OPT2022: 14th Annual Workshop on Optimization for Machine Learning 1–27

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

Zonghao Chen[1][2], *University College London*
Xupeng Shi[1], *Northeastern University*
Tim G. J. Rudner, *University of Oxford*
Qixuan Feng, *University of Oxford*
Weizhong Zhang, *HKUST*
Tong Zhang, *HKUST*

Abstract

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 performance 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 amount of data is available. However, in areas where data is scarce or data labeling is expensive, neural networks may over-fit easily and require particularly effective regularization techniques to systematically control model complexity and improve generalization (Vapnik, 1998).

The empirical phenomenon of double decent (Belkin et al., 2018) has 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 parameter regularization is unable to effectively control model complexity (Triki et al., 2018), that neural network parameter regularization is a poor proxy for regularization on the predictive functions (Benjamin et al., 2018; Rudner et al., 2022a,b), and that minimizing the parameter norm is not necessary to obtain good generalization (Zhang et al., 2021; Kawaguchi et al., 2017). These observations raise the question of why parameter-space regularization techniques fall short and to what extent over-parameterized neural networks

1. Equal contribution.
2. Correspondence to: zonghao.chen.22@ucl.ac.uk.

may benefit from alternative regularization techniques that do not explicitly regularize the network parameters but rather regularize an alternative quantity more closely related to neural network complexity.

In this paper, we propose to directly regularize the norm of function mappings induced by the neural network architecture to more effectively control the complexity of the learned function and improve generalization in over-parameterized models. A natural candidate for such a function norm would be the *reproducing kernel Hilbert space* (RKHS) norm, but there is no direct evidence that the function space defined by a neural network is identical to an RKHS $\mathcal{H}_K$. To address this disconnect, 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 2 we propose to compute the norm of functions f in the same way and use the function norm as a function-space regularizer. We further prove in Theorem 1 that trained neural network predictive functions under this regularizer are arbitrarily close to kernel ridge regression solutions under NTK and we also provide a generalization error bound in Theorem 2.

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) proposed to use the Jacobian norm as a lower bound on the RKHS norm. However, both methods fall short, as the L_2 space does not contain essential topological information, and regularizing the lower bound on the RKHS norm 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 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 highly-overparameterized neural networks that are prone to over-fitting (Tolstikhin et al., 2021; Yu et al., 2022). The proposed function-space regularizer can effectively alleviate over-fitting to achieve improved generalization. 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 training that penalizes complexity in the predictive functions by jointly minimizing the loss and the norm on the functions in the RKHS associated with a 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 and demonstrate that it results in state-of-the-art performance on downstream prediction tasks where effective regularization is particularly important.

2. 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 $\mathbf{y}^{(n)} \in \mathcal{Y} \subseteq \mathbb{R}^Q$. 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.

2.1. 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 considers the objective function

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

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 $K_{\mathbf{x}^{(1)}}, \ldots, K_{\mathbf{x}^{(N)}}$ evaluated on the training set, that is, for an arbitrary evaluation point $\mathbf{x}$,

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

This solution 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:

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

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

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

Eqn. (4) shows that the RKHS norm is completely determined by function values and the inverse of the kernel matrix. In Section 3, we will show how to exploit this structure to construct a simple function-space regularizer for neural network training.

2.2. Kernel Methods and Deep Learning

We recursively define an L-layer fully-connected neural network by

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

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 an 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 then given by

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

where $\boldsymbol{\theta} = \mathrm{vec}(\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L+1)})$ represents a vectorization of all parameters in the network. Jacot et al. (2018) define 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* fully-connected ReLU neural network, f_{nn}^*, can be made to be arbitrarily close to the predictions under the analytic NTK Θ_{ntk} kernel regression solution f_{ntk}^*:

Theorem (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, $1/\kappa = \mathrm{poly}(1/\varepsilon, \log(N/\delta))$, $d_1 = d_2 = \ldots = d_L = m$ with $m \geq \mathrm{poly}(1/\kappa, L, 1/\lambda_0, n\log(1/\delta))$. Then for any $\mathbf{x}$ with $\|\mathbf{x}\| = 1$, with probability at least $1 - \delta$ over the random initialization,*

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

3. Approximate Function-Space Regularization for Neural Networks

In this section, we extend the Theorem in Arora et al. (2019) to kernel ridge regression and use this simple extension to motivate a function-space regularization method for neural network training.

Since the explicit function-space regularizer in Equation (1) cannot be computed tractably for arbitrary neural network architectures, we consider a variation of the setting of the theorem above 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:

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

To justify this approximate regularizer, we return to the regularized kernel ridge regression problem in Equation (1). We denote as $f_\infty^{\mathcal{R}*}$ the solution of the regularized kernel ridge regression problem under the analytic NTK derived from an infinitely-wide neural network, Θ_{ntk}. For an ℓ_2 squared loss function, the solution $f_{\infty}^{\mathcal{R}*}$ can be expressed by

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

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 $f_{nn}^{\hat{\mathcal{R}}*} = \arg\min_{f_{\boldsymbol{\theta}}} \mathcal{L}^{\hat{\mathcal{R}}}(f_{\boldsymbol{\theta}})$, should be close to $f_{\infty}^{\mathcal{R}*}$. The following theorem shows that $f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x})$ and $f_{\infty}^{\mathcal{R}*}(\mathbf{x})$ can indeed be made arbitrarily close to one another:

Theorem 1 *Let f be an m-layer fully-connected neural network mapping with ReLU activation functions, $f_{nn}^{\hat{\mathcal{R}}*}$ and $f_{\infty}^{\mathcal{R}*}$ as defined above, $1/\kappa = \text{poly}(1/\varepsilon, \log(N/\delta))$, $d_i = m$ for $i = 1, ..., L$ 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,*

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

Proof See Appendix 2. ■

Furthermore, based on Theorem 1 and generalization theory for kernel ridge regression, we can obtain the following generalization error bound for the solution $f_{nn}^{\hat{\mathcal{R}}*}$:

Theorem 2 *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:*

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

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 1.

