On Kernel Method-Based Connectionist Models and Supervised Deep Learning Without Backpropagation

Shiyu Duan[1]

Shujian Yu[1]

Yunmei Chen[2]

Jose C. Principe[1]

[1]Department of Electrical and Computer Engineering, University of Florida

[2]Department of Mathematics, University of Florida

Keywords: Kernel method, connectionist model, supervised learning, deep learning

Abstract

We propose a novel family of connectionist models based on kernel machines and consider the problem of learning layer-by-layer a compositional hypothesis class, i.e., a feedforward, multilayer architecture, in a supervised setting. In terms of the models, we present a principled method to "kernelize" (partly or completely) any neural network

(NN). With this method, we obtain a counterpart of any given NN that is powered by kernel machines instead of neurons. In terms of learning, when learning a feedforward deep architecture in a supervised setting, one needs to train all the components simultaneously using backpropagation (BP) since there are no explicit targets for the hidden layers (Rumelhart et al., 1986). We consider without loss of generality the two-layer case and present a general framework that explicitly characterizes a target for the hidden layer that is optimal for minimizing the objective function of the network. This characterization then makes possible a purely greedy training scheme that learns one layer at a time, starting from the input layer. We provide realizations of the abstract framework under certain architectures and objective functions. Based on these realizations, we present a layer-wise training algorithm for an l-layer feedforward network for classification, where $l \geq 2$ can be arbitrary. This algorithm can be given an intuitive geometric interpretation that makes the learning dynamics transparent. Empirical results are provided to complement our theory. We show that the kernelized networks, trained layer-wise, compare favorably with classical kernel machines as well as other connectionist models trained by BP. We also visualize the inner workings of the greedy kernelized models to validate our claim on the transparency of the layer-wise algorithm.

1 Introduction

One can "kernelize" any neural network (NN) by replacing each artificial neuron (McCulloch & Pitts, 1943), i.e., function approximator of the form $f(\mathbf{x}) = \sigma\big(\mathbf{w}^\top \mathbf{x} + b\big)$, with a kernel machine, i.e., function approximator of the form $f(\mathbf{x}) = \langle \mathbf{w},\, \phi(\mathbf{x}) \rangle_H + b$

with kernel function $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_H$. While the nonlinearities in deep NNs make it notoriously difficult to analyze these models, the simple interpretation of a kernel machine as a hyperplane in a reproducing kernel Hilbert space (RKHS) makes the kernelized networks more tractable mathematically. We shall refer to the kernelized NNs in general as kernel networks (KNs).

We then revisit the problem of learning a composite hypothesis class, by which we mean a trainable model that consists of more elementary trainable submodels, in a supervised learning setting. In this paper, we shall only consider the special case of a compositional hypothesis class, in which the elementary submodels are linked via function compositions and therefore the overall model can be written as $\mathbf{F} = \mathbf{F}_l \circ \cdots \circ \mathbf{F}_1$ for some l, with each $\mathbf{F}_i$ being a submodel with proper domain and codomain. For example, a deep, feedforward NN can be considered as a compositional hypothesis class.

When it comes to training these models, the usual method is to learn all its trainable submodels simultaneously using, for example, backpropagation (BP) (Rumelhart et al., 1986). However, in the context of supervised learning, the need for BP is caused by the fact that there is no explicit target information to tune the latent submodels (Rumelhart et al., 1986). Moreover, when the model is large, BP usually becomes computationally intensive and can suffer from issues such as vanishing gradient. Also, BP returns very little information on the training of each submodel to the user and therefore forces the user to treat the model as a "black box". For example, it is usually not possible to know which specific part or parts of the network is responsible when the performance is suboptimal. Also, it is extremely difficult to interpret or assess the hidden representations during or after training.

We consider the problem of reducing the compositional learning problem into a set of noncompositional ones and then solving each one of them individually. We approach by deriving explicit targets for the hidden submodels. The targets are optimal for minimizing a given objective function of the overall model. The central idea can be summarized as follows: let input data $S_{\mathbf{X}}$, supervision S_Y (labels in classification, dependent variable in regression), a two-layer feedforward architecture $\mathbf{F}_2 \circ \mathbf{F}_1$, and an objective function $\tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1(S_{\mathbf{X}}),\, S_Y)$ be given, define $\mathbf{F}_2^\star \circ \mathbf{F}_1^\star \coloneqq \arg\min_{\mathbf{F}_2 \circ \mathbf{F}_1} \tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1(S_{\mathbf{X}}),\, S_Y)$. If we could find functions $s,\, u$, and a new objective $\tilde{R}_1(s(\mathbf{F}_1(S_{\mathbf{X}})),\, u(S_Y))$ whose minimizer is equivalent to $\mathbf{F}_1^\star$ for minimizing the objective $\tilde{R}$, then finding $\mathbf{F}_1^\star$ is equivalent to finding an $\mathbf{F}_1$ that minimizes $\tilde{R}_1$. If the dependence of s and u on $\mathbf{F}_2$ can be reduced to a point where this search for $\mathbf{F}_1^\star$ does not involve the trainable parameters of $\mathbf{F}_2$, then we have reduced the original compositional learning problem into two noncompositional ones that can be solved sequentially.

As examples, we provide realizations of the abstract framework and also, based on these realizations, a sample greedy training algorithm for a multilayer feedforward architecture for classification. This greedy learning algorithm enjoys the same optimality guarantee as BP in the sense that they both effectively train each layer to minimize the overall objective. But the former is faster, more memory efficient, and evidently less susceptible to vanishing gradient. It also greatly increases the transparency of deep models: the quality of learning in the hidden layers can be directly assessed during or after training, providing the user with more information about training. Also, alternative model selection and hyperparameter tuning paradigms are now available since unsatisfying performance of the network can be traced to a certain layer or layers,

allowing the user to "debug" the layers individually. Moreover, the target for each hidden layer in this algorithm can be given an intuitive geometric interpretation, making the learning dynamics transparent.

