizes an approximation to the norm on functions in the RKHS associated with the network's NTK. We provide a theoretical justification for this approach and derive a corresponding generalization error bound. We empirically verify that the proposed regularization technique results in improved generalization performance and demonstrate that it results in state-of-the-art performance on to downstream prediction tasks where effective regularization is particularly important.

2 Related Work

Function-Space Regularization. Benjamin et al. (2018) used L_2 norm as a trivial function-space norm and Bietti et al. (2019) used Jacobian norm as a lower bound of the function norm under the assumption that the RKHS associated with deep convolutional kernels contains deep convolutional networks up to smooth approximations of rectified linear units Bietti et al. (2019).

The concept of function-space regularization has also been used in other areas, such as variational inference (Blei et al., 2017; Sun et al., 2018; Burt et al., 2020; Khan et al., 2019; Rudner et al., 2021) and continual learning (Titsias et al., 2019; Pan et al., 2020; Rudner et al., 2022). These function-space regularization techniques directly constrain the space of functions and as such allow for a more explicit incorporation of prior information about the functions to be learned.

For variational inference, function-space approaches avoid the limitations of parameter-space variational inference (Blundell et al., 2015; Farquhar et al., 2020) and achieve better uncertainty estimation and calibration. However, many function-space regularization approaches lack scalabaility. Sun et al. (2018) uses an expensive gradient estimator which limits the application to low-dimensional data and Pan et al. (2020) requires expensive matrix inversion operations.

For continual learning, most function-space approaches use variational inference. The regularization term is the Kullback-Leibler divergence from the variational posterior to the posterior of the previous task which is used as prior in the current task (Titsias et al., 2019). Benjamin et al. (2018) uses function L_2 norm as a function-space regularizer; FROMP (Pan et al., 2020) first constructs a GP based on DNN2GP (Khan et al., 2019) and then uses the GP posterior in KL divergence.

Generalization in Neural Networks. Classical learning theory attributes generalization ability to low-capacity class of hypotheses space (Vapnik, 1998; Mohri et al., 2012; Bartlett and Mendelson, 2002), but empirical findings about the performance of neural networks to generalize have contradicted those results, demonstrating that good generalization can be achieved despite, and due to, over-parameterization (Zhang et al., 2021; Kawaguchi et al., 2017). Keskar et al. (2016) argue that flat minima lead to better generalization, but Dinh et al. (2017) show that sharp minima can also generalize well. So far, it remains an open problem why over-parameterization in neural networks leads to improved generalization.

Kernel Methods. Kernel methods, especially support vector machines (Schölkopf et al., 2002), are a popular class of statistical learning methods. Classical choices of kernels include Gaussian, polynomial, and Matérn kernel functions. Previous works also considered families of Gaussian kernels (Micchelli et al., 2005) and hyperkernels (Ong et al., 2005). The generalization performance of kernel methods has been demonstrated theoretically (Lanckriet et al., 2004; Cortes et al., 2009). Despite the theoretical success of kernel methods, their high computational cost limits their applicability to many large-scale problems, and many approximation methods have been proposed to scale up kernel machines to large training datasets, such as random features (Rahimi et al., 2007) and the Nyström method (Williams and Seeger, 2000).

3 Preliminaries

In the following sections, we denote the training set as $\mathcal{D} \doteq \left\{\left(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\right)\right\}_{n=1}^{N} = (\mathbf{X}, \mathbf{y})$ with inputs $\mathbf{x}^{(n)} \in \mathcal{X} \subseteq \mathbb{R}^d$ and targets $\mathbf{y}^{(n)} \in \mathcal{Y}$. We use different subscripts under $\mathbf{X}$ to indicate different subset of training inputs. Note that $\mathbf{x} \in \mathcal{X}$ is also used for function inputs for convenience. We consider the function $f_{\boldsymbol{\theta}}(\cdot)$ encoded by a neural network with parameters $\boldsymbol{\theta} \in \mathbb{R}^P$ and denote the Jacobian of $f_{\boldsymbol{\theta}}(\cdot)$ with respect to $\boldsymbol{\theta}$ by $\mathcal{J}_{\boldsymbol{\theta}}(\cdot)$. ζ denotes a complexity-penalty coefficient.

3.1 Kernel Ridge Regression

Consider a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is symmetric and positive definite. According to Moore–Aronszajn theorem (Aronszajn, 1950), for any such kernel K on $\mathcal{X}$ we have a unique *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}_K$, which is the completion of the pre-Hilbert space consisting of all linear span of feature maps $\{K_{\mathbf{x}}(\cdot) : \mathbf{x} \in \mathcal{X}\}$, where $K_{\mathbf{x}}(\cdot) = K(\cdot, \mathbf{x})$. Classical kernel ridge regression consider the following optimization objective:

$$\mathcal{L}(f) = \frac{1}{N}\sum_{n=1}^{N} \ell(f(\mathbf{x}^{(n)})) + \zeta \, \|f\|_{\mathcal{H}_K}^2 . \tag{1}$$

with solution $f^* = \arg\min_{f \in \mathcal{H}_K} \mathcal{L}(f)$. The Representer Theorem (Schölkopf et al., 2001) implies that the solution f^* must be a finite linear combination of feature maps $K_{\mathbf{x}^{(1)}}, \cdots, K_{\mathbf{x}^{(N)}}$ evaluated on the training set, that is, for an arbitrary evaluation point $\mathbf{x}$

$$f^*(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n K(\mathbf{x}^{(n)}, \mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}. \tag{2}$$

This reduces the hypothesis function space to the finite linear span of feature maps evaluated on the training set. Hence the function norm can be calculated as follows:

$$\|f\|_{\mathcal{H}_K}^2 = \sum_{1 \le i,j \le N} \alpha_i \alpha_j K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \boldsymbol{\alpha}^\top K(\mathbf{X}, \mathbf{X})\boldsymbol{\alpha}, \tag{3}$$

where $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N]^\top$ represents the coordinate of f in $\mathcal{H}_K$ with respect to the basis $\Phi = [K_{\mathbf{x}^{(1)}}, \cdots, K_{\mathbf{x}^{(N)}}]^\top$. $K(\mathbf{X}, \mathbf{X}) = [K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})]_{1 \le i,j \le N}$ is the kernel matrix evaluated on training set, so it is symmetric and positive definite, i.e invertible. According to Eqn. (2) that $f(\mathbf{X}) = [f(\mathbf{x}^{(1)}), \cdots, f(\mathbf{x}^{N})]^\top$ and $f(\mathbf{X}) = K(\mathbf{X}, \mathbf{X})\boldsymbol{\alpha}$, we can then compute the RKHS norm as follows:

$$\|f\|_{\mathcal{H}_K}^2 = \boldsymbol{\alpha}^\top K(\mathbf{X}, \mathbf{X})\boldsymbol{\alpha} = f(\mathbf{X})^\top K(\mathbf{X}, \mathbf{X})^{-1} K(\mathbf{X}, \mathbf{X}) K(\mathbf{X}, \mathbf{X})^{-1} f(\mathbf{X}) = f(\mathbf{X})^\top K(\mathbf{X}, \mathbf{X})^{-1} f(\mathbf{X}). \tag{4}$$

Eqn. (4) shows that the RKHS norm is completely determined by function values and the inverse of kernel matrix. In Section 4, we show how to exploit this simple structure to compute a simple function-space regularizer for neural network optimization.

3.2 Kernel Methods and Deep Learning

We define an L-layer fully-connected neural network recursively as follows:

$$f^{(h)}(\mathbf{x}) = \mathbf{W}^{(h)} g^{(h-1)}(\mathbf{x}) \in \mathbb{R}^{d_h}, \quad g^{(h)}(\mathbf{x}) = \sqrt{\frac{c_\sigma}{d_h}} \sigma\big(f^{(h)}(\mathbf{x})\big), \quad \forall h = 1, 2, \cdots, L, \tag{5}$$

where we denote $g^{(0)}(\mathbf{x}) = \mathbf{x}$, $\mathbf{W}^{(h)} \in \mathbb{R}^{d_h \times d_{h-1}}$ is the weight matrix in the h-th layer, σ is the activation function and $c_\sigma^{-1} = \mathbb{E}_{z \sim \mathcal{N}(0,1)}[\sigma(z)^2]$ is a scaling factor. The resulting predictive function is given by