Proof See Appendix 2. ■

Theorems 1 and 2 demonstrate that for fully-connected ReLU neural networks and with 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 analogous results cannot be derived for more general loss functions (e.g., cross-entropy loss functions) and neural network architectures (e.g., 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.

Algorithm 1 Neural Network Optimization with Function-Space Regularization

```
1: Initialize $\boldsymbol{\theta}$
2: for training epoch $m = 1, 2, \ldots, M$ do
3:     for each training iteration do
4:         Sample a mini-batch of data $\mathcal{B} = \left\{ \left(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}\right), \ldots, \left(\mathbf{x}^{(B)}, \mathbf{y}^{(B)}\right) \right\}$
5:         Sample the context set from 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 $\boldsymbol{\theta}$
```

3.1. Optimization Objective and Computational Cost

There are several challenges to computing the function-space regularizer. The first challenge is the computation of the analytic NTK Θ_{ntk}. Although Arora et al. (2019) 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 Equation (7).

The second challenge is the computation of the empirical NTK Θ itself. Computing the empirical NTK $\Theta(\mathbf{X}, \mathbf{X})$ naively has $\mathcal{O}(P^2)$ space complexity and $\mathcal{O}(N^2 P)$ 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). By noting that the empirical NTK can be written as

$$\Theta(\mathbf{X}, \mathbf{X}) = J(\mathbf{X}) \, J(\mathbf{X})^\top = d\big[J(\mathbf{X}) \, J(\mathbf{X})^\top \mathbf{v}\big] / d\mathbf{v}^\top$$

for a vector $\mathbf{v}$, the implicit computation can be expressed as a memory-efficient, nested application of two Jacobian-vector products instead of instantiating the Jacobian explicitly and computing its inner product.

The third challenge is that computing the function norm in Equation (11) requires inverting the empirical 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 a context set $\mathbf{X}_\mathcal{C}$ from $\mathbf{X}$ at every gradient step and compute the function norm on this particular context set 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.

For the context set $\mathbf{X}_\mathcal{C}$ sampled uniformly from the entire dataset $\mathbf{X}$ at every gradient step $\mathbf{X}_\mathcal{C} \sim p_\mathbf{X}$, the final, approximate function-space-regularization objective is given by

$$\mathcal{L}^{\hat{\mathcal{R}}}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)} | f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) + \zeta f_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C})^\top \Theta^{-1}(\mathbf{X}_\mathcal{C}, \mathbf{X}_\mathcal{C}) f_{\boldsymbol{\theta}}(\mathbf{X}_\mathcal{C}). \tag{13}$$

Algorithm 1 describes stochastic gradient descent under this optimization objective.

4. Related Work

Function-Space Regularization. Benjamin et al. (2018) use the L_2 norm as a trivial function-space norm and (Bietti et al., 2019) use the Jacobian norm as a lower bound of the function norm. Both methods fall short, as the L_2 space is a trivial function space that loses essential topological information, and regularizing the lower bound has no direct relation to controlling the function complexity. Bietti and Mairal (2018) construct an RKHS which contains CNN prediction functions, but unfortunately, the RKHS is of very complicated forms and is not applicable in practice.

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., 2022b) and continual learning (Titsias et al., 2019; Pan et al., 2020; Rudner et al., 2022a). These function-space regularization techniques directly constrain the space of functions and as such allow for more explicit incorporation of prior information about the functions to be learned.

In the context of variational inference, function-space approaches seek to avoid the limitations of parameter-space variational inference (Blundell et al., 2015; Farquhar et al., 2020) and have been shown to achieve better uncertainty estimation and calibration (Rudner et al., 2022b). However, many function-space approaches to regularization lack scalability. Sun et al. (2018) use an expensive gradient estimator which limits the application to low-dimensional data and Pan et al. (2020) require expensive matrix inversion operations. Ma and Hernández-Lobato (2021) propose to use a grid Kullback-Leibler (KL) divergence to replace the traditional Kullback-Leibler divergence.

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

Generalization in Neural Networks. Classical learning theory attributes the ability to generalize well to low-capacity classes of hypothesis space (Vapnik, 1998; Mohri et al., 2012; Bartlett and Mendelson, 2002), but empirical findings about the performance of neural networks to generalize well have contradicted those results, demonstrating that good generalization can be achieved despite, and possibly due to, over-parameterization (Zhang et al., 2021; Kawaguchi et al., 2017). Keskar et al. (2016) argue that flat minima can lead to better generalization, but Dinh et al. (2017) show that sharp minima can also lead to good generalization. Neyshabur et al. (2019) propose a novel measure based on unit-wise capacities for two-layer ReLU networks. Allen-Zhu et al. (2020) establish a new notion of quadratic approximation applicable beyond two-layer neural networks. Why over-parameterization in neural networks leads to improved generalization remains an open question.

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). Kernel methods have also been considered in neural networks. Williams (1998) show that the covariance function induced by a distribution over the parameters of an infinite-width neural network can be given in closed form. Jacot et al. (2018) consider the training dynamics of neural network outputs and introduced the neural tangent kernel (NTK) and Lee et al. (2019) show that wide deep neural networks evolve approximately as linear models.