Empirical results are provided to complement our theory. First, we compare KNs with classical kernel machines and show that KNs consistently outperform Support Vector Machine (SVM) (Cortes & Vapnik, 1995) as well as several SVMs enhanced by Multiple Kernel Learning (MKL) algorithms (Bach et al., 2004; Gönen & Alpaydın, 2011). We then fully or partly kernelized both fully-connected and convolutional NNs and trained them with the proposed layer-wise algorithm. The resulting KNs compare favorably with their NN equivalents trained with BP as well as some other commonly-used deep architectures trained with BP together with unsupervised greedy pre-training. We also visualize the learning dynamics and hidden representations in the greedy kernelized networks to validate our claim on the transparency of the greedy algorithm.

2 Setting and Notations

We consider the following supervised set-up: let a realization of an i.i.d. random sample be given: $S = \{(\mathbf{x}_n,\, y_n)\}_{n=1}^N$, where $(\mathbf{x}_n,\, y_n) \in \mathbb{R}^{d_0} \times \mathbb{R}$. Denote $\{\mathbf{x}_n\}_{n=1}^N$ as $S_{\mathbf{X}}$ and $\{y_n\}_{n=1}^N$ as S_Y for convenience. We consider only real, continuous, symmetric, positive definite (PD) kernels (Schölkopf & Smola, 2001), which possess the reproducing property $k(\mathbf{x},\, \mathbf{y}) = \langle \phi(\mathbf{x}),\, \phi(\mathbf{y}) \rangle_H$, where H is the RKHS induced by k. Further, we assume, for all kernels considered in all results, that $k(\mathbf{x},\, \mathbf{x}) = c < +\infty,\, \forall \mathbf{x}$, and that $\inf_{\mathbf{x},\mathbf{y}} k(\mathbf{x},\, \mathbf{y}) = a > -\infty$. It is straightforward to check using Cauchy-Schwarz

inequality that the first condition implies $\max_{\mathbf{x},\mathbf{y}} k(\mathbf{x}, \mathbf{y}) = c$. Note that by construction of a PD kernel, we always have $a < c$.

For the rest of this paper, we shall use bold letters to denote vectors or vector-valued functions. For random elements, we use capital letter to denote the random element and lower-case letter a realization of it. Notations similar to the following will be used whenever convenient: for a general l-layer feedforward architecture $\mathbf{F}_l \circ \cdots \circ \mathbf{F}_1$ and for $i = 2, 3, \ldots, l$, $\mathbf{x} \in \mathbb{R}^{d_0}$, $\mathbf{F}_i(\mathbf{x}) \coloneqq \mathbf{F}_i \circ \cdots \circ \mathbf{F}_1(\mathbf{x})$. For any $\mathbf{F}$, the shorthand $\mathbf{F}(S_{\mathbf{X}})$ represents $\{\mathbf{F}(\mathbf{x}_n)\}_{n=1}^N$. And likewise for $\mathbf{F}(S_Y)$. When there is no confusion, we shall suppress the dependency of any loss function on the example for brevity, i.e., for a loss function ℓ, instead of writing $\ell(f(\mathbf{x}), y)$, we write $\ell(f)$.

Given a loss function $\ell(f(\mathbf{x}), y)$, we define the risk as $R(f) \coloneqq \mathbb{E}_{(\mathbf{X},Y)}\ell(f(\mathbf{X}), Y)$ and an objective function $\tilde{R}(f(S_{\mathbf{X}}), S_Y)$ to be a bound on the risk that is computable using the given data only. In this paper, we shall take any objective as given without rigorously justifying why it is a bound of some risk since that is not the purpose of this paper. Nevertheless, the objectives we use in this paper are fairly common and the corresponding justifications are routine. We make this distinction between risk and objective here as it will be needed in later discussions.

3 Kernelizing a Neural Network

Kernel machines are parametric models defined as $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle_H + b$ with kernel $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_H$ and $\mathbf{w}$, b being the learnable weights and ϕ being a map into the RKHS H. NNs are connectionist models defined by arbitrarily combining the

parametric base units defined as $f(\mathbf{x}) = \sigma\big(\mathbf{w}^\top \mathbf{x} + b\big)$ with $\mathbf{w}$, b being the learnable weights and σ a (usually nonlinear) gating function. These base units are sometimes called neurons.

While NNs are flexible models and have strong expressive power in practice, they are notoriously difficult to analyze due to each nonlinear neuron being a nontrivial function itself and the arbitrariness involved in the overall architecture design. Kernel machines, in comparison, are much more mathematically tractable since they are linear models in the feature space H, i.e., the f is linear in $\mathbf{w}$. This allows one to reduce otherwise abstract problems into geometric ones, making possible simpler and more intuitive solutions. However, their architectures are not as flexible and their practical performance in most cutting-edge machine learning applications has been unsatisfying (Bengio et al., 2013).

The question we consider is how to combine the idea of connectionism, which is central to NNs, with kernel machines and build families of models that are flexible, expressive, and at the same time, more mathematically tractable than NNs. We hope this will be a first step toward explaining why deep learning performs so well in the most challenging AI tasks.

In this section, we discuss how to kernelize an NN to build models that combine the best of both worlds. We first present the generic approach and then as an example, concretely define a fully-kernelized Multilayer Perceptron (MLP). To further shed light on the effect of kernelization on the expressive power of the original model, we give an analysis on the model complexity of a fully-kernelized MLP.

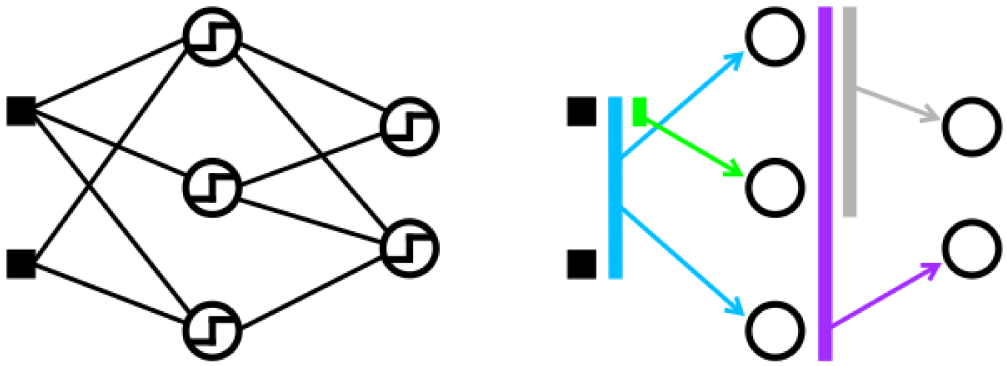


Figure 1: Any NN (left, presented in the usual weight-nonlinearity abstraction) can be abstracted as a "graph" (right) with each node representing a neuron and each edge the input-output relationship between neurons. If a node receives multiple inputs, we view its input as a vector in some Euclidean space, as indicated by the colored rectangles. Under this abstraction, each neuron can be directly replaced by a kernel machine mapping from the same Euclidean space into the real line without altering the architecture and functionality of the model.