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = f^{(L+1)}(\mathbf{x}) = \mathbf{W}^{(L+1)} g^{(L)}(\mathbf{x}), \tag{6}$$

where $\boldsymbol{\theta} = (\mathbf{W}^{(1)}, \cdots, \mathbf{W}^{(L+1)})$ represents all parameters in the network. Jacot et al. (2018) defined the (empirical) Neural Tangent Kernel (NTK) as

$$\Theta(\mathbf{x}, \mathbf{x}') = \langle \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}), \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{x}') \rangle \tag{7}$$

and showed that, when the number of hidden units in the layers of a neural network goes to infinity, the empirical NTK Θ converges to a deterministic limiting kernel Θ_{ntk}, which we will refer to as the analytic NTK to distinguish it from the empirical NTK Θ. Complementing this result, Arora et al. (2019) showed that for ℓ_2 loss functions, the predictions of a fully-trained, *finite-width* full-connected ReLU neural network, f^*_{nn}, are arbitrarily close to the predictions under the (unregularized) analytic NTK Θ_{ntk} kernel regression solution, f^*_{ntk}:

Theorem 1 (Adapted from Arora et al. (2019))**.** *Let f be an m-layer fully-connected neural network mapping with ReLU activation functions, f^*_{nn} and f^*_{ntk} as defined above, $1/\kappa = \operatorname{poly}(1/\varepsilon, \log(N/\delta))$, $d_1 = d_2 = \cdots = d_L = m$ with $m \geq \operatorname{poly}(1/\kappa, L, 1/\lambda_0, n \log(1/\delta))$. Then for any $\mathbf{x}$ with $\|\mathbf{x}\| = 1$, with probability at least $1 - \delta$ over the random initialization,*

$$|f^*_{nn}(\mathbf{x}) - f^*_{ntk}(\mathbf{x})| \leq \varepsilon. \tag{8}$$

In the next section, we will extend this result to kernel ridge regression and use it to motivate a function-space regularization method for neural networks.

4 Function-Space Regularization in Neural Networks

While the function norm regularized in Eqn. (1) can in general not be computed for arbitrary neural network architectures, we consider a variation of the setting of Theorem 1 to motivate a function-space regularizer for arbitrary neural network architectures. In particular, we propose the following approximation as a general function-space regularizer for neural network optimization:

$$\mathcal{R}(f) = \|f\|^2_{\mathcal{H}_K} \approx f_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}) = \hat{\mathcal{R}}(f_{\boldsymbol{\theta}}). \tag{9}$$

To justify this regularizer, we return to the regularized optimization problem in Eqn. (1) and note that for an ℓ_2 loss function, the solution to the kernel ridge regression problem is given by

$$f^{\mathcal{R}*}_{ntk}(\cdot) = \Theta_{ntk}(\cdot, \mathbf{X}) \left(\Theta_{ntk}(\mathbf{X}, \mathbf{X}) + \zeta N \mathbf{I}\right)^{-1} \mathbf{y}. \tag{10}$$

If $\hat{\mathcal{R}}(f_{\boldsymbol{\theta}})$ is a good approximation to $\mathcal{R}(f)$, then the minimizer of the regularized objective

$$\mathcal{L}^{\hat{\mathcal{R}}}(f_{\boldsymbol{\theta}}) = \frac{1}{N} \sum_{n=1}^{N} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}^{(n)})) + \zeta f_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}), \tag{11}$$

denoted by $f^{\hat{\mathcal{R}}*}_{nn} = \arg\min_{f_{\boldsymbol{\theta}}} \mathcal{L}^{\hat{\mathcal{R}}}(f_{\boldsymbol{\theta}})$, should be close to $f^{\mathcal{R}*}_{ntk}$. The following result shows that this is indeed the case:

Theorem 2. *Let f be an m-layer fully-connected neural network mapping with ReLU activation functions, $f^{\hat{\mathcal{R}}*}_{nn}$ and $f^{\mathcal{R}*}_{ntk}$ as defined above, $1/\kappa = \operatorname{poly}(1/\varepsilon, \log(N/\delta))$, $d_1 = d_2 = \cdots = d_L = m$ with $m \geq \operatorname{poly}(1/\kappa, L, 1/(\lambda_0 + \zeta N))$. Then for any $\mathbf{x}$ with $\|\mathbf{x}\| = 1$, with probability at least $1 - \delta$ over random initialization,*

$$|f^{\hat{\mathcal{R}}*}_{nn}(\mathbf{x}) - f^{\mathcal{R}*}_{ntk}(\mathbf{x})| \leq \varepsilon. \tag{12}$$

Proof. See Appendix B. □

Algorithm 1 Neural Network Optimization with Function-Space Regularization

```
1: Initialize $\boldsymbol{\theta}$
2: for training epoch $m = 1, 2 \dots M$ do
3:     for each training iteration do
4:         Sample a mini batch of data $\mathcal{B} = \left\{ \left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \dots, \left(\mathbf{x}^{(B)}, \mathbf{y}^{(B)}\right) \right\}$
5:         Sample a sub batch of data $\mathcal{C} \subset \mathcal{B}$
6:         Compute NTK $\Theta(\mathbf{X}_\mathcal{C}, \mathbf{X}_\mathcal{C}) = \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C})\mathcal{J}_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C})^\top$
7:         Compute Loss $\mathcal{L}_\mathcal{B}^{\hat{\mathcal{R}}} = -\sum_{i=1}^{B} \log p(\mathbf{y}^{(i)} | f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) + \zeta f_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C})^\top \Theta^{-1}(\mathbf{X}_\mathcal{C}, \mathbf{X}_\mathcal{C}) f_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C})$
8:         Update $\boldsymbol{\theta}$: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_\mathcal{B}$
9:     end for
10: end for
output $\boldsymbol{\theta}$
```

Furthermore, we can obtain the following generalization error bound for the solution $f_{nn}^{\hat{\mathcal{R}}*}$:

Theorem 3. *Let $f_{nn}^{\hat{\mathcal{R}}*}$ be as defined above. Let $R(f_{nn}^{\hat{\mathcal{R}}*}) = \mathbb{E}[(f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) - \mathbf{y}(\mathbf{x}))^2]$ be the generalization error and $\hat{R}(f_{nn}^{\hat{\mathcal{R}}*})$ the corresponding empirical error. Assume for all $\|\mathbf{x}\| \leq 1$, $|f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x})| \leq C_0\kappa$ and $\left\|\partial_{\boldsymbol{\theta}} f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x})\right\| \leq C_1$ and $|\mathbf{y}| \leq c$. Then for any $\delta > 0$, with probability at least $1 - \delta$, we have the following bound:*

$$R(f_{nn}^{\hat{\mathcal{R}}*}) - \hat{R}(f_{nn}^{\hat{\mathcal{R}}*}) \leq 4(C_0\kappa + c)\varepsilon + \frac{8C_1^2\Lambda^2}{\sqrt{N}} \left(\sqrt{\frac{C_K}{NC_1^2}} + \frac{3}{4}\sqrt{\frac{\log \frac{2}{\delta}}{2}} \right), \tag{13}$$

where N is the number of training points and Λ is a constant that only depends on the regularization coefficient ζ. C_K is the trace of the empirical NTK. ε is the discrepancy between predictive functions from Theorem 2.

Proof. See Appendix B. □

Theorems 2 and 3 demonstrate that for fully-connected ReLU neural networks and ℓ_2 loss functions, the proposed function-space regularization method yields solutions that are close to the kernel ridge regression solution and generalize well. As these results do not apply to more general loss functions (such as cross-entropy loss functions) and neural network architectures (such as convolutional layers or attention mechanisms), we provide an extensive empirical evaluation in which we use the proposed function-space regularization technique in classification problems with convolutional neural network architectures and cross-entropy loss functions.

4.1 Computational Cost and Further Approximations

In this section, we discuss the computational cost of computing the function-space regularizer in Eqn. (9) and how to reduce it with suitable approximations.

The first challenge is the computation of the analytic NTK Θ_{ntk}. Although Arora et al. (2019) have derived an analytical expression of the analytic NTK for both fully-connected and convolutional networks, the layer-wise computation is too expensive to perform for large neural networks and at every gradient step. Fortunately, as Jacot et al. (2018) and Arora et al. (2019) showed that the Frobenius norm of the matrix difference between Θ and Θ_{ntk} is arbitrarily small for sufficiently wide neural networks, we approximate the analytic NTK of large (finite-width) neural networks by the empirical NTK as in Eqn.(7)[1].

The second challenge is the computation of the empirical NTK Θ itself. Computing the NTK $\Theta(\mathbf{X}, \mathbf{X})$ naively has $\mathcal{O}(P^2)$ space complexity and $\mathcal{O}(N^2P)$ time complexity. To reduce both time and space complexity, we use an implicit NTK computation included in the JAX-based `neural-tangents` python library (Novak et al., 2020).