5. Empirical Evaluations

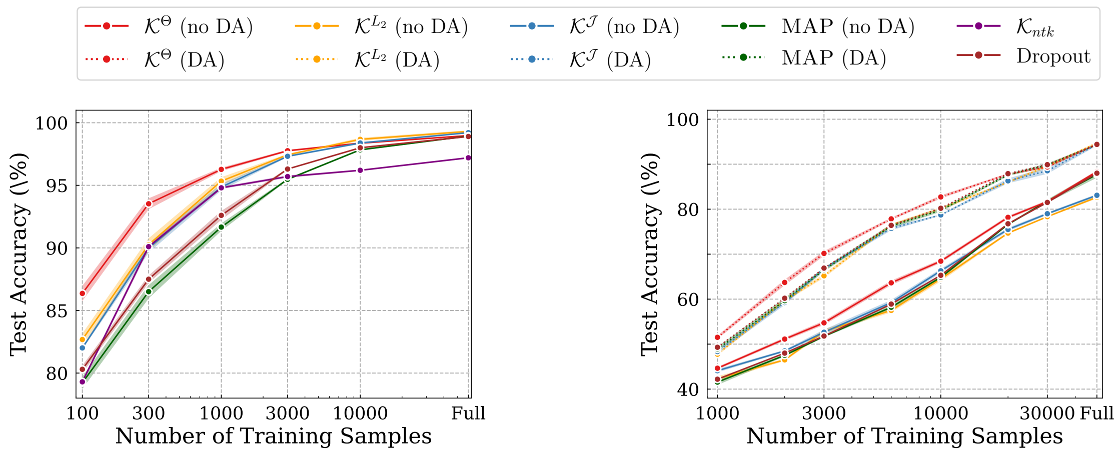
In Sections 2 and 3, we proposed a function-space regularization technique for neural network training. In this section, we evaluate the empirical performance of this method. To do so, we consider four different prediction tasks in which regularization is particularly important.

First, we follow the experiment setup in Bietti et al. (2019) and evaluate the generalization performance of the proposed function-space regularization technique 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 over-fitting on the training data. Second, we evaluate the effectiveness of the proposed method on highly-overparameterized neural networks. In particular, we consider the MLP-Mixer (Tolstikhin et al., 2021) model, which has recently drawn attention for its competitive performance on large-scale datasets with much higher throughput compared to convolutional networks or self-attention models. However, the MLP-Mixer model has been shown to be very prone to over-fitting (Tolstikhin et al., 2021). We use the proposed function-space regularization method to train the MLP-Mixer model and show that it alleviates over-fitting and leads to improved predictive performance. Third, we evaluate the proposed function-space regularization technique on the small-sample downstream task of protein homology detection. Fourth, we evaluate the proposed 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 detailed description of the datasets as well as training and evaluation protocols, see Appendix 4.[1]

In contrast to methods designed for related tasks, such as few-shot learning, which specialized techniques to extract relevant information from related tasks (Wang et al., 2020), we propose a general-purpose regularization technique. The experiments in our empirical evaluation are carefully tailored to illustrate the usefulness of the proposed method in finding models that generalize well without requiring task- or model-specific modifications to the training procedure.

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 $\mathcal{K}^{L_2}$, the Jacobian norm regularization technique proposed by Bietti et al. (2019) as $\mathcal{K}^{\mathcal{J}}$, weight decay as maximum a posteriori (MAP) estimation, and kernel regression under the NTK as $\mathcal{K}_{ntk}$.

1. Our code can be accessed at `https://github.com/hudsonchen/FS-REG`.



(a) Accuracy on MNIST

(b) Accuracy on CIFAR10

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

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 under-fitting 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 batch normalization by computing the normalization mean and standard deviation for any single evaluation point deterministically at every iteration from a fixed "conditioning set" $\mathbf{X}_{\text{cond}}$. For details, see Appendix 4.

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 a ResNet-18 architecture, even under the Nyström approximation. Figures 1(a) and 1(b) 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 the training set and higher test accuracy indicates better a better ability to generalize. We can see that our approach, $\mathcal{K}^{\Theta}$, generalizes best when the number of training samples is small, but MAP gradually catches up as the number of training samples grows.

Moreover, we find that our method also outperforms $\mathcal{K}_{ntk}$, which may be due to the expressiveness of neural networks. Lastly, we observe that as the number of training samples grows, $\mathcal{K}^{\Theta}$ loses its advantage vis-a-vis other methods, since the large number of training samples makes the ability to generalize well from a small set of training points less important. We also note that in Figure 1(b), our approach $\mathcal{K}^{\Theta}$ is more effective when there is no data augmentation. This may be due to the fact that data augmentation aids neural networks in generalizing well and as such may offset the positive regularization effect of $\mathcal{K}^{\Theta}$.

5.2. Reducing over-fitting in Highly Overparameterized Models: MLP-Mixer

Based on the practical techniques discussed in Section 3.1, 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 pre-trained on ImageNet (Deng et al., 2009) and then train until convergence using the proposed function-space regularization method.

In this experiment, we observe that function-space regularization results in a higher level of predictive accuracy than maximum likelihood training from a pre-trained model without function-space regularization (see Table 1). The improvement in predictive accuracy is statistically significant on both datasets, demonstrating that the proposed regularization technique is effective in reducing over-fitting in highly over-parameterized neural networks.

Table 1: MLP-Mixer performance on CIFAR-10 and CIFAR-100. Reported mean and standard errors are computed over ten random seeds. [1]Negative log-likelihood. [2]Regularization used in Tolstikhin et al. (2021).

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

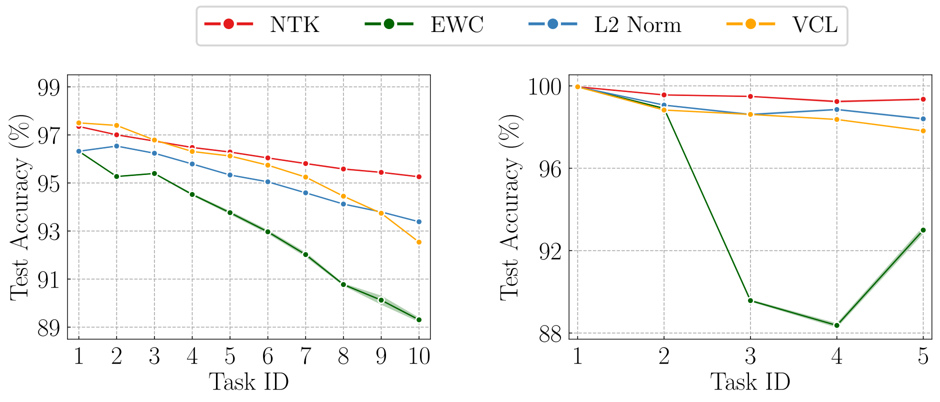
5.3. Improving Generalization Across Datasets: Protein Homology Detection

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 4.4, we get 100 balanced binary classification tasks with 200 protein sequences each. We also cut the length of the 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). We use a convolutional neural network with 3 convolutional layers followed by a fully-connected layer.