3.1 A Generic Approach to Kernelization

The general idea we adopt is to build connectionist models with the base units being not neurons but kernel machines. This is mathematically viable since in an NN, any neuron can be directly replaced by a kernel machine without altering the architecture and functionality of the network. An illustration of this kernelization procedure is provided in Fig. 1. In this way, one can kernelize an NN to any degree: a node, several nodes, a layer, several layers, or the entire network.

KN is flexible in the sense that one can inject prior knowledge into the architecture design, as is done for NNs. KN inherits the expressive power of the original NN since a kernel machine is a universal function approximator under mild conditions (Park & Sandberg, 1991; Micchelli et al., 2006). Moreover, KN works in a more mathematically intuitive way since each base unit is a simple linear model in an RKHS.

Further, a general criticism toward kernel methods in machine learning is that their performance usually relies heavily on the parameterization of the kernels used. This issue is mitigated in KN, thanks to the introduction of connectionism. To be specific, KN performs nonparametric kernel learning alongside learning to perform the given task. Indeed, to build the network one only needs generic kernels, but in a connectionist model, the kernels on the non-input layers admit the form $k(\mathbf{F}(\mathbf{x}),\, \mathbf{G}(\mathbf{y}))$, where $\mathbf{F},\, \mathbf{G}$ are some other trainable submodels. The fact that $\mathbf{F},\, \mathbf{G}$ are trainable makes this kernel "adaptive", mitigating to some extent any limitation of the fixed generic kernel k. The training of $\mathbf{F}$ and $\mathbf{G}$ makes this adaptive kernel optimal as a constituent part of the corresponding kernel machine for the task the network was trained for. And it is always a valid kernel if the generic kernel k is. Note that observations similar to this one have

been made in different contexts by, for example, Huang & LeCun (2006) and Bengio et al. (2013), we include it here only for completeness.

3.2 Kernelized MLP: The Architecture

As a more concrete example, we now define a fully-kernelized l-layer MLP, which we will specifically refer to as kernel MLP (kMLP).[1]

The l-layer kMLP is defined as follows. For $i \geq 1$, the i^{th} layer in a kMLP, denoted $\mathbf{F}_i$, is an array of d_i kernel machines: $\mathbf{F}_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}, \mathbf{F}_i(\mathbf{x}) = \left(f_i^1(\mathbf{x}),\ f_i^2(\mathbf{x}),\ \ldots,\ f_i^{d_i}(\mathbf{x})\right)$ with the f_i^j all using kernel k_i. Let $\mathbf{F}_0$ be the identity map on $\mathbb{R}^{d_0}$, each $f_i^j : \mathbb{R}^{d_{i-1}} \to \mathbb{R}$ is a hyperplane in RKHS H_i: $f_i^j(\mathbf{x}) = \left\langle \mathbf{w}_{f_i^j},\ \phi_i(\mathbf{F}_{i-1} \circ \cdots \circ \mathbf{F}_0(\mathbf{x})) \right\rangle_{H_i} + b_{f_i^j}$, $\mathbf{w}_{f_i^j} \in H_i$, $b_{f_i^j} \in \mathbb{R}$. The set of mappings

$$\left\{ \mathbf{F}_l \circ \cdots \circ \mathbf{F}_1 :\ \mathbf{w}_{f_i^j} \in H_i,\ b_{f_i^j} \in \mathbb{R} \text{ for all admissible } n,\ j,\ i \right\}$$

defines an l-layer kMLP.

In practice, $\mathbf{w}_{f_i^j}$ is usually not accessible but can be approximated using, for instance, $\sum_{n=1}^{N} \alpha_{i,\,n}^{j} \phi_i(\mathbf{F}_{i-1} \circ \cdots \circ \mathbf{F}_0(\mathbf{x}_n))$, where the $\alpha_{i,\,n}^{j} \in \mathbb{R}$ are the learnable parameters.[2]

3.3 Kernelized MLP: Model Complexity

We give a bound on the model complexity of an l-layer kMLP using a well-known complexity measure called Gaussian complexity (Bartlett & Mendelson, 2002). In

[1] A PyTorch-based (Paszke et al., 2017) library for implementing KN and the proposed layer-wise training algorithm is available at: *https://github.com/michaelshiyu/kerNET*.

[2] The optimality of this expansion can be justified in the following layer-wise setting by directly applying the representer theorem (Schölkopf et al., 2001).

particular, the bound describes the relationship between the depth/width of the model and the complexity of its hypothesis class, providing insights into the effect of kernelization on the expressive power of the model as well as useful information for model selection. We first review the definition of Gaussian complexity.

Definition 3.1 (*Gaussian complexity*)**.** *Let $X_1, \ldots, X_N$ be i.i.d. random elements defined on metric space $\mathbb{X}$ and let $\mathbb{F}$ be a set of functions mapping from $\mathbb{X}$ into $\mathbb{R}$. Define*

$$\hat{\mathcal{G}}_N(\mathbb{F}) = \mathbb{E}\left[\sup_{f\in\mathbb{F}} \frac{1}{N}\sum_{n=1}^{N} Z_n f(X_i) \,\middle|\, X_1, \ldots, X_N\right],$$

where $Z_1, \ldots, Z_N$ are independent standard normal random variables. The Gaussian complexity of $\mathbb{F}$ is defined as $\mathcal{G}_N(\mathbb{F}) = \mathbb{E}\,\hat{\mathcal{G}}_N(\mathbb{F})$.