The third challenge is that computing the function norm requires inverting the NTK $\Theta(\mathbf{X}, \mathbf{X})$. To reduce the computational cost, instead of inverting the entire N-by-N matrix $\Theta(\mathbf{X}, \mathbf{X})$, we sample

[1]We provide a detailed discussion of the empirical and analytic NTKs in Appendix B.

a mini-batch $\mathbf{X}_\mathcal{C}$ from $\mathbf{X}$ at every gradient step and compute the function norm on this particular mini-batch only. This way, we only need to invert an $|\mathbf{X}_\mathcal{C}|$-by-$|\mathbf{X}_\mathcal{C}|$ matrix, which, for a sufficiently small mini-batch size, can be done at every gradient step without a significant slowdown in training. We provide an empirical investigation into the accuracy of this approximation in Section 5.5.

A complete description of the proposed function-space regularization procedure is shown in algorithm 1, and the objective is given by

$$\mathcal{L}^{\hat{\mathcal{R}}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)} | f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) + \zeta f_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C})^\top \Theta^{-1}(\mathbf{X}_\mathcal{C}, \mathbf{X}_\mathcal{C}) f_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C}) \quad \text{with} \quad \mathbf{X}_\mathcal{C} \sim p_\mathbf{X} = \mathcal{U}(\mathbf{X}). \tag{14}$$

4.2 Sequential Optimization: Continual Learning

Continual learning is a setting in which a model seeks to represent S distinct datasets, arriving sequentially one at a time, so that a model needs to memorize previously learned predictions when learning to solve a new task (Kirkpatrick et al., 2017). We denote S sequential datasets by $\mathcal{D}_s = \{\mathbf{x}_s^{(n)}, \mathbf{y}_s^{(n)}\}_{n=1}^{N_s}$, $s = 0, 1, \ldots, S-1$. When learning task to solve any given task, a model has no access to the full datasets associated with previous tasks but only to a small number of samples from them. These subsets are referred to as "coresets" and denoted by C_s. We denote the parameters and the corresponding induced functions at task s by $\boldsymbol{\theta}_s$ and $f_{\boldsymbol{\theta}_s}(\cdot)$, respectively. We also denote the empirical NTK obtained at task s by Θ_s.

At task $s = 0$, we use maximum likelihood estimation without any regularization to fit the model parameters. When learning tasks $s > 0$, in order to avoid catastrophic forgetting of the previous task, we penalize the changes of the function encoded by the neural network using the proposed regularization technique described in the previous section. Specifically, we propose to regularize the distance between the function learned at task $s-1$ and the function to be learned at task s. In Benjamin et al. (2018), this distance is trivially measured by the L_2 norm as $\|f_s - f_{s-1}\|_{L_2}$. Based on the discussion of the function norm associated with a network's NTK in Section 3, we diverge from Benjamin et al. (2018) and instead compute the norm of $f_s - f_{s-1}$ the same way as in Eqn. (11) except for using the empirical NTK Θ_{s-1} of task $s-1$.

In the continual learning setting, we do not need to explicitly sample $\mathbf{X}_\mathcal{C}$ from $\mathbf{X}$ as we can directly evaluate function values on coresets C_s, which are typically small. The objective for continual learning at task s can then be expressed as

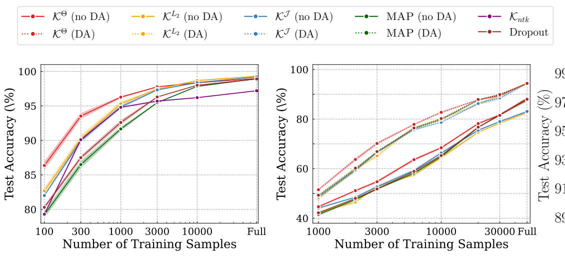
$$\mathcal{L}_s^{\hat{\mathcal{R}}}(\boldsymbol{\theta}_s) = \sum_{n=1}^{N_s} \log p(\mathbf{y}^{(n)} | f_{\boldsymbol{\theta}_s}(\mathbf{x}^{(n)})) + \zeta \Delta f(C_s)^\top \Theta_{s-1}^{-1}(C_s, C_s) \Delta f(C_s), \tag{15}$$

where $\Delta f(C_s) = f_{\boldsymbol{\theta}_s}(C_s) - f_{\boldsymbol{\theta}_{s-1}}(C_s)$ denotes the relative change of function values evaluated on coreset C_s from task $s-1$ to task s. Intuitively, the objective trades off fitting the data at task s while also maintaining "predictive memory" from previous tasks. This approach is memory efficient as it only needs to keep a working memory of function values at coresets and the NTK matrix.

5 Experiments

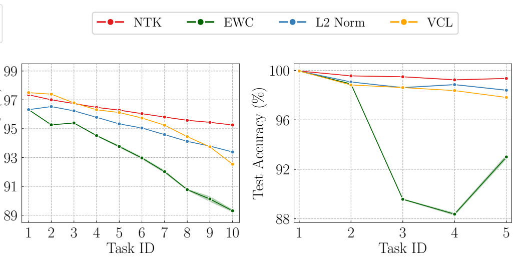
In Sections 3 and 4, we proposed a function-space regularization technique for neural network optimization In this section, we evaluate the empirical performance of the function-space regularization method proposed in Sections 3 and 4. To do so, we consider four different prediction tasks in which regularization is particularly important.

First, we follow the experiment setup in Bietti et al. (2019) to evaluate the generalization performance of our approach on MNIST and CIFAR-10 by training a neural network with only a small subset of samples from each of these datasets. This way, regularization becomes crucial in preventing overfitting on the training data. Second, the MLP-Mixer (Tolstikhin et al., 2021) model has recently drawn much attention as it has demonstrated competitive performance on large-scale datasets with much higher throughput compared to convolutional networks or self-attention models. However, the application of the MLP-Mixer model is limited by the fact that is prone to overfitting (Tolstikhin et al., 2021). We use the proposed function-space regularization method to train the MLP-Mixer model and show that it alleviates overfitting and leads to improved predictive performance. Third, we evaluate



(a) Accuracy on MNIST **(b)** Accuracy on CIFAR10

Figure 1: Test accuracy on MNIST and CIFAR-10.



(a) Permuted MNIST **(b)** Split MNIST

Figure 2: Continual learning experiments.

our method on the small-sample downstream task of protein homology detection. Fourth, based on the discussion in Section 4.2, we evaluate our method on continual learning tasks where effective regularization is needed to prevent catastrophic forgetting (Kirkpatrick et al., 2017; Nguyen et al., 2018; Benjamin et al., 2018; Titsias et al., 2019; Pan et al., 2020). For a comprehensive description of the datasets as well as training and evaluation protocols, see Appendix D. For a discussion of computational time complexity, see Appendix C.1.

We note that few-shot learning (FSL) problems, which tackle the problem of generalizing from a few examples, were not included in the empirical evaluation, since prior work (Wang et al., 2020) has demonstrated that FSL requires specialized techniques such as extracting prior knowledge from other related tasks. In contrast, this work proposes a general-purpose regularization technique and we performed experiments that best illustrate the usefulness of this technique on problems that require good generalization and can be solved without specialized techniques.

Notation. In the remainder of this section, we refer to our method as $\mathcal{K}^{\Theta}$, the function L_2 norm regularization approach proposed by Benjamin et al. (2018) as $\mathcal{K}^{L_2}$, and the Jacobian norm regularzation technique proposed by Bietti et al. (2019) as $\mathcal{K}^{\mathcal{J}}$. We refer to weight decay as maximum a posteriori (MAP) estimation) and to kernel regression under the NTK as $\mathcal{K}_{ntk}$.

5.1 Generalization From Small Datasets

Following Bietti et al. (2019), we train with a small number of samples on MNIST and CIFAR-10 to demonstrate generalization performance. We gradually increase the number of training samples to show that our method does not lead to underfitting even when the number of samples is large. We use a LeNet-style (Lecun et al., 1998) network for MNIST and ResNet18 (He et al., 2016) for CIFAR-10. In order to deal with the fact that *batch normalization* (BN) and therefore NTK computation is dependent on the evaluation set, we took special care to implement BN by computing the normalization mean and standard deviation for any single evaluation point deterministically at every iteration from a fixed "conditioning set" $\mathbf{X}_{\text{cond}}$.[2]

We compare our method with MAP, dropout (Srivastava et al., 2014), $\mathcal{K}^{L_2}$ and $\mathcal{K}^{\mathcal{J}}$. The comparison against $\mathcal{K}_{ntk}$ is only implemented on MNIST because the kernel regression under Θ_{ntk} is too expensive on ResNet18 even with Nyström approximation. Actually, there are many methods that improve generalization by regularizing implicitly, such as batch normalization and SGD (Bjorck et al., 2018; Zhou et al., 2020), and we are not comparing against these methods mainly because these implicit methods can be easily combined with explicit regularization techniques like ours, and we are already using them throughout our experiments.