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
$\|\theta\|_2$	$0.55_{\pm 0.02}$
$\mathcal{K}^{\mathcal{J}}$	$0.59_{\pm 0.09}$
$\mathcal{K}^{L_2}$	$0.58_{\pm 0.04}$
Dropout	$0.55_{\pm 0.02}$
$\|f\|_\delta^2$ [1]	0.61
grad-ℓ_2 [1]	0.57
Ours	$\mathbf{0.62}_{\pm 0.05}$

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 on the small-data protein homology detection task.



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

Figure 2: Predictive performance of different continual learning methods and the proposed function-space regularization technique (denoted 'NTK') on Permuted MNIST (single-head) and Split MNIST (multi-head). Function-space regularization outperforms other methods.

5.4. 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 each of S sequential datasets by $\mathcal{D}_s = \{\mathbf{x}_s^{(n)}, \mathbf{y}_s^{(n)}\}_{n=1}^{N_s}$ with $s = 0, 1, \ldots, S-1$. When learning to solve a given task, a model is assumed not to have 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, the corresponding induced functions, and the induced NTK at task s by $\boldsymbol{\theta}_s$, $f_{\boldsymbol{\theta}_s}(\cdot)$, Θ_s, respectively.

At task $s = 0$, we use maximum likelihood estimation without 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 between the function learned at task $s-1$ and the function to be learned at task s using the proposed function-space regularization technique. In Benjamin et al. (2018), this distance is trivially measured by the L_2 norm as $\|f_s - f_{s-1}\|_{L_2}$. As noted in the discussion of the function norm associated with a network's NTK in Section 2, the L_2 function space may be unable to capture important topological information of the data-generating process. To capture such information, we instead use the function-space regularization method proposed in Section 3 and compute the function norm of $f_s - f_{s-1}$ following Equation (11), 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 function for continual learning at task s can then be expressed as

$$\mathcal{L}_s^{\hat{\mathcal{R}}}(\boldsymbol{\theta}_s) = \sum_{n=1}^{N_s} \log p(\mathbf{y}^{(n)} | f_{\boldsymbol{\theta}_s}(\mathbf{x}^{(n)})) + \zeta \Delta f(C_s)^\top \Theta_{s-1}^{-1}(C_s, C_s) \Delta f(C_s), \tag{14}$$

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.

We evaluate this approach on single-head permuted-MNIST and multi-head split-MNIST, which are standard continual learning benchmarks. Following prior works, we use a two-layer fully-connected neural network with 100 hidden units per layer for permuted MNIST and a 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 2(*a*) and 2(*b*) 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 regularization methods, our method, $\mathcal{K}^{\Theta}$, achieves the highest average accuracy. We attribute this result to the effective regularization of the relative changes in the predictive functions.

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%±0.3	63.1%±0.2
SI	86.0%	98.9%
VCL	92.5%±0.1	97.8%±0.0
FROMP	94.9%±0.0	99.0%±0.0
FRCL	94.3%±0.0	97.8%±0.2
L_2	93.5%±0.0	98.4%±0.1
Ours	**95.2%**±0.0	**99.3%**±0.0

5.5. Sensitivity Study on Function Norm Kernel Approximations

In order to assess the accuracy of the approximations described in Section 3, we conducted a sensitivity study on how well the norm under the empirical NTK, Θ, evaluated at a context set $\mathbf{X}_{\mathcal{C}}$ approximates the norm under the analytic NTK, Θ_{ntk}. Specially, in Figure 4, the red line represents the norm

$$\|f\|^2_{\mathcal{H}_\Theta} = f_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta^{-1}_{ntk}(\mathbf{X}, \mathbf{X}) f_{\boldsymbol{\theta}}(\mathbf{X}), \tag{15}$$

that is, the function norm used in Theorems 1 and 2. The blue line in Figure 4 represents

$$\mathop{\mathbb{E}}_{\mathbf{X}_{\mathcal{C}} \sim p_{\mathbf{X}}} \left[f_{\boldsymbol{\theta}}\left(\mathbf{X}_{\mathcal{C}}\right)^\top \Theta^{-1}\left(\mathbf{X}_{\mathcal{C}}, \mathbf{X}_{\mathcal{C}}\right) f_{\boldsymbol{\theta}}\left(\mathbf{X}_{\mathcal{C}}\right)\right], \tag{16}$$

the norm under the approximate kernel used in the empirical evaluations (see Equation (13)). The norms are computed using a LeNet-style neural network and a sample of 1,000 MNIST training points is used to compute the analytic NTK Θ_{ntk} and invert the full matrix. Figure 4 shows that the approximate function norm is close to the function norm under the analytical NTK when $|\mathbf{X}_{\mathcal{C}}| \geq 10$. $|\mathbf{X}_{\mathcal{C}}| = 10$ was used for the empirical evaluations presented in the previous section.