Intuitively, Gaussian complexity quantifies how well elements in a given function class can be correlated with a normally-distributed noise sequence of length N (Bartlett & Mendelson, 2002).

For Proposition 3.2 and the lemma based on which this proposition is proven (Lemma B.2 in Appendix B), we impose the following smoothness assumption on all kernels considered: for each fixed $\mathbf{x} \in \mathbb{R}^{d_{i-1}}$, we assume that $k_i(\mathbf{x}, \mathbf{y})$, as a function of $\mathbf{y}$, is $L_{i,\mathbf{x}}$-Lipschitz with respect to the Euclidean metric on $\mathbb{R}^{d_{i-1}}$. Let $\sup_{\mathbf{x}\in\mathbb{R}^{d_{i-1}}} L_{i,\mathbf{x}} = L_i$, which we assume to be finite.

Proposition 3.2. *Given an l-layer kMLP, approximate $\mathbf{w}_{f_i^j}$ using*

$$\sum_{\nu=1}^{m} \alpha_{i,\nu}^{j} \phi_i(\mathbf{F}_{i-1} \circ \cdots \circ \mathbf{F}_1(\mathbf{x}_\nu)),$$

where the $\mathbf{x}_\nu$ are an m-subset of $S_{\mathbf{X}}$, $1 \le m \le N$, $\boldsymbol{\alpha}_i^j \coloneqq \left(\alpha_{i,1}^j, \ldots, \alpha_{i,m}^j\right) \in \mathbb{R}^m$ and $b_{f_i^j} \in \mathbb{R}$. Assume $\left\|\boldsymbol{\alpha}_i^j\right\|_1 \le A_i$ and let $d_l = 1$. Consider

$$\mathbb{F}_1 = \left\{\mathbf{x} \mapsto \left(f_1^1(\mathbf{x}), \ldots, f_1^{d_1}(\mathbf{x})\right) \mid f_1^j \in \Omega,\, j = 1, \ldots, d_1\right\},$$

where Ω is a given hypothesis class of functions from $\mathbb{R}^{d_0}$ to $\mathbb{R}$. Denote the class of functions implemented by this kMLP as $\mathbb{F}_{l\text{-}kMLP}$, if $\mathbf{F}_1 \in \mathbb{F}_1$, for $i \geq 2$, we have

$$\mathcal{G}_N(\mathbb{F}_{l\text{-}kMLP}) \leq 2d_1 \prod_{i=2}^{l} A_i L_i d_i \mathcal{G}_N(\Omega).$$

It is worth noting that the model complexity kMLP grows in the depth and width of the network in a similar way as that of an MLP (Sun et al., 2016). In particular, the expressive power of the model increases linearly in the width of a given layer and roughly exponentially in the depth of the network.

4 A Layer-Wise Learning Framework

We now formally present our greedy framework for learning compositional hypothesis classes in a supervised setting. To simplify discussion, we first consider the two-layer case, i.e.,

$$\mathbb{F} = \{\mathbf{F} = \mathbf{F}_2 \circ \mathbf{F}_1 \,|\, \mathbf{F}_i \in \mathbb{F}_i,\, i = 1,\, 2\}.$$

Define $\mathbf{F}_1^\star \circ \mathbf{F}_2^\star = \mathbf{F}^\star \coloneqq \arg\min_{\mathbf{F}\in\mathbb{F}} \tilde{R}(\mathbf{F}(S_\mathbf{X}),\, S_Y)$. The goal is to learn the input layer to find $\mathbf{F}_1^\star$ (without touching the output layer), freeze the input layer afterwards, and then learn the output layer to find $\mathbf{F}_2^\star$.

To disentangle the learnings of the two layers, we must disentangle the definitions of $\mathbf{F}_1^\star$ and $\mathbf{F}_2^\star$. The idea is to re-characterize $\mathbf{F}_1^\star$, i.e., to derive conditions under which $\mathbf{F}_1 = \mathbf{F}_1^\star$, using no information on the trainable parameters of the output layer. Then, we need to translate these conditions into choosing a new loss ℓ_1 (inducing a new risk R_1), a function s, and a function u accordingly with the property that

$$\arg\min_{\mathbf{F}_1\in\mathbb{F}_1} R_1(s(\mathbf{F}_1(\mathbf{X})),\, u(Y)) = \mathbf{F}_1^\star$$

and that $s,\ u$ do not rely on the trainable parameters of the output layer. An objective $\tilde{R}_1(s(\mathbf{F}_1(S_\mathbf{X})),\ u(S_Y))$ can be subsequently chosen, and we can find $\mathbf{F}_1^\star$ by training the input layer to minimize this new objective. This training process requires no tuning on the output layer as this new objective does not involve the trainable parameters of it.

The re-characterization of $\mathbf{F}_1^\star$ is dependent on $\mathbf{F}_2$ and $\tilde{R}$. Therefore, different choices induce different realizations of our general framework.

The search of u can be understood as the procedure of explicitly backpropagating the targets S_Y to the hidden layers. This contrasts how learning is made possible in BP via backpropagating derivative information but not the targets directly.

We proceed by first describing the general framework and then, as examples, provide realizations under a specific choice of $\mathbb{F}_2$ and two families of objectives. Finally, based on these realizations, we provide a sample layer-wise training algorithm for learning an l-layer feedforward network for classification, where $l \geq 2$ can be arbitrary. This layer-wise algorithm is simple to implement and its learning dynamics enjoy an intuitive geometric interpretation.

4.1 The Framework

Let the architecture $\mathbb{F}$ and objective $\tilde{R}$ be given, our greedy learning framework for the two-layer compositional hypothesis class consists of the following steps:

1. Finding $\mathbf{F}_1^\star$

    (a) Define an equivalence relation between hypotheses.

    (b) Give an equivalent definition for $\mathbf{F}_1^\star$ under the new equivalence relation.

(c) Re-characterize $\mathbf{F}_1^\star$ for the given $\mathbb{F}$ under the given objective $\tilde{R}$.

(d) Choose $s,\ u,\ \ell_1,$ and $\tilde{R}_1$ accordingly.