Figures 1a and 1b show the test accuracy on MNIST and CIFAR-10 as we increase the number of training samples. On small datasets like MNIST or CIFAR-10, our networks can easily reach 100% accuracy on training set, therefore higher test accuracy indicates better generalization performance. We can see that our approach $\mathcal{K}^{\Theta}$ has the best generalization performance when the number of training samples are small, but MAP gradually catches up as the number of training samples grows. Moreover, our method also outperforms $\mathcal{K}_{ntk}$ due to the expressiveness of neural networks. However, as the number of training samples grow, $\mathcal{K}^{\Theta}$ loses its advantage because the large number of training samples makes the generalization ability less important. We also note that in Figure 1b, our approach $\mathcal{K}^{\Theta}$ is more effective when there is no data augmentation, because data augmentation is also a strong regularization technique that offsets the regularization effect of $\mathcal{K}^{\Theta}$.

[2]For further details, see Appendix D.

Table 1: MLP-Mixer performance on CIFAR-10 and CIFAR-100. The reported mean and standard error are computed over ten random seeds. [1]Maximum likelihood estimation. [2]Negative log likelihood.

Method	CIFAR-10		CIFAR-100	
	NLL[2]	Acc (%)	NLL	Acc (%)
MLE[1]	$0.10_{\pm 0.01}$	$96.5_{\pm 0.1}$	$0.56_{\pm 0.01}$	$84.5_{\pm 0.1}$
Ours	$0.10_{\pm 0.00}$	$\mathbf{97.0}_{\pm 0.1}$	$\mathbf{0.54}_{\pm 0.00}$	$\mathbf{85.2}_{\pm 0.1}$

Table 2: Regularization on protein homology detection tasks with data augmentation Bietti et al. (2019). Average auROC50 score is reported with standard error across five seeds. [1]Values obtained from Bietti et al. (2019).

Method	auROC50
MAP	$0.55_{\pm 0.02}$
$\mathcal{K}^{\mathcal{J}}$	$0.59_{\pm 0.09}$
$\mathcal{K}^{L_2}$	$0.58_{\pm 0.04}$
Dropout	$0.55_{\pm 0.02}$
$\|f\|_{\delta}^{2}$ [1]	0.61
grad-ℓ_2 [1]	0.57
Ours	$\mathbf{0.62}_{\pm 0.05}$

Table 3: Comparisons of predictive average accuracy computed across all S tasks against a selection of popular continual learning methods. Both the mean and standard error are computed across ten different random seeds.

Method	Permuted MNIST	Split MNIST
EWC	$84.0\%_{\pm 0.3}$	$63.1\%_{\pm 0.2}$
SI	86.0%	98.9%
VCL	$92.5\%_{\pm 0.1}$	$97.8\%_{\pm 0.0}$
FROMP	$94.9\%_{\pm 0.0}$	$99.0\%_{\pm 0.0}$
FRCL	$94.3\%_{\pm 0.0}$	$97.8\%_{\pm 0.2}$
L_2	$93.5\%_{\pm 0.0}$	$98.4\%_{\pm 0.1}$
Ours	$\mathbf{95.2\%}_{\pm 0.0}$	$\mathbf{99.3\%}_{\pm 0.0}$

5.2 Reducing Overfitting in Highly Overparameterized Models: MLP-Mixer

Based on the pratical techniques discussed in Section 4.1, we are able to train an MLP-Mixer of the B/16 architecture on CIFAR-10 and CIFAR-100 using the proposed function-space regularization technique. To train the model, we initialize the MLP-Mixer B/16 parameters with a set of parameters pretrained on ImageNet (Deng et al., 2009) and then train until convergence using the proposed function-space regularization method.

In this experiments, we observe that function-space regularization results in a higher level of predictive accuracy than maximum likelihood training from a pretrained model without function-space regularization (see Table 1). The improvement in predictive accuracy is statistically significant even for CIFAR-10 (for which the predictive accuracies of both are close to 100%), demonstrating that the proposed regularization technique is effective in reducing overfitting for MLP-Mixer models.

5.3 Improving Generalization Across Datasets: Protein Homology Detection

We consider the task of detecting a protein's superfamily based on its sequence, which is an important task in the area of bioinformatics. We used the *Structural Classification Of Proteins* (SCOP) version 1.67 dataset (Murzin et al., 1995). Applying the preprocessing steps described in Appendix D.4, we get 100 balanced binary classification tasks with 200 protein sequences each. We also cut the length of protein sequence to $l = 400$ amino acids and each amino acid is represented as $\{0, 1\}^{20}$ using a one-hot encoding. This way, each protein sequence becomes a $20 \times l$ matrix, which can be easily processed by convolutional neural networks Alipanahi et al. (2015). In our experiment, we use a simple CNN consisting of 3 convolutional layers followed by a fully-connected layer.

The performance from this experiment is reported in Table 2. The auROC50 score (area under the ROC curve up to 50% of false positives) is used as the main evaluation metric. We use the first 50 datasets as validation to select the hyperparameter, and report the auROC50 score over the remaining 50 datasets. As shown in Table 2 the proposed regularization method, $\mathcal{K}^{\Theta}$, performs best.

5.4 Preventing Catastrophic Forgetting in Continual Learning

We evaluate our approach discussed in Section 4.2 on single-head permuted-MNIST and multi-head split-MNIST, which are standard continual learning benchmarks. Following prior works, we use two-layer fully-connected neural network with 100 hidden units per layer for permuted MNIST and

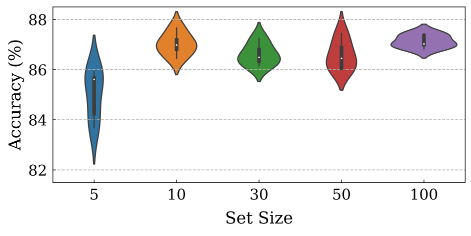


Figure 4: Effect of $|\mathbf{X}_\mathcal{C}|$ on predictive performance when training on MNIST

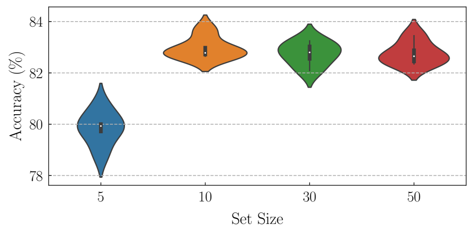


Figure 5: Effect of $|\mathbf{X}_\mathcal{C}|$ on predictive performance when training on CIFAR-10

two-layer fully-connected neural network with 256 hidden units per layer for split-MNIST. We use 200 coreset points for permuted-MNIST and 40 coreset points for split-MNIST.

We can see from Figures 2a and 2b and Table 3 that function-space methods provide higher average accuracy across all tasks than parameter-space methods, which shows that function space regularization results in better memorization of past knowledge. Among all function space approaches, our method $\mathcal{K}^\Theta$ provides the highest average accuracy. We attribute our performance to effective regularization on the relative function changes.

5.5 Ablation Study: Mini-Batch Approximation of the Function Norm

In order to validate our practical approximations used in Section 4, we provide some ablation studies here. We show how good the approximation is to use the empirical NTK Θ to approximate the analytic NTK Θ_{ntk} and also to use mini-batches to compute the norm. Specially, we show in Figure 3 with red line representing $f_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta_{ntk}^{-1}(\mathbf{X}, \mathbf{X}) f_{\boldsymbol{\theta}}(\mathbf{X})$ and the blue line representing

$$\frac{|\mathbf{X}|}{|\mathbf{X}_\mathcal{C}|} \mathop{\mathbb{E}}_{\mathbf{X}_\mathcal{C} \sim p_\mathbf{X}} \left[f_{\boldsymbol{\theta}}\left(\mathbf{X}_\mathcal{C}\right)^\top \Theta^{-1}\left(\mathbf{X}_\mathcal{C}, \mathbf{X}_\mathcal{C}\right) f_{\boldsymbol{\theta}}\left(\mathbf{X}_\mathcal{C}\right) \right].$$

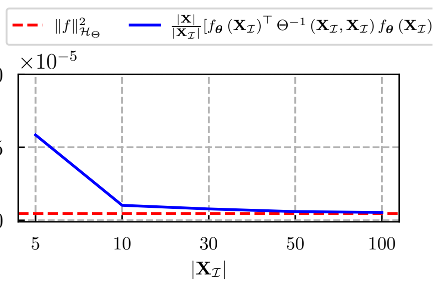


Figure 3: Comparison of function norms.

The experiment is implemented using MNIST on a LeNet style network because we can compute analytic NTK Θ_{ntk} and also invert the full matrix only on a small dataset. We can see that the approximation is rather close when $|\mathbf{X}_\mathcal{C}|$ is no smaller than 10, which is exactly what we use in all our experiments.