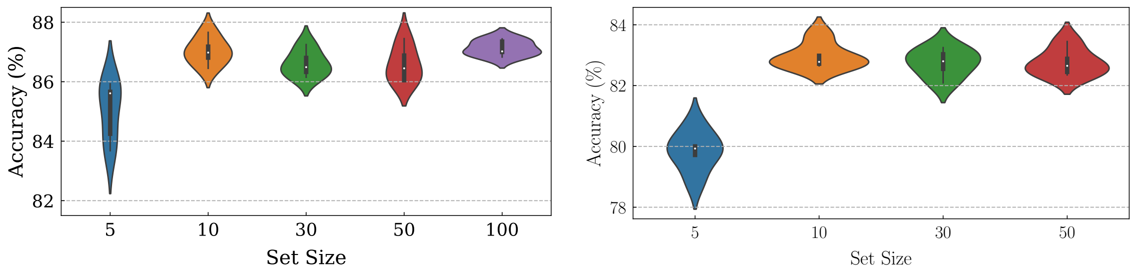


Figure 3: Top: Violin plots showing the effect of $|\mathbf{X}_{\mathcal{C}}|$ on predictive performance when training on MNIST using 50,000 samples. **Bottom:** Violin plots showing the effect of $|\mathbf{X}_{\mathcal{C}}|$ on predictive performance when training on CIFAR-10 using 10,000 samples. The plots were generated from ten random seeds.

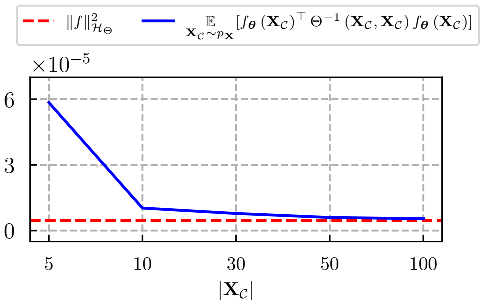


Figure 4: Ablation study on the approximations of the function norm practically used (blue line) to the theoretical function norm (red line) under different context set sizes, $|\mathbf{X}_{\mathcal{C}}|$.

Additionally, we carried out a sensitivity study on the effect of the context set size $|\mathbf{X}_{\mathcal{C}}|$ on the final predictive performance on larger datasets and larger neural network architectures. In Figure 3, we report the predictive performance on MNIST with 50,000 training samples and on CIFAR-10 with 10,000 training samples while varying the size of the context set. We observe that as long as the context set size is chosen to be above a certain threshold, that is, $|\mathbf{X}_{\mathcal{C}}| > 0$, predictive performance does not increase further as the size of the context set increases. This result is consistent with the results in Figure 4.

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 the functions in the RKHS associated with a neural 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 the scalability of the proposed method. This work provides further evidence in support of the hypothesis that carefully designed function-space regularizers improve generalization, and we hope that the proposed approach will inspire further research into general-purpose function-space regularization methods for deep learning as well as into tailored function-space regularization methods for challenging downstream tasks.

Acknowledgments

Tim G. J. Rudner is funded by the Rhodes Trust and the Engineering and Physical Sciences Research Council (EPSRC).

References

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

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers, 2020.

Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

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

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8141–8150, 2019.

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

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*, 2018.

Ari Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. In *International Conference on Learning Representations*, 2018.

Alberto Bietti and Julien Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations, 2018.

Alberto Bietti, Grégoire Mialon, Dexiong Chen, and Julien Mairal. A kernel perspective for regularizing deep neural networks. In *International Conference on Machine Learning*, pages 664–674. PMLR, 2019.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Apr 2017. ISSN 1537-274X. doi: 10.1080/01621459.2017.1285773.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

David R Burt, Sebastian W Ober, Adrià Garriga-Alonso, and Mark van der Wilk. Understanding variational inference in function-space. *arXiv preprint arXiv:2011.09421*, 2020.

John B Conway. *A course in functional analysis*, volume 96. Springer, 2019.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. New generalization bounds for learning kernels. *arXiv preprint arXiv:0912.3309*, 2009.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

Sebastian Farquhar, Michael A Osborne, and Yarin Gal. 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, 2020.

Amnon Geifman, Abhay Yadav, Yoni Kasten, Meirav Galun, David Jacobs, and Basri Ronen. On the similarity between the laplace and neural tangent kernels. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1451–1461. Curran Associates, Inc., 2020.

Tony Håndstad, Arne JH Hestnes, and Pål Sætrom. Motif kernel generated by genetic programming improves remote homology and fold detection. *BMC bioinformatics*, 8(1): 1–16, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Mohammad Emtiyaz E Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into gaussian processes. *Advances in neural information processing systems*, 32, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks, 2017.

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8572–8583. Curran Associates, Inc., 2019.

Chao Ma and José Miguel Hernández-Lobato. Functional variational inference based on stochastic process generators. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21795–21807. Curran Associates, Inc., 2021.

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

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

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.

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

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

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

Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. *Advances in Neural Information Processing Systems*, 33:4453–4464, 2020.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3. Citeseer, 2007.

Michael Reed and Barry Simon. *Methods of modern mathematical physics*, volume 1. Elsevier, 1972.

Tim G. J. Rudner, Freddie Bickford Smith, Qixuan Feng, Yee Whye Teh, and Yarin Gal. Continual Learning via Function-Space Variational Inference. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2022a.

Tim G. J. Rudner, Zonghao Chen, Yee Whye Teh, and Yarin Gal. Tractabe Function-Space Variational Inference in Bayesian Neural Networks. In *Advances in Neural Information Processing Systems 35*, 2022b.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001.

Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. In *International Conference on Learning Representations*, 2018.

Michalis K Titsias, Jonathan Schwarz, Alexander G de G Matthews, Razvan Pascanu, and Yee Whye Teh. Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*, 2019.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34, 2021.

Amal Rannen Triki, Maxim Berman, and Matthew B. Blaschko. Function norms and regularization in deep networks, 2018.

Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

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

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

Christopher KI Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.

Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. S2-mlp: Spatial-shift mlp architecture for vision. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 297–306, 2022.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Supplementary Material

Table of Contents

1 Reproducing Kernel Hilbert Space Theory **20**

2 Proofs and Derivations **21**
2.1 Discrepancy Between Predictive Functions 21
2.2 Generalization Error Bound 25

3 Ablation Studies **26**
3.1 Empirical Evaluation of Time Complexity 26
3.2 Batch Normalization in NTK Computation 26

4 Implementation, Training, and Evaluation Details **26**
4.1 MNIST and CIFAR-10 26
4.2 MLP-Mixer 27
4.3 Continual Learning 27
4.4 Protein Homology Detection 27

1. 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:

$$T_K(f)(\cdot) = \int_{\mathcal{X}} K(\cdot, \mathbf{x}) f(\mathbf{x}) \mathrm{d}\mu(\mathbf{x}). \tag{1.1}$$

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 $\sigma_0 \ge \sigma_1 \ge \ldots$, $\sigma_j \to 0$ as $j \to \infty$. Furthermore, Mercer's theorem (Mercer, 1909) states that K admits a decomposition as follows:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{j=0}^{\infty} \sigma_j \phi_j(\mathbf{x}) \phi_j(\mathbf{x}'). \tag{1.2}$$

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

$$\mathcal{H}_K = \left\{ f(\cdot) = \sum_{j=1}^{\infty} \alpha_j K(\mathbf{x}^{(j)}, \cdot) : (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots) \subset \mathcal{X} \wedge \|f\|_{\mathcal{H}_K}^2 = \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) < \infty \right\}. \tag{1.3}$$

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

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathcal{H}_K}. \tag{1.4}$$

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

$$\mathcal{H}_K = \left\{ f \in L_2(\mathcal{X}, \mu) : \|f\|_{\mathcal{H}_K}^2 = \sum_{j=0}^{\infty} \frac{\langle f, \phi_j \rangle_{L_2}^2}{\sigma_j} < \infty \right\}. \tag{1.5}$$

We make the following observations:

1. As we can see from Equation (1.3), 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 that Equation (3) is not true for general f. But as a result of the 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 Equation (3) is valid for kernel regression.
2. By Equation (1.5), we can easily see that $\|f\|_{\mathcal{H}_K}^2 \ge c \sum_{j=0}^{\infty} \langle f, \phi_j \rangle_{L_2}^2 = c \|f\|_{L_2}^2$ where $c = \sigma_0^{-1}$. Therefore, the RKHS norm provides more effective regularization than the L_2 norm.

2. Proofs and Derivations

2.1. Discrepancy Between Predictive Functions

The proof of Theorem 1 requires a study of neural network training dynamics, and Theorem 2 can be proved by applying the classical kernel ridge regression bound. We start by recalling the definition of fully-connected neural networks:

$$\begin{aligned} f^{(h)}(\mathbf{x}) &= \mathbf{W}^{(h)} g^{(h-1)}(\mathbf{x}) \in \mathbb{R}^{d_h} \quad g^{(0)}(\mathbf{x}) = \mathbf{x} \\ g^{(h)}(\mathbf{x}) &= \sqrt{\frac{c_\sigma}{d_h}} \sigma\big(f^{(h)}(\mathbf{x})\big) \quad \forall h = 1, 2, \ldots, L. \end{aligned} \tag{2.6}$$

And the last layer is given by

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = f^{(L+1)}(\mathbf{x}) = \mathbf{W}^{(L+1)} g^{(L)}(\mathbf{x}). \tag{2.7}$$

The optimization problem we propose is

$$\min_{\boldsymbol{\theta}} \mathcal{L}^{\hat{\mathcal{R}}} = \min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{j=1}^{N} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}^{(j)})) + \zeta f_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}). \tag{2.8}$$

Training Dynamics under Function-Space Regularization. We first study the dynamics of the optimization problem stated in Equation (2.8). Suppose the loss function is the squared loss $\ell(f) = (f(x) - y)^2$. Then the continuous gradient descent training dynamics are given by

$$\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}^{\hat{\mathcal{R}}}}{\mathrm{d}\boldsymbol{\theta}} = -\eta \left(\frac{2}{N} \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{X})^\top (f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{X})^\top \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}) \right) \tag{2.9}$$

and

$$\frac{\mathrm{d}f_{\boldsymbol{\theta}}}{\mathrm{d}t}(\mathbf{X}) = \mathcal{J}_{\boldsymbol{\theta}}(\mathbf{X}) \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} = -\eta \left(\frac{2}{N} \Theta(\mathbf{X}, \mathbf{X})(f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta \Theta(\mathbf{X}, \mathbf{X}) \Theta_{ntk}(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}) \right). \tag{2.10}$$

Note that, in practice, we use the empirical kernel Θ instead of the analytic NTK, and the inverse kernel matrix is treated as constant in the network parameters at each gradient update step. The training dynamics under the empirical NTK are therefore given by

$$\begin{aligned} \frac{\mathrm{d}f_{\boldsymbol{\theta}}}{\mathrm{d}t}(\mathbf{X}) &= -\eta \left(\frac{2}{N} \Theta(\mathbf{X}, \mathbf{X})(f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta \Theta(\mathbf{X}, \mathbf{X}) \Theta(\mathbf{X}, \mathbf{X})^{-1} f_{\boldsymbol{\theta}}(\mathbf{X}) \right) \\ &= -\eta \left(\frac{2}{N} \Theta(\mathbf{X}, \mathbf{X})(f_{\boldsymbol{\theta}}(\mathbf{X}) - \mathbf{y}) + 2\zeta f_{\boldsymbol{\theta}}(\mathbf{X}) \right) \end{aligned}. \tag{2.11}$$

First, we prove that the two training dynamics are close and that the proposed practical algorithm, therefore, approximates the true solution well.