(e) Train the input layer to minimize $\tilde{R}_1(s(\mathbf{F}_1(S_{\mathbf{X}})),\ u(S_Y))$.

(f) After training, freeze the input layer at, say, $\mathbf{F}_1^\circ$.

2. Finding $\mathbf{F}_2^\star$

(a) Train the output layer to minimize $\tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1^\circ(S_{\mathbf{X}}),\ S_Y)$.

We now provide more details for a couple of the listed steps.

Step 1a. Define an equivalence relation between hypotheses

In our framework, we use the following definition of equivalence between hypotheses of the input layer:

$$\mathbf{F}_1 = \mathbf{G}_1 \quad \text{if and only if} \quad \min_{\mathbf{F}_2 \in \mathbb{F}_2} \tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1) = \min_{\mathbf{F}_2 \in \mathbb{F}_2} \tilde{R}(\mathbf{F}_2 \circ \mathbf{G}_1),\ \forall S.$$

It is easy to check that this is indeed an equivalence relation. Intuitively, this means that we consider two hypotheses of the input layer to be equally good if the best networks one can build with these two hypotheses minimize the objective function equally well, i.e., when they have the same "potential". Evidently, this notion of equivalence is proper and sufficient as we have no knowledge of $\mathbf{F}_2$ while we train the input layer.

Step 1b. Give an equivalent definition for $\mathbf{F}_1^\star$ under the new equivalence relation

Compared to the original minimizer definition of $\mathbf{F}_1^\star$, it is easier to work with the following more concrete definition under the equivalence relation described in Step 1a.

Lemma 4.1. *Suppose* $\mathbf{F}_1^\star \in \mathbb{F}_1' \subseteq \mathbb{F}_1$ *and* $\mathbf{F}_2^\star \in \mathbb{F}_2' \subseteq \mathbb{F}_2$*, we have*

$$\mathbf{F}_1^\star = \arg\min_{\mathbf{F}_1 \in \mathbb{F}_1'} \min_{\mathbf{F}_2 \in \mathbb{F}_2'} \tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1).$$

This definition is easier to work with when we later re-characterize $\mathbf{F}_1^\star$ because it shrinks the range of $\mathbf{F}_2$ we need to consider for each $\mathbf{F}_1$ to only the minimizer $\mathbf{F}_2$ under that specific $\mathbf{F}_1$.

4.2 Some Realizations

We now provide realizations of the greedy learning framework under a specific family of $\mathbb{F}_2$ and two classes of objective functions. Note that these realizations are certainly not all that can be derived from our layer-wise framework. We leave the exploration of more such realizations as future work.

Steps 1a and 1b are the same for all realizations. Therefore, the only nontrivial steps in our framework to discuss for specific realizations are steps 1c and 1d.

The specific $\mathbb{F}_2$ we consider in this section is defined as the set of functions of the following form: $\mathbf{F}_2(\mathbf{x}) = (f_2^1(\mathbf{x}), \ldots, f_2^{d_2}(\mathbf{x}))$, $f_2^j(\mathbf{x}) = \left\langle \mathbf{w}_{f_2^j}, \phi(\mathbf{x}) \right\rangle_H + b_{f_2^j}$ with kernel $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_H$, $j = 1, \ldots, d_2$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{d_0}$, where we have omitted and will continue to omit writing out explicitly the function composition: for example, for $\mathbf{x} \in \mathbb{R}^{d_0}$, we write $\mathbf{F}_2(\mathbf{x})$ in place of $\mathbf{F}_2(\mathbf{F}_1(\mathbf{x}))$. There is no assumption needed on $\mathbb{F}_1$.

For these realizations, we consider the case $Y \in \{+1, -1\}$, and we shall use subscript $+$ or $-$ to indicate the class of a particular example, if needed.

Re-characterize $\mathbf{F}_1^\star$ under regularized hinge loss as objective

Let $d_2 = 1$, write f_2 in place of $\mathbf{F}_2$ accordingly. Let the objective function $\tilde{R}(f_2 \circ \mathbf{F}_1)$ be $\hat{R}(f_2 \circ \mathbf{F}_1) + \tau \|\mathbf{w}_{f_2}\|_H$, where $\tau > 0$ is a hyperparameter that can be chosen as desired and

$$\hat{R}(f_2 \circ \mathbf{F}_1) = \frac{1}{N} \sum_{n=1}^{N} \ell(f_2 \circ \mathbf{F}_1,\, (\mathbf{x}_n,\, y_n))$$

with $\ell(f_2 \circ \mathbf{F}_1,\, (\mathbf{x}_n,\, y_n)) = \max(0, 1 - y_n f_2(\mathbf{x}_n))$, the hinge loss. Let $\kappa = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}_{\{y_n = +\}}$. We now re-characterize $\mathbf{F}_1^\star$.

Theorem 4.2. *Assume that* $\tau < \sqrt{2(c-a)} \min(\kappa,\, 1-\kappa)$ *and that there exist* $(\mathbf{x}_+,\, y_+),\, (\mathbf{x}_-,\, y_-) \in S$ *such that* $\ell(f_2^\star \circ \mathbf{F}_1^\star,\, (\mathbf{x}_n,\, y_n)) = 0$, $n = +, -$.

If $\mathbf{F}_1$ *satisfies*

$$\begin{aligned} &k(\mathbf{F}_1(\mathbf{x}_+),\, \mathbf{F}_1(\mathbf{x}_-)) = a \quad \textit{and} \\ &k(\mathbf{F}_1(\mathbf{x}),\, \mathbf{F}_1(\mathbf{x}')) = c \end{aligned} \tag{1}$$

for all pairs of $\mathbf{x}_+,\, \mathbf{x}_- \in S_{\mathbf{X}}$ *and all pairs of* $\mathbf{x},\, \mathbf{x}' \in S_{\mathbf{X}}$ *with* $y = y'$, *then* $\mathbf{F}_1 = \mathbf{F}_1^\star$.