Additionally, we also carried out an ablation study on the effect of the size of $|\mathbf{X}_\mathcal{C}|$ on predictive performance when using larger datasets and larger neural network architectures. In Figures 4 and 5, we observe that as long as $|\mathbf{X}_\mathcal{C}|$ is chosen to be above a certain threshold, predictive performance does not increase as the size of the context set increases.

6 Conclusion

In this paper, we proposed a function-space regularization technique for neural networks. The proposed optimization objective explicitly regularizes an approximation to the norm on functions in the RKHS associated with the network's NTK. We showed empirically that this approach yields improved generalization on challenging small-data prediction problems and demonstrated that it leads to state-of-the-art performance in continual learning and protein homology detection tasks. We proposed a set of approximations that allow scaling up the regularization method to very large neural network architectures and hope that future research will enable further improvements in scalability of the proposed method and shed light on its theoretical properties of the regularization method for general loss functions and neural network architectures. We believe that this work provides an important theoretical insight into generalization in neural network, as it demonstrates the usefulness of carefully designed function-space regularizers in improving generalization, and that the proposed approach will inspire further research into function-space regularization methods for deep learning.

References

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8141–8150.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2018). Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*.

Benjamin, A., Rolnick, D., and Kording, K. (2018). Measuring and regularizing networks in function space. In *International Conference on Learning Representations*.

Bietti, A., Mialon, G., Chen, D., and Mairal, J. (2019). A kernel perspective for regularizing deep neural networks. In *International Conference on Machine Learning*, pages 664–674. PMLR.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in neural information processing systems*, 31.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Burt, D. R., Ober, S. W., Garriga-Alonso, A., and van der Wilk, M. (2020). Understanding variational inference in function-space. *arXiv preprint arXiv:2011.09421*.

Conway, J. B. (2019). *A course in functional analysis*, volume 96. Springer.

Cortes, C., Mohri, M., and Rostamizadeh, A. (2009). New generalization bounds for learning kernels. *arXiv preprint arXiv:0912.3309*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. (2017). Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR.

Farquhar, S., Osborne, M. A., and Gal, Y. (2020). Radial bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1352–1362. PMLR.

Geifman, A., Yadav, A., Kasten, Y., Galun, M., Jacobs, D., and Ronen, B. (2020). On the similarity between the laplace and neural tangent kernels. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1451–1461. Curran Associates, Inc.

Håndstad, T., Hestnes, A. J., and Sætrom, P. (2007). Motif kernel generated by genetic programming improves remote homology and fold detection. *BMC bioinformatics*, 8(1):1–16.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580.

Kawaguchi, K., Kaelbling, L. P., and Bengio, Y. (2017). Generalization in deep learning. *arXiv preprint arXiv:1710.05468*.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Khan, M. E. E., Immer, A., Abedi, E., and Korzepa, M. (2019). Approximate inference turns deep networks into gaussian processes. *Advances in neural information processing systems*, 32.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks.

Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Mercer, J. (1909). Functions ofpositive and negativetypeand theircommection with the theory ofintegral equations. *Philos. Trinsdictions Rogyal Soc*, 209:4–415.

Micchelli, C. A., Pontil, M., and Bartlett, P. (2005). Learning the kernel function via regularization. *Journal of machine learning research*, 6(7).

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press.

Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. (1995). Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2018). Variational continual learning. In *International Conference on Learning Representations*.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. (2020). Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*.

Ong, C., Smola, A., and Williamson, R. (2005). Learning the kernel with hyperkernels. *The Journal of Machine Learning Research*, 6:1043–1071.

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R., and Khan, M. E. E. (2020). Continual deep learning by functional regularisation of memorable past. *Advances in Neural Information Processing Systems*, 33:4453–4464.

Rahimi, A., Recht, B., et al. (2007). Random features for large-scale kernel machines. In *NIPS*, volume 3. Citeseer.

Reed, M. and Simon, B. (1972). *Methods of modern mathematical physics*, volume 1. Elsevier.

Rudner, T. G. J., Bickford Smith, F., Feng, Q., Teh, Y. W., and Gal, Y. (2022). Continual Learning via Function-Space Variational Inference. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR.

Rudner, T. G. J., Chen, Z., Teh, Y. W., and Gal, Y. (2021). Tractable Function-Space Variational Inference in Bayesian Neural Networks. In *ICML Workshop on Uncertainty & Robustness in Deep Learning*.

Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer.

Schölkopf, B., Smola, A. J., Bach, F., et al. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sun, S., Zhang, G., Shi, J., and Grosse, R. (2018). Functional variational bayesian neural networks. In *International Conference on Learning Representations*.

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., and Teh, Y. W. (2019). Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34.

Triki, A. R., Berman, M., and Blaschko, M. B. (2018). Function norms and regularization in deep networks.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34.

Williams, C. and Seeger, M. (2000). Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13.

Yu, T., Li, X., Cai, Y., Sun, M., and Li, P. (2022). S2-mlp: Spatial-shift mlp architecture for vision. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 297–306.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021). Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Zhou, P., Feng, J., Ma, C., Xiong, C., Hoi, S. C. H., et al. (2020). Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33:21285–21296.

Supplementary Material

Table of Contents

Appendix A Reproducing Kernel Hilbert Space Theory

Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric and positive definite kernel function. That is to say $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x})$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and $\sum_{1 \le i,j \le N} K(\mathbf{x}_i, \mathbf{x}_j) c_i c_j \ge 0$ hold for any $\{\mathbf{x}_i\}_{i=1}^n \subseteq \mathcal{X}$ for given $n \in \mathbb{N}$ and $\{c_i\}_{i=1}^n \subseteq \mathbb{R}$.

Let $L_2(\mathcal{X}, \mu)$ be the squared integrable function space over $\mathcal{X}$ with respect to a strictly positive finite Borel measure (e.g. probability measure) μ on $\mathcal{X}$, there is an associated integral operator $T_K : L_2(\mathcal{X}, \mu) \to L_2(\mathcal{X}, \mu)$ defined as follows:

$$T_K(f)(\cdot) = \int_{\mathcal{X}} K(\cdot, \mathbf{x}) f(\mathbf{x}) \mathrm{d}\mu(\mathbf{x}). \tag{A.1}$$

If $\mathcal{X}$ is compact, T_K is compact self-adjoint. Spectral theorem (Conway, 2019) then implies T_K has at most countable eigenvalues $\{\sigma_j\}_{j\ge 0}$ such that $T_K(\phi_j) = \sigma_j \phi_j$, where $\{\phi_j\}_{j\ge 0}$ is an orthonormal basis of $L_2(\mathcal{X}, \mu)$ and $\sigma_0 \ge \sigma_1 \ge \cdots$, $\sigma_j \to 0$ as $j \to \infty$. Furthermore, Mercer's theorem (Mercer, 1909) states that K admits a decomposition as follows:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{j=0}^{\infty} \sigma_j \phi_j(\mathbf{x}) \phi_j(\mathbf{x}'). \tag{A.2}$$

According to Moore–Aronszajn theorem (Aronszajn, 1950), for any such kernel K on $\mathcal{X}$ we have a unique *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}_K$, which is the completion of the pre-Hilbert space consisting of all linear span of feature maps $\{K_{\mathbf{x}}(\cdot) : \mathbf{x} \in \mathcal{X}\}$, where $K_{\mathbf{x}}(\cdot) = K(\cdot, \mathbf{x})$. To be precise, we have

$$\mathcal{H}_K = \left\{ f(\cdot) = \sum_{j=1}^{\infty} \alpha_j K(\mathbf{x}^{(j)}, \cdot) : (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots) \subset \mathcal{X} \wedge \|f\|_{\mathcal{H}_K}^2 = \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) < \infty \right\}. \tag{A.3}$$

The kernel function K is called *reproducing kernel* since for any $f \in \mathcal{H}_K$, we have the following reproducing property:

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}_K}. \tag{A.4}$$

One can show that the RKHS can also be viewed as a subspace of $L_2(\mathcal{X}, \mu)$ as follows:

$$\mathcal{H}_K = \left\{ f \in L_2(\mathcal{X}, \mu) : \|f\|^2_{\mathcal{H}_K} = \sum_{j=0}^{\infty} \frac{\langle f, \phi_j \rangle^2_{L_2}}{\sigma_j} < \infty \right\}. \tag{A.5}$$

We make the following observations:

- As we can see from Eqn. (A.3), the computation of norm depends on the choice of $\mathbf{x}_i$, hence is not uniform for general f if the underlying space $\mathcal{X}$ is uncountable. This implies Eqn. (3) is not true for general f. But as a result of Representer Theorem, the hypothesis space for kernel regression is $\mathcal{H} = \{f(\cdot) = \sum_{j=1}^{N} \alpha_j K(\mathbf{x}^{(j)}, \cdot) : \mathbf{x}^{(j)} \in \mathbf{X}\}$, which is a finite dimensional subspace of $\mathcal{H}_K$. This is the same as taking $\mathcal{X} = \mathbf{X}$, which is finite. Hence Eqn. (3) is valid for kernel regression.
- By Eqn. (A.5), we can easily see that $\|f\|^2_{\mathcal{H}_K} \geq c \sum_{j=0}^{\infty} \langle f, \phi_j \rangle^2_{L_2} = c \|f\|^2_{L_2}$ where $c = \sigma_0^{-1}$. Therefore, RKHS norm provides a stronger regularizer than L_2 norm.