Lemma 1 *Let f^*_{ntk} and f_{entk} be the solutions of Equation (2.10) and Equation (2.11), respectively, with the same initial condition. Assume f is bounded, then for ultra-wide networks, we have*

$$\sup_{t \geq 0} \| f^*_{ntk}(\mathbf{X}) - f_{entk}(\mathbf{X}) \| < \gamma \varepsilon. \tag{2.12}$$

Proof By comparing the dynamics in Equation (2.10) and Equation (2.11), we have

$$\begin{aligned}
&-\frac{1}{\eta}\left[\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{\theta}}-\frac{\mathrm{d}g}{\mathrm{d}\boldsymbol{\theta}}\right]\\
&=\frac{2}{N}\Theta_f(f_{\boldsymbol{\theta}}(\mathbf{X})-\mathbf{y})-\frac{2}{N}\Theta_g(g_{\boldsymbol{\theta}}(\mathbf{X})-\mathbf{y})+2\zeta\Theta_f\Theta_{ntk}^{-1}f_{\boldsymbol{\theta}}(\mathbf{X})-2\zeta g_{\boldsymbol{\theta}}(\mathbf{X})\\
&=\frac{1}{N}\big(2\Theta_f+2\zeta\Theta_f\Theta_{ntk}^{-1}\big)(f_{\boldsymbol{\theta}}(\mathbf{X})-g_{\boldsymbol{\theta}}(\mathbf{X}))+\big(\Theta_f-\Theta_g\big)\left(\frac{2}{N}(g_{\boldsymbol{\theta}}(\mathbf{X})-\mathbf{y})+2\zeta\Theta_{ntk}^{-1}g_{\boldsymbol{\theta}}(\mathbf{X})\right).
\end{aligned} \tag{2.13}$$

We denote the empirical NTK matrix with respect to f by $\Theta_f=\Theta_f(\mathbf{X},\mathbf{X})$ (and analogously for g). 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 smallest eigenvalue of Θ_{ntk}. Since $h(0)=0$, we have

$$f_{ntk}^*(\mathbf{X})-f_{entk}(\mathbf{X})=h(t)=-\eta e^{-\eta\int_0^t\mathbf{A}\mathrm{d}t}\int_0^t e^{\int_0^s\eta\mathbf{A}\mathrm{d}t}\boldsymbol{v}(s)\mathrm{d}s \tag{2.14}$$

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

$$\|\Theta_f-\Theta_g\|\le\|\Theta_f-\Theta_g\|_F\le\|\Theta_f-\Theta_{ntk}\|_F+\|\Theta_g-\Theta_{ntk}\|_F\le 2N(L+1)\varepsilon \tag{2.15}$$

So we have

$$\begin{aligned}
\|\boldsymbol{v}\|\le&\|\Theta_f-\Theta_g\|\left(\frac{2}{N}\|g_{\boldsymbol{\theta}}(\mathbf{X})-\mathbf{y}\|+2\zeta\left\|\Theta_{ntk}^{-1}g_{\boldsymbol{\theta}}(\mathbf{X})\right\|\right)\\
\le&4N(L+1)\varepsilon\left(C_1+\zeta\frac{NC_0}{\lambda_0}\right):=C''\varepsilon.
\end{aligned} \tag{2.16}$$

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

$$\|\mathbf{A}-\mathbf{A}_0\|\le\frac{1}{N}(2\|\Theta_f-\Theta_{ntk}\|+2\zeta\left\|\Theta_{ntk}^{-1}\right\|\|\Theta_f-\Theta_{ntk}\|)\le 2(L+1)\varepsilon\left(1+\frac{\zeta}{\lambda_0}\right). \tag{2.17}$$

Hence, by Lemma 2, we have

$$\begin{aligned}
\lambda_{\min}(\int_s^t\mathbf{A}\mathrm{d}s)\ge&\lambda_{\min}(\int_s^t\mathbf{A}_0\mathrm{d}s)-\left\|\int_s^t(\mathbf{A}-\mathbf{A}_0)\mathrm{d}s\right\|\\
\ge&\left[\frac{2}{N}(\lambda_0+\zeta)-2(L+1)\varepsilon\left(1+\frac{\zeta}{\lambda_0}\right)\right](t-s):=C'(t-s).
\end{aligned} \tag{2.18}$$

Hence we see that

$$\begin{aligned}
\|h(t)\|&\le\eta\int_0^t\left\|e^{\int_t^s\eta\mathbf{A}\mathrm{d}t}\right\|\|\boldsymbol{v}\|\mathrm{d}s\\
&\le\eta C''\varepsilon\int_0^t e^{-\eta\lambda_{\min}(\int_s^t\mathbf{A}\mathrm{d}t)}\mathrm{d}s\\
&\le\eta C''\varepsilon\int_0^t e^{\eta C'(s-t)}\mathrm{d}s\\
&\le\frac{\eta C''\varepsilon}{\eta C'}(1-e^{-\eta C't})\le\frac{C''}{C'}\varepsilon.
\end{aligned} \tag{2.19}$$

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 $\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A)$ (and for B as well). Then for all $1 \leq i \leq n$, we have*

$$|\lambda_i(A+B) - \lambda_i(A)| \leq \|B\|. \tag{2.20}$$

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

$$\lambda_i(A) = \sup_{\dim V = i} \inf_{v \in V, \|v\|=1} v^* A v \tag{2.21}$$

$$\lambda_i(A) = \inf_{\dim V = n-i+1} \sup_{v \in V, \|v\|=1} v^* A v. \tag{2.22}$$

According to Equation (2.21), 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 (2.22), 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

$$\lambda_i(A+B) - \lambda_i(A) \leq v^*(A+B)v - v^* A v = v^* B v \leq \|B\|. \tag{2.23}$$

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

$$|f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) - f_{\infty}^{\mathcal{R}*}(\mathbf{x})| \leq \varepsilon. \tag{2.24}$$

Proof Note that the training dynamics 2.11 can be written as

$$\frac{\mathrm{d}f_{\boldsymbol{\theta}}}{\mathrm{d}t} + \frac{2\eta}{N}(\Theta + \zeta N \mathbf{I}) f_{\boldsymbol{\theta}} = \frac{2\eta}{N}\Theta \mathbf{y}. \tag{2.25}$$