Re-characterize $\mathbf{F}_1^\star$ under regularized supervised representation similarity (SRS) loss as objective

Consider function $h : \mathbb{R}^{d_2} \times \mathbb{R}^{d_2} \to \mathbb{R}$ with the property that $h(\mathbf{x},\, \mathbf{y})$, as a function of $\mathbf{x}$ and $\mathbf{y}$, has the following properties:

- $\inf_{\mathbf{x},\mathbf{y}} h(\mathbf{x},\, \mathbf{y}) = b > -\infty$, $\sup_{\mathbf{x},\mathbf{y}} h(\mathbf{x},\, \mathbf{y}) = d < \infty$, $d > b$;
- h depends only on $\|\mathbf{x} - \mathbf{y}\|_q$ for some $q \geq 1$, i.e., $h(\mathbf{x},\, \mathbf{y}) = h\Big(\|\mathbf{x} - \mathbf{y}\|_q\Big)$;
- h strictly decreases in $\|\mathbf{x} - \mathbf{y}\|_q$ for all $\mathbf{x},\, \mathbf{y} \in \mathbb{R}^{d_2}$ with $h(\mathbf{x},\, \mathbf{y}) > b$.

Define the following SRS loss:

$$\ell(\mathbf{F}_2 \circ \mathbf{F}_1,\, (\mathbf{x},\, y),\, (\mathbf{x}',\, y')) = |g(y,\, y') - h(\mathbf{F}_2(\mathbf{x}),\, \mathbf{F}_2(\mathbf{x}'))|^p,$$

where $p \geq 1$ can be arbitrarily chosen and $g(y,\, y') = b$ if $y \neq y'$ and d if otherwise.

It is easy to see that this loss penalizes the similarity between images of examples under the mapping $\mathbf{F}_2 \circ \mathbf{F}_1$ based on their classes, therefore the name supervised representation similarity.

Let the objective function be

$$\begin{aligned}&\tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1)\\ &\quad = \frac{1}{N^2} \sum_{n,\, m=1}^{N} \ell(\mathbf{F}_2 \circ \mathbf{F}_1,\, (\mathbf{x}_n,\, y_n),\, (\mathbf{x}_m,\, y_m)) + \tau t\Big(\Big\|\mathbf{w}_{f_2^1}\Big\|_H,\, \ldots,\, \Big\|\mathbf{w}_{f_2^{d_2}}\Big\|_H\Big),\end{aligned}$$

where $\tau > 0$ can be freely chosen and t can be any function that strictly decreases in all of its arguments.

Theorem 4.3. *Assume that there exist* $(\mathbf{x}_+,\, y_+),\, (\mathbf{x}_-,\, y_-) \in S$ *such that*

$\ell(\mathbf{F}_2^\star \circ \mathbf{F}_1^\star,\, (\mathbf{x}_+,\, y_+),\, (\mathbf{x}_-,\, y_-)) = 0$. *Also assume that for all* j, $\Big\|\mathbf{w}_{f_2^{j\star}}\Big\|_H > 0$.

If $\mathbf{F}_1$ *satisfies*

$$\begin{aligned}&k(\mathbf{F}_1(\mathbf{x}_+),\, \mathbf{F}_1(\mathbf{x}_-)) = a;\ \textit{and}\\ &k(\mathbf{F}_1(\mathbf{x}),\, \mathbf{F}_1(\mathbf{x}')) = c\end{aligned} \tag{2}$$

for all pairs of $\mathbf{x}_+,\, \mathbf{x}_- \in S_{\mathbf{X}}$ *and all pairs of* $\mathbf{x},\, \mathbf{x}' \in S_{\mathbf{X}}$ *with* $y = y'$, *then* $\mathbf{F}_1 = \mathbf{F}_1^\star$.

On selecting $s,\, u,\, \ell_1$**, and** $\tilde{R}_1$

For both of the two objectives described above, we may choose, for example, s to be the kernel function k, u to be the function defined as $u(y,\, y') = a$ if $y \neq y'$ and c if otherwise, and ℓ_1 to be the SRS loss defined earlier with g set to u and h set to s, i.e.,

$$\ell_1(\mathbf{F}_1,\, (\mathbf{x},\, y),\, (\mathbf{x}',\, y')) = |u(y,\, y') - k(\mathbf{F}_1(\mathbf{x}),\, \mathbf{F}_1(\mathbf{x}'))|^p,$$

where $p \geq 1$ can be freely chosen. This of course requires k to satisfy the aforementioned conditions on h for the resulting loss to be a valid SRS loss.

Under this selection of ℓ_1, it is evident that the minimizers of ℓ_1 (and also R_1) are all equal to $\mathbf{F}_1^\star$ by Theorems 4.2 and 4.3. $\tilde{R}_1$ can be set to the empirical SRS loss plus an arbitrary regularization term on norms of the weights.

Generalizing to l-layer feedforward models with $l \geq 2$

The generalization to a feedforward model with l layers, where $l \geq 2$ can be arbitrary, is trivial. To begin with, treat $\mathbf{F}_l$ and $\mathbf{F}_{l-1} \circ \cdots \circ \mathbf{F}_1$ as the earlier $\mathbf{F}_2$ and $\mathbf{F}_1$, respectively. Then work as in the two-layer case to find an objective $\tilde{R}_{l-1}$ for $\mathbf{F}_{l-1} \circ \cdots \circ \mathbf{F}_1$. This reduces the l-layer problem to an $l-1$-layer problem. Repeat this procedure on the rest of the layers until we return to the original two-layer case.

4.3 A Layer-Wise Training Algorithm for an l-Layer ($l \geq 2$) Feedforward Network for Classification

We can build upon the above realizations an certified (in the sense that the optimality is guaranteed) layer-wise algorithm for training an l-layer ($l \geq 2$) feedforward network for classification tasks. In this section, we describe this algorithm and show that it enjoys a geometric interpretation that makes the learning dynamics transparent. Moreover, we show that there is a simple acceleration method for the kernelized non-input layers, making the architecture more practical.

We present this algorithm for binary classification. Nevertheless, as multi-class problems can be reduced to a set of binary classification problems by using either the

one-vs-all or the one-vs-one strategy (Schölkopf & Smola, 2001), an extension of this algorithm to multi-class problems is trivial.