Appendix B Proofs and Derivations

B.1 Discrepancy Between Predictive Functions

The proof of Theorem 2 requires a study of neural network training dynamics, and Theorem 3 can be proved by applying the classical kernel ridge regression bound. We start by recalling the definition of fully-connected neural networks:

$$\begin{aligned} f^{(h)}(\mathbf{x}) &= \mathbf{W}^{(h)} g^{(h-1)}(\mathbf{x}) \in \mathbb{R}^{d_h} \quad g^{(0)}(\mathbf{x}) = \mathbf{x} \\ g^{(h)}(\mathbf{x}) &= \sqrt{\frac{c_\sigma}{d_h}} \sigma\big(f^{(h)}(\mathbf{x})\big) \quad \forall h = 1, 2, \cdots, L. \end{aligned} \tag{B.6}$$

And the last layer is given by

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = f^{(L+1)}(\mathbf{x}) = \mathbf{W}^{(L+1)} g^{(L)}(\mathbf{x}). \tag{B.7}$$

The optimization problem we propose is

$$\min_{\boldsymbol{\theta}} \mathcal{L}^{\hat{\mathcal{R}}} = \min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{j=1}^{N} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)})) + \zeta f_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}). \tag{B.8}$$

Proof of Equivalence. We first study the dynamics of problem (B.8). Suppose the loss function is squared loss $\ell(f) = (f(x) - y)^2$, the continuous SGD is

$$\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}^{\hat{\mathcal{R}}}}{\mathrm{d}\boldsymbol{\theta}} = -\eta \bigg(\frac{2}{N} \mathcal{J}_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{X})^\top (f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta \mathcal{J}_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}) \bigg). \tag{B.9}$$

Therefore, the training dynamics is

$$\frac{\mathrm{d}f_{\boldsymbol{\theta}}}{\mathrm{d}t}(\mathbf{X}) = \mathcal{J}_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{X}) \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} = -\eta \bigg(\frac{2}{N} \Theta(\mathbf{X}, \mathbf{X})(f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta \Theta(\mathbf{X}, \mathbf{X}) \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}) \bigg). \tag{B.10}$$

Note that in practice, we use the empirical kernel Θ in the training, and the inverse kernel matrix is not involved in auto-differentiation in each gradient update step. This gives another training dynamics:

$$\begin{aligned} \frac{\mathrm{d}f_{\boldsymbol{\theta}}}{\mathrm{d}t}(\mathbf{X}) &= -\eta \bigg(\frac{2}{N} \Theta(\mathbf{X}, \mathbf{X})(f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta \Theta(\mathbf{X}, \mathbf{X}) \Theta(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}) \bigg) \\ &= -\eta \bigg(\frac{2}{N} \Theta(\mathbf{X}, \mathbf{X})(f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta f_{\boldsymbol{\theta}}(\mathbf{X}) \bigg) \end{aligned} \tag{B.11}$$

First, we prove these two dynamics are close, hence our practical algorithm can approximate the true solution very well.

Lemma 1. *Let f^*_{ntk} and f_{entk} be the solutions of* (B.10) *and* (B.11)*, respectively, with the same initial condition. Assume f is bounded, then for ultra-wide networks, we have*

$$\sup_{t\geq 0} \|f^*_{ntk}(\mathbf{X}) - f_{entk}(\mathbf{X})\| < \gamma\varepsilon. \tag{B.12}$$

Proof. By comparing the dynamics in (B.10) and (B.11), we have

$$\begin{aligned}
&-\frac{1}{\eta}\left[\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{\theta}} - \frac{\mathrm{d}g}{\mathrm{d}\boldsymbol{\theta}}\right] \\
&= \frac{2}{N}\Theta_f(f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) - \frac{2}{N}\Theta_g(g_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta\Theta_f\Theta_{ntk}^{-1}f_{\boldsymbol{\theta}}(\mathbf{X}) - 2\zeta g_{\boldsymbol{\theta}}(\mathbf{X}) \\
&= \frac{1}{N}\big(2\Theta_f + 2\zeta\Theta_f\Theta_{ntk}^{-1}\big)(f_{\boldsymbol{\theta}}(\mathbf{X}) - g_{\boldsymbol{\theta}}(\mathbf{X})) + \big(\Theta_f - \Theta_g\big)\bigg(\frac{2}{N}(g_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta\Theta_{ntk}^{-1}g_{\boldsymbol{\theta}}(\mathbf{X})\bigg).
\end{aligned} \tag{B.13}$$

Here for short, we use $\Theta_f = \Theta_f(\mathbf{X}, \mathbf{X})$ for empirical kernel matrix with respect to f (and g similarly). Let $\mathbf{A} = \frac{1}{N}\big(2\Theta_f + 2\zeta\Theta_f\Theta_{ntk}^{-1}\big)$, $\boldsymbol{v} = \big(\Theta_f - \Theta_g\big)\big(\frac{2}{N}(g_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta\Theta_{ntk}^{-1}g_{\boldsymbol{\theta}}(\mathbf{X})\big)$ and $h = f - g$. Also let λ_0 be the small eigenvalue of Θ_{ntk}. Since $h(0) = 0$, we have

$$f^*_{ntk}(\mathbf{X}) - f_{entk}(\mathbf{X}) = h(t) = -\eta e^{-\eta\int_0^t \mathbf{A}\mathrm{d}t}\int_0^t e^{\int_0^s \eta\mathbf{A}\mathrm{d}t}\boldsymbol{v}(s)\mathrm{d}s \tag{B.14}$$

Note that for ultra-wide neural networks, we have $\|\Theta - \Theta_{ntk}\|_F \leq N(L+1)\varepsilon$ (Theorem 3.1 Arora et al. (2019)), hence

$$\|\Theta_f - \Theta_g\| \leq \|\Theta_f - \Theta_g\|_F \leq \|\Theta_f - \Theta_{ntk}\|_F + \|\Theta_g - \Theta_{ntk}\|_F \leq 2N(L+1)\varepsilon \tag{B.15}$$

So we have

$$\begin{aligned}
\|\boldsymbol{v}\| \leq& \|\Theta_f - \Theta_g\|\left(\frac{2}{N}\|g_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}\| + 2\zeta\left\|\Theta_{ntk}^{-1}g_{\boldsymbol{\theta}}(\mathbf{X})\right\|\right) \\
\leq& 4N(L+1)\varepsilon\bigg(C_1 + \zeta\frac{NC_0}{\lambda_0}\bigg) := C''\varepsilon.
\end{aligned} \tag{B.16}$$

Let $\mathbf{A}_0 = \frac{1}{N}(2\Theta_{ntk} + 2\zeta I)$, then

$$\|\mathbf{A} - \mathbf{A}_0\| \leq \frac{1}{N}(2\|\Theta_f - \Theta_{ntk}\| + 2\zeta\left\|\Theta_{ntk}^{-1}\right\|\|\Theta_f - \Theta_{ntk}\|) \leq 2(L+1)\varepsilon\bigg(1 + \frac{\zeta}{\lambda_0}\bigg). \tag{B.17}$$

Hence by Lemma 2, we have

$$\begin{aligned}
\lambda_{\min}(\int_s^t \mathbf{A}\mathrm{d}s) \geq& \lambda_{\min}(\int_s^t \mathbf{A}_0\mathrm{d}s) - \left\|\int_s^t (\mathbf{A} - \mathbf{A}_0)\mathrm{d}s\right\| \\
\geq& \bigg[\frac{2}{N}(\lambda_0 + \zeta) - 2(L+1)\varepsilon\bigg(1 + \frac{\zeta}{\lambda_0}\bigg)\bigg](t-s) := C'(t-s).
\end{aligned} \tag{B.18}$$