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

$$\frac{\mathrm{d}f_{\boldsymbol{\theta}}}{\mathrm{d}t} + \frac{2\eta}{N}(\Theta_{ntk} + \zeta N \mathbf{I}) f_{\boldsymbol{\theta}} = \frac{2\eta}{N}\Theta_{ntk} \mathbf{y}. \tag{2.26}$$

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

$$f_t(\mathbf{x}) = \Theta_{ntk}(\mathbf{x}, \mathbf{X})\big(\Theta_{ntk}(\mathbf{X}, \mathbf{X}) + \zeta N \mathbf{I}\big)^{-1}\left(I - e^{-\frac{2\eta}{N} t \Theta_{ntk}(\mathbf{X}, \mathbf{X})}\right)\mathbf{y}. \tag{2.27}$$

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

$$\frac{\mathrm{d}h}{\mathrm{d}t} + \frac{2\eta}{N}(\Theta_{ntk} + \zeta N \mathbf{I}) h = \frac{2\eta}{N}(\Theta - \Theta_{ntk})(\mathbf{y} - f). \tag{2.28}$$

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

$$f_{entk}(\mathbf{X}) - f_{\infty}^{\mathcal{R}*}(\mathbf{X}) = h(t) = e^{-\int_0^t \mathbf{B} \mathrm{d}t} \int_0^t e^{\int_0^s \mathbf{B} \mathrm{d}t} \mathbf{u}(s) \mathrm{d}s. \tag{2.29}$$

This implies

$$\begin{aligned} |h(t)| &\le \int_0^t \left\| e^{\int_s^t \mathbf{B} \mathrm{d}t} \right\| \|\mathbf{u}\| \, \mathrm{d}s \\ &\le \frac{2\eta}{N} \|\Theta - \Theta_{ntk}\| \, \|\mathbf{y} - f(\mathbf{X})\| \, \frac{1}{\|\mathbf{B}\|} \left(1 - e^{-t\|\mathbf{B}\|}\right) \le \frac{N C_0 \varepsilon}{\lambda_0 + \zeta N}, \end{aligned} \tag{2.30}$$

where C_0 is constant bounds the function value and λ_0 is the smallest eigenvalue of the NTK. Equation (2.30) together with Theorem 1 prove the Theorem on the 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 3. Although we propose to use function norm under the analytic NTK Θ_{ntk} as a regularizer in Equation (11), this exact computation is costly in practice, so the empirical kernel is used instead.

Specifically, in practice, we replace Θ_{ntk} in Equation (2.8) 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 Equation (2.10) and Equation (2.11). 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.

2.2. Generalization Error Bound

Theorem 2 *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 $\|\mathbf{x}\| \leq 1$, $|f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x})| \leq C_0\kappa$ 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:*

$$R(f_{nn}^{\hat{\mathcal{R}}*}) - \hat{R}(f_{nn}^{\hat{\mathcal{R}}*}) \leq 4(C_0\kappa + c)\varepsilon + \frac{8C_1^2\Lambda^2}{\sqrt{N}}\left(\sqrt{\frac{C_K}{NC_1^2}} + \frac{3}{4}\sqrt{\frac{\log\frac{2}{\delta}}{2}}\right), \tag{2.31}$$

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 1.

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 $\|\mathbf{x}\| \leq 1$ for all $\mathbf{x} \in \mathcal{X}$.

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

$$R(f_{nn}^{\hat{\mathcal{R}}*}) - R(f_{\infty}^{\mathcal{R}*}) = \mathbb{E}\big[\big(f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) - f_{\infty}^{\mathcal{R}*}(\mathbf{x})\big)\big(f_{nn}^{\hat{\mathcal{R}}*}(\mathbf{x}) + f_{\infty}^{\mathcal{R}*}(\mathbf{x}) - 2\mathbf{y}(\mathbf{x})\big)\big] \leq 2\varepsilon(C_0\kappa + c), \tag{2.32}$$

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

$$\hat{R}(f_{\infty}^{\mathcal{R}*}) - \hat{R}(f_{nn}^{\hat{\mathcal{R}}*}) \leq 2\varepsilon(C_0\kappa + c) \tag{2.33}$$

Therefore,

$$R(f_{nn}^{\hat{\mathcal{R}}*}) - \hat{R}(f_{nn}^{\hat{\mathcal{R}}*}) \leq R(f_{\infty}^{\mathcal{R}*}) - \hat{R}(f_{\infty}^{\mathcal{R}*}) + 4\varepsilon(C_0\kappa + c). \tag{2.34}$$

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

$$R(f_{\infty}^{\mathcal{R}*}) - \hat{R}(f_{\infty}^{\mathcal{R}*}) \leq \frac{8r^2\Lambda^2}{\sqrt{N}}\left(\sqrt{\frac{\mathrm{Tr}(\Theta_{\mathrm{ntk}})}{Nr^2}} + \frac{3}{4}\sqrt{\frac{\log\frac{2}{\delta}}{2}}\right), \tag{2.35}$$

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

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

3. Ablation Studies

3.1. Empirical Evaluation of Time Complexity

We provide a comparison of 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 a 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

3.2. 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 $\mathbf{X}_{\mathcal{B}}$.

4. 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 hyperparameters that yielded the lowest validation negative log-likelihood for all experiments. All experiments are implemented in JAX (Bradbury et al., 2018).

4.1. 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 every 10 epochs. During training, the 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 held-out 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.

4.2. MLP-Mixer

We use MLP-Mixer of B/16 architecture (Tolstikhin et al., 2021) and load the pre-trained parameters on ImageNet from a public online source. The training of MLP-Mixer on CIFAR-100 requires 8 GTX 3090 GPUs. Stochastic gradient descent with a learning rate of 0.01 and momentum with a decay rate of 0.9 are used for optimization.

4.3. 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 inputs 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 are 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 the 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 a batch size of 128 in all experiments.

4.4. Protein Homology Detection

The protein homology detection experiment used the dataset of Structural Classification Of Proteins (SCOP) version 1.67 (Murzin 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 does 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.