The architecture considered is as follows:

$$\Bigg\{ f_l \circ \cdots \circ \mathbf{F}_1 : \mathbb{R}^{d_0} \to \mathbb{R} \mid$$

$$\mathbf{F}_i(\mathbf{x}) = (f_i^1(\mathbf{x}),\, \ldots,\, f_i^{d_i}(\mathbf{x})), f_i^j(\mathbf{x}) = \left\langle \mathbf{w}_{f_i^j},\, \phi_i(\mathbf{x}) \right\rangle_{H_i} + b_{f_i^j},\, \mathbf{w}_{f_i^j} \in H_i,\, b_{f_i^j} \in \mathbb{R},$$

$$\forall l > i > 1,\, \forall j,\, f_l(\mathbf{x}) = \langle \mathbf{w}_{f_l},\, \phi_l(\mathbf{x}) \rangle_{H_l} + b_{f_l},\, \mathbf{w}_{f_l} \in H_l,\, b_{f_l} \in \mathbb{R} \Bigg\}.$$

Note that we have made no assumption on the input layer.

For $i < l$, define

$$\tilde{R}_i(\mathbf{F}_i) = \frac{1}{N^2} \sum_{n,\, m}^{N} \ell_i(\mathbf{F}_i,\, (\mathbf{x}_n,\, y_n),\, (\mathbf{x}_m,\, y_m)) + \tau_i t\left(\left\| \mathbf{w}_{f_i^1} \right\|_{H_i},\, \ldots,\, \left\| \mathbf{w}_{f_i^{d_i}} \right\|_{H_i} \right),$$

where

$$\ell_i(\mathbf{F}_i,\, (\mathbf{x}_n,\, y_n),\, (\mathbf{x}_m,\, y_m)) = |u(y_n,\, y_m) - k_{i+1}(\mathbf{F}_i(\mathbf{x}_n),\, \mathbf{F}_i(\mathbf{x}_m))|^p,$$

$p \geq 1$ can be chosen freely, $\tau > 0$, $u(y,\, y') = a$ if $y = y'$ and c otherwise, and t can be any function that strictly decreases in all of its arguments.

For $i = l$, define :

$$\tilde{R}_l(f_l) = \frac{1}{N} \sum_{n=1}^{N} \ell_l(f_l,\, (\mathbf{x}_n,\, y_n)) + \tau_l \|\mathbf{w}_{f_l}\|_{H_l},$$

where

$$\ell_l(f_l,\, (\mathbf{x}_n,\, y_n)) = \max(0, 1 - y_n f_l(\mathbf{x}_n))$$

Then the training algorithm is given in Algorithm 1.

The optimality of this training algorithm is justified by Theorems 4.2 and 4.3 when the τ_i and k_i satisfy the corresponding conditions for all $i = 1,\, \ldots,\, l$.

Algorithm 1 A certified layer-wise training algorithm for classification.

input: training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$

initialize: initialize and freeze all layers

for $i = 1, 2, \ldots, l$ **do**

unfreeze layer i

train layer i to minimize $\tilde{R}_i$

freeze layer i

end for

We emphasize that this particular training algorithm gives great freedom to the choice of $\mathbf{F}_1$: it can be any arbitrary architecture. In particular, it can be a stack of multiple layers in practice. This stack can be trained with an end-to-end method such as BP.

4.3.1 Geometric Interpretation of Learning Dynamics

The sufficient conditions described by Eq. 1 and Eq. 2 can be interpreted geometrically: under an $\mathbf{F}_1$ satisfying these conditions, images of examples from distinct classes are as distant as possible in the RKHS induced by k whereas images of examples from the same class are as concentrated as possible (see proof of Theorem 4.2 in Appendix B). Intuitively, such a representation is the "easiest" for the classification task. And our earlier theorems essentially justified this intuition in a rigorous fashion.

Therefore, the learning dynamics of this training algorithm can be given a straightforward geometric interpretation: it trains each layer to push apart examples from different classes while squeeze together those within the same class. In other words, each layer learns a better representation of the data. Eventually, the output layer works as a classifier

on the final hidden representation.

4.3.2 Accelerating the Kernelized Layers

There is a natural method to accelerate the kernelized non-input layers: the hidden targets are sparse in the sense that for $1 \leq i < l$ and any $\mathbf{F}_i$ satisfying Eq. 1 or Eq. 2, we have $\phi_{i+1}(\mathbf{F}_i(\mathbf{x}_m)) = \phi_{i+1}(\mathbf{F}_i(\mathbf{x}_n))$ if $y_m = y_n$ and $\phi_{i+1}(\mathbf{F}_i(\mathbf{x}_m)) \neq \phi_{i+1}(\mathbf{F}_i(\mathbf{x}_n))$ if $y_m \neq y_n$ (see proof of Theorem 4.2 in Appendix B). Since we usually approximate $\mathbf{w}_{i+1}^j$ using $\sum_{n=1}^N \alpha_{i+1,\,n}^j \phi_{i+1}(\mathbf{F}_i(\mathbf{x}_n))$, retaining only one example from each class would result in exactly the same hypothesis class $\mathbb{F}_{i+1}$ because $\left\{\sum_{n=1}^N \alpha_{i+1,\,n}^j \phi_{i+1}(\mathbf{F}_i(\mathbf{x}_n)) \mid \alpha_{i+1,\,n}^j \in \mathbb{R}\right\} = \left\{\sum_{n=+,\,-} \alpha_{i+1,\,n}^j \phi_{i+1}(\mathbf{F}_i(\mathbf{x}_n)) \mid \alpha_{i+1,\,n}^j \in \mathbb{R}\right\}$ for arbitrary $\mathbf{x}_+$, $\mathbf{x}_-$ in $S_{\mathbf{X}}$.

Thus, after training a given layer, depending on how well its objective function has been minimized, one may discard some of the centers for kernel machines of the next layer to speed up the training of that layer without sacrificing performance. This trick also has a regularization effect on the kernel machines since the number of trainable parameters of a kernel machine grows linearly in the number of its centers.