Hence we see that

$$\begin{aligned}
\|h(t)\| &\leq \eta\int_0^t \left\|e^{\int_t^s \eta\mathbf{A}\mathrm{d}t}\right\|\|\boldsymbol{v}\|\mathrm{d}s \\
&\leq \eta C''\varepsilon\int_0^t e^{-\eta\lambda_{\min}(\int_s^t \mathbf{A}\mathrm{d}t)}\mathrm{d}s \\
&\leq \eta C''\varepsilon\int_0^t e^{\eta C'(s-t)}\mathrm{d}s \\
&\leq \frac{\eta C''\varepsilon}{\eta C'}(1 - e^{-\eta C't}) \leq \frac{C''}{C'}\varepsilon.
\end{aligned} \tag{B.19}$$

This proves the lemma by letting $\gamma = C''/C'$. □

Lemma 2. *Let A, B be $n \times n$ Hermitian matrices, arrange the eigenvalues in descending order as $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ (and for B as well). Then for all $1 \leq i \leq n$, we have*

$$|\lambda_i(A+B) - \lambda_i(A)| \leq \|B\| \,. \tag{B.20}$$

Proof. Courant-Fischer min-max theorem (Reed and Simon (1972)) implies that for any Hermitian matrix, we have

$$\lambda_i(A) = \sup_{\dim V = i} \inf_{v \in V, \|v\|=1} v^* A v \tag{B.21}$$

$$\lambda_i(A) = \inf_{\dim V = n-i+1} \sup_{v \in V, \|v\|=1} v^* A v. \tag{B.22}$$

According to Equation (B.21), we can find a subspace U with $\dim U = i$, such that $\lambda_i(A) \geq u^* A u$ for all unit vector $u \in U$. And similarly using Equation (B.22), we can find a subspace W with $\dim W = n - i + 1$, we have $\lambda_i(A+B) \leq w^*(A+B)w$. Since $\dim U \cap W = \dim U + \dim W - \dim U \cup W \geq i + n - i + 1 - n = 1$, we find a non-trivial solution $v \in U \cap W$ such that

$$\lambda_i(A+B) - \lambda_i(A) \leq v^*(A+B)v - v^* A v = v^* B v \leq \|B\| \,. \tag{B.23}$$

Similarly we can prove $\lambda_i(A+B) - \lambda_i(A) \geq -\|B\|$ using the same argument. These two inequalities prove the claim of the lemma. □

Theorem 4. *Let f be an m-layer fully-connected neural network mapping with ReLU activation functions, $f_{nn}^{\hat{\mathcal{R}}*}$ and $f_{ntk}^{\mathcal{R}*}$ as defined above, $1/\kappa = \text{poly}(1/\varepsilon, \log(N/\delta))$, $d_1 = d_2 = \cdots = d_L = m$ with $m \geq \text{poly}(1/\kappa, L, 1/(\lambda_0 + \zeta N))$. Then for any $\mathbf{x}$ with $\|\mathbf{x}\| = 1$, with probability at least $1 - \delta$ over random initialization,*

$$|f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) - f_{ntk}^{\mathcal{R}*}(\mathbf{x})| \leq \varepsilon. \tag{B.24}$$

Proof. Note that the training dynamics B.11 can be written as

$$\frac{\mathrm{d} f_{\boldsymbol{\theta}}}{\mathrm{d}t} + \frac{2\eta}{N}(\Theta + \zeta N \mathbf{I}) f_{\boldsymbol{\theta}} = \frac{2\eta}{N} \Theta \mathbf{y}. \tag{B.25}$$

if we replace Θ by the analytic kernel Θ_{ntk}, the linear system above becomes

$$\frac{\mathrm{d} f_{\boldsymbol{\theta}}}{\mathrm{d}t} + \frac{2\eta}{N}(\Theta_{ntk} + \zeta N \mathbf{I}) f_{\boldsymbol{\theta}} = \frac{2\eta}{N} \Theta_{ntk} \mathbf{y}. \tag{B.26}$$

Since Θ_{ntk} is constant, we have a closed form solution

$$f_t(\mathbf{x}) = \Theta_{ntk}(\mathbf{x}, \mathbf{X}) \big(\Theta_{ntk}(\mathbf{X}, \mathbf{X}) + \zeta N \mathbf{I}\big)^{-1} \left(I - e^{-\frac{2\eta}{N} t \Theta_{ntk}(\mathbf{X}, \mathbf{X})} \right) \mathbf{y}. \tag{B.27}$$

This is exactly the solution of kernel ridge regression under the NTK when $t \to \infty$, i.e. $f_t \to f_{ntk}^{\mathcal{R}*}$. Taking the difference of the two linear systems, we have

$$\frac{\mathrm{d} h}{\mathrm{d}t} + \frac{2\eta}{N}(\Theta_{ntk} + \zeta N \mathbf{I}) h = \frac{2\eta}{N}(\Theta - \Theta_{ntk})(\mathbf{y} - f). \tag{B.28}$$

Let $\mathbf{B} = \frac{2\eta}{N}(\Theta_{ntk} + \zeta N \mathbf{I})$ and $\mathbf{u} = \frac{2\eta}{N}(\Theta - \Theta_{ntk})(\mathbf{y} - f)$, we see that

$$f_{entk}(\mathbf{X}) - f_{ntk}^{\mathcal{R}*}(\mathbf{X}) = h(t) = e^{-\int_0^t \mathbf{B} \mathrm{d}t} \int_0^t e^{\int_0^s \mathbf{B} \mathrm{d}t} \mathbf{u}(s) \mathrm{d}s. \tag{B.29}$$

This implies

$$\begin{aligned} |h(t)| &\leq \int_0^t \left\| e^{\int_s^t \mathbf{B} \mathrm{d}t} \right\| \|\mathbf{u}\| \, \mathrm{d}s \\ &\leq \frac{2\eta}{N} \|\Theta - \Theta_{ntk}\| \, \|\mathbf{y} - f(\mathbf{X})\| \, \frac{1}{\|\mathbf{B}\|} \big(1 - e^{-t\|\mathbf{B}\|}\big) \leq \frac{N C_0 \varepsilon}{\lambda_0 + \zeta N}, \end{aligned} \tag{B.30}$$

where C_0 is constant bounds the function value and λ_0 is the smallest eigenvalue of the NTK. Equation (B.30) together with Theorem 1 prove the Theorem on training set.

To complete the proof for any test point $\mathbf{x}$, note that we only need to mimic the proof of Lemma F.1 in Arora et al. (2019). All the bounds remain the same, except now Θ_{ntk} is replaced by $\Theta_{ntk} + \zeta N \mathbf{I}$, so λ_0 is replaced by $\lambda_0 + \zeta N$. □

Analytic and Empirical NTK. Here we provide the details about using the empirical NTK Θ instead of the analytic NTK Θ_{ntk} as we discussed in Section 4. Although we propose to use function norm under the analytic NTK Θ_{ntk} as a regularizer in Eqn. (11), this exact computation is costly in practice, so the empirical kernel is used instead.

Specifically, in practice we replace Θ_{ntk} in Eqn. (B.8) by Θ, and furthermore we do not compute the gradient of the inverse NTK Θ during SGD since the analytical kernel is constant. The kernel approximation has been well studied in the literature. Under the infinite-width assumption, that is, $m \to \infty$, $\Theta \to \Theta_{ntk}$ (Jacot et al., 2018). This result was extended to the non-asymptotic case in Arora et al. (2019), which implies that $|\Theta(\mathbf{x}, \mathbf{x}') - \Theta_{ntk}(\mathbf{x}, \mathbf{x}')| \le (L+1)\varepsilon$. Hence, intuitively, the two objective functions using the analytic and empirical kernel are almost identical for sufficiently-wide neural networks. The corresponding training dynamics are described by Eqn. (B.10) and (B.11). As we have proven in Lemma 1, these dynamics are sufficiently close during training, which indicates that the approximation by the empirical kernel is theoretically justified.

B.2 Generalization Error Bound

Theorem 5. *Let $f_{nn}^{\hat{\mathcal{R}}*}$ be as defined in the main text. Let $R(f_{nn}^{\hat{\mathcal{R}}*}) = \mathbb{E}[(f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) - \mathbf{y}(\mathbf{x}))^2]$ be the generalization error and $\hat{R}(f_{nn}^{\hat{\mathcal{R}}*})$ the corresponding empirical error. Assume for all $\|\mathbf{x}\| \le 1$, $|f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x})| \le C_0\kappa$ and $\left\|\partial_{\boldsymbol{\theta}} f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x})\right\| \le C_1$ and $|\mathbf{y}| \le c$. Then for any $\delta > 0$, with probability at least $1-\delta$, we have the following bound:*

$$R(f_{nn}^{\hat{\mathcal{R}}*}) - \hat{R}(f_{nn}^{\hat{\mathcal{R}}*}) \le 4(C_0\kappa + c)\varepsilon + \frac{8C_1^2\Lambda^2}{\sqrt{N}}\left(\sqrt{\frac{C_K}{NC_1^2}} + \frac{3}{4}\sqrt{\frac{\log\frac{2}{\delta}}{2}}\right), \tag{B.31}$$

where N is the number of training points and Λ is a constant that only depends on the regularization coefficient ζ. C_K is the trace of the empirical NTK. ε is the discrepancy between predictive functions from Theorem 2.

Proof. First note that the definition of neural network function in the very beginning is homogeneous since it is bias free, and ReLU is also homogeneous. Hence for any t, we have $f(t\mathbf{x}) = tf(\mathbf{x})$. Therefore, it is sufficient to assume $\|\mathbf{x}\| \le 1$ for all $\mathbf{x} \in \mathcal{X}$.

Since we have proven the equivalence between neural network optimization and kernel ridge regression, we have

$$R(f_{nn}^{\hat{\mathcal{R}}*}) - R(f_{ntk}^{\mathcal{R}*}) = \mathbb{E}\big[\big(f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) - f_{ntk}^{\mathcal{R}*}(\mathbf{x})\big)\big(f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) + f_{ntk}^{\mathcal{R}*}(\mathbf{x}) - 2\mathbf{y}(\mathbf{x})\big)\big] \le 2\varepsilon(C_0\kappa + c), \tag{B.32}$$

where C_e is the supremum of the error term. Similarly, we have

$$\hat{R}(f_{ntk}^{\mathcal{R}*}) - \hat{R}(f_{nn}^{\hat{\mathcal{R}}*}) \le 2\varepsilon(C_0\kappa + c) \tag{B.33}$$

Therefore,

$$R(f_{nn}^{\hat{\mathcal{R}}*}) - \hat{R}(f_{nn}^{\hat{\mathcal{R}}*}) \le R(f_{ntk}^{\mathcal{R}*}) - \hat{R}(f_{ntk}^{\mathcal{R}*}) + 4\varepsilon(C_0\kappa + c). \tag{B.34}$$

Classical result for kernel ridge regression (e.g. Theorem 10.7 Mohri et al. (2012)) implies

$$R(f_{ntk}^{\mathcal{R}*}) - \hat{R}(f_{ntk}^{\mathcal{R}*}) \le \frac{8r^2\Lambda^2}{\sqrt{N}}\left(\sqrt{\frac{\mathrm{Tr}(\Theta_{\mathrm{ntk}})}{Nr^2}} + \frac{3}{4}\sqrt{\frac{\log\frac{2}{\delta}}{2}}\right), \tag{B.35}$$

where r and Λ are constants such that $\Theta_{ntk}(\mathbf{x}, \mathbf{x}) \le r^2$ and $\|f\|_{\mathcal{H}_K} \le \Lambda$. Note that the condition $\|f\|_{\mathcal{H}_K} \le \Lambda$ corresponds to the regularizer, hence Λ is uniquely determined by ζ. We can take $r = C_1$ since $\Theta_{ntk}(\mathbf{x}, \mathbf{x}) \approx \langle \mathcal{J}_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}), \mathcal{J}_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})\rangle \le C_1^2$ under the ultra-wide assumption.

It has been proven in Geifman et al. (2020) that the eigenvalues of the NTK satisfy $ck^{-d} \le \lambda_k \le Ck^{-d}$ for all $k > k_0$, where d is the dimension of input data points. Therefore, $\mathrm{Tr}(\Theta_{ntk}) \le \lambda_1 + \cdots + \lambda_{k_0} + C\sum_{k>k_0} k^{-d} := C_K < \infty$. Combining these facts completes the proof of the theorem. $\square$

Appendix C Ablation Studies

C.1 Empirical Evaluation of Time Complexity

We provide a comparison on the time of every epoch under ResNet-18 using a single RTX 2080 Ti in Table 4. Although $\mathcal{K}^{\Theta}$ is slower than maximum a posteriori estimation (MAP), this is not a significant issue, since we propose to use $\mathcal{K}^{\Theta}$ in small-data problems, so computational time is not the main bottleneck. Moreover, compared to other regularization techniques like $\mathcal{K}^{\mathcal{J}}$ $\mathcal{K}^{\Theta}$ has similar computational cost but demonstrates better performance in practice.

Table 4: Wall-clock time (in second) of different methods.

Method	$\mathcal{K}^{\Theta}$	$\mathcal{K}^{\mathcal{J}}$	Dropout	MAP
Time (s)	170	141	26	25

C.2 Batch Normalization in NTK Computation

To remain consistent with NTK theory, we took special care in implementing batch normalization (BN). To compute the normalization mean and standard deviation deterministically at every iteration, we use a fixed "conditioning set" of 20 data points, $\mathbf{X}_{\text{cond}}$, randomly sampled from the training set at the beginning of training. Normalization means and variances for a given NTK evaluation point x_i are then computed by evaluating the network on $[x_i, \mathbf{X}_{\text{cond}}]$. Vectorization allows implementing this procedure efficiently with a cost of $\mathcal{O}(|\mathbf{X}_{\mathcal{B}}| + |\mathbf{X}_{\text{cond}}|)$ for a mini-batch $\mathbf{X}_{\mathcal{B}}$.

Appendix D Implementation, Training, and Evaluation Details

We use 10% of the training set as the validation set to conduct a hyperparameter search over the scaling factor ζ and learning rate η for all methods. We choose the set of hyperparameter that yielded the lowest validation negative log-likelihood for all experiments. All experiments are implemented in JAX (Bradbury et al., 2018).

D.1 MNIST and CIFAR-10

For MNIST, we use a LeNet style network with three convolutional layers of 6, 16 and 120 5×5 filters and a fully-connected final layer of 84 hidden units. An average pooling operation is placed after each convolutional layer and ReLU activation is used. For CIFAR-10, we use ResNet-18. For both MNIST and CIFAR-10, we use SGD optimizer with a learning rate of 0.03 and momentum of 0.9. The learning rate would decay by a rate of 0.3 at every 10 epochs. During training, batch size is set to 128 for MNIST and 256 for CIFAR-10. At every iteration, 10 input points are randomly sampled from the current batch as $\mathbf{X}_{\mathcal{C}}$ to compute the RKHS norm.

When we compare our method against $\mathcal{K}^{L_2}$ and $\mathcal{K}^{\mathcal{J}}$, we copy the same experimental set-up as in the original paper. For MAP and Dropout, we use a outstanding validation set to select the best hyper-parameter. In the end, we find that for MAP, the best ζ is $5e^{-4}$, and for Dropout, the best dropout rate is 0.15.

D.2 MLP-Mixer

We use MLP-Mixer of B/16 architecture (Tolstikhin et al., 2021) and load the pretrained parameters on ImageNet from public source online. The training of MLP-Mixer on CIFAR-100 requires 8 GTX 3090 GPUs. We use the SGD optimizer with learning rate 0.01 and momentum 0.9.

D.3 Continual Learning

For all continual learning experiments, 60,000 data samples are used for training and 10,000 data samples are used for testing. The input are converted to float values in the range of [0, 1].

Multi-Head Split MNIST In the multi-head setup, a model receives 5 sequential datasets and uses a different output head for each task. The original 10 classes in MNIST is split into 5 different binary classification tasks. The network is a multilayer perceptron with two layers and 100 hidden units for each layer. 40 coreset points are randomly chosen at each task.

Single-Head Permuted MNIST In the single-head setup, a model receives 10 sequential datasets and uses a same output head for each task. At every task, the MNIST images undergo a random permutation of pixels. The network is a multilayer perceptron with two layers and 256 hidden units for each layer. 200 coreset points are randomly chosen at each task.

For both multi-head split MNIST and single-head permuted MNIST, we use the Adam optimizer of learning rate 10^{-3} with default settings of $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$ and we use batch size of 128 in all experiments.

D.4 Protein Homology Detection

The protein homology detection experiment used the dataset of Structural Classification Of Proteins (SCOP) version 1.67Murzin et al. (1995) and we followed the data preprocessing method in Håndstad et al. (2007). We construct 100 balanced binary classification dataset by sampling 200 positive samples from a specific family and sampling 200 negative samples outside that family. If the number of positive samples do not reach 100, we extend their numbers to 100 by using PSI-BLAST (Altschul et al., 1997). In order to remain comparable to the results in Bietti et al. (2019), we also follow their data augmentation techniques by randomly flipping some of the binary characters of a given sequence at probability 0.1. We use the Adam optimizer with a learning rate fixed to 0.01 (with default β (0:9; 0:999)), and a batch size of 100 for 300 epochs.