4.4 How is our layer-wise framework different from the existing layer-wise pre-training schemes?

Existing layer-wise pre-training methods such as those proposed in (Hinton et al., 2006) and (Bengio et al., 2007) require backpropagation (BP) fine-tuning. This is because, to the best of our knowledge, no optimality guarantee comparable to that provided by BP can be made for these pre-training algorithms. In other words, the layer-wise pre-training commonly used in the deep learning community does not necessarily learn

the hypothesis that minimizes the objective function for the network and thus can only be used as an add-on to BP that helps BP converge faster.

In contrast, our work proves such optimality for our layer-wise training scheme in certain specific learning settings and therefore completely removes the need for BP in these settings. To put this in another way, even if one applies BP after performing our layer-wise training, one will not (in theory) end up with a hypothesis that is strictly better than the one learned by the layer-wise learning process in terms of minimizing the objective function of the network.

Coming up with a purely layer-wise substitute for BP is relevant because, as we have mentioned, BP can be computationally expensive and its end-to-end nature makes it practically impossible to precisely trace the source of unsatisfying performance and find out which layer or layers is to be blamed. This can make the architecture search process lengthy and sometimes painful. Furthermore, training all layers simultaneously complicates the parameter space and may introduce more local minima to the learning process, which can be another unwanted factor for gradient descent-based learning. In contrast, a fully layer-wise training process allows one to divide and conquer the learning problem and reveals more useful information about training, mitigating the aforementioned issues to some extent.

5 Related Works

The link between NNs and the kernel method has been long known. In (Vapnik, 2000), the hyperbolic tangent kernel was defined and used in SVM, leading to an architecture

equivalent to a shallow MLP. Suykens & Vandewalle (1999) viewed MLP as SVM by treating the hidden layer as the feature map and proposed accordingly a modified support vector method to train the former. More recently, Cho & Saul (2009) defined an "arc cosine" kernel to imitate the computations performed by a one-layer MLP. Zhuang et al. (2011) extended the idea to arbitrary kernels with a focus on MKL, using an architecture similar to a two-layer kMLP. As a further generalization, Zhang et al. (2017) proposed kMLP and fully-kernelized CNN. However, they did not extend the idea to more network architectures. These works essentially combine kernel method with deep learning by substituting neurons in NNs with kernel machines, which is similar to what we are pursuing in this work. However, to the best of our knowledge, our work enjoys perhaps the greatest generality among works that follow this line of research.

There are also works that attempt to integrate kernel method with deep learning using other methods. Suykens (2017) drew connections between restricted Boltzmann machines (RBM) and kernel machines by creating RBM-like representations for the latter. The resulting restricted kernel machines (RKMs) are then combined to form deep RKMs. Mairal et al. (2014) proposed to learn hierarchical representations by learning mappings of kernels that are invariant to irrelevant variations in images. Hermans & Schrauwen (2012) used the kernel method to expand the echo state networks to essentially infinite-sized recurrent neural networks. The resulting network can then be viewed as a recursive kernel that can be used in SVMs. Wilson et al. (2016) proposed to learn the covariance matrix of a Gaussian process using an NN in order to make the kernel "adaptive". Such an interpretation of "adaptive" kernels can be given to KNs as well. This idea also underlies the now standard approach of combining a deep NN with

SVM for classification, which was first explored by Huang & LeCun (2006) and Tang (2013) and can be viewed as a special case of the proposed kernelization framework. In terms of the training of such hybrid systems, there are mainly two methods. The first is to apply BP to the entire model (Tang, 2013), which enjoys an optimality guarantee from BP but forces the SVM to be trained with gradient descent instead of the more efficient optimization algorithms that are usually used for SVMs. The alternative is to feed the hidden representations from a trained NN to the SVM and train the latter in the usual way (Huang & LeCun, 2006), but this practice is not theoretically solid. The proposed layer-wise learning framework serves as another alternative that combines the best of both worlds: one can train the NN and SVM separately with an optimality guarantee as that given by BP.

Much works have been done to improve or substitute BP in learning a deep architecture. Most aim at improving the classical method, working as add-ons for BP. The most notable ones are perhaps the unsupervised greedy pre-training techniques proposed by Hinton et al. (2006) and Bengio et al. (2007). Among works that try to completely substitute BP, none provided a comparable optimality guarantee in theory as that given by BP. Fahlman & Lebiere (1990) pioneered the idea of greedily learn the architecture of an NN. In their work, each new node is added to maximize the correlation between its output and the residual error signal. Several authors explored the idea of approximating error signals propagated by BP locally at each layer or each node (Bengio, 2014; Carreira-Perpinan & Wang, 2014; Lee et al., 2015; Balduzzi et al., 2015; Jaderberg et al., 2016). Zhou & Feng (2017) proposed a BP-free deep architecture based on decision trees. Raghu et al. (2017) attempted to quantify the quality of hidden representations

toward learning more interpretable deep architectures, sharing a motivation similar to ours.

6 Experiments

We now demonstrate the competence of the kernelized models and the effectiveness of the proposed layer-wise framework via experiments. We will be implementing the sample training algorithm described in Section 4.3 throughout. This section will be divided into two parts. The first one will be dedicated to comparing KNs with traditional kernel machines. In the second part, we compare KNs with other popular connectionist models. These empirical results serve as proofs of concept for the proposed architectures as well as the greedy training framework.

6.1 Comparing KNs with Classical Kernel Machines

We now compare a single-hidden-layer kMLP using simple, generic kernels with the classical SVM and SVMs enhanced by MKL algorithms that used significantly more kernels to demonstrate the competence of kMLP and in particular, its ability to perform well without excessive kernel parameterization. The standard SVM and seven other SVMs enhanced by popular MKL methods were compared (Zhuang et al., 2011), including the classical convex MKL (Lanckriet et al., 2004) with kernels learned using the extended level method proposed in (Xu et al., 2009) ($\text{MKL}^{\text{LEVEL}}$); MKL with L^p norm regularization over kernel weights (Kloft et al., 2011) (L^pMKL), for which the cutting plane algorithm with second order Taylor approximation of L^p was adopted;

Generalized MKL in (Varma & Babu, 2009) (GMKL), for which the target kernel class was the Hadamard product of single Gaussian kernel defined on each dimension; Infinite Kernel Learning in (Gehler & Nowozin, 2008) (IKL) with $\text{MKL}^{\text{LEVEL}}$ as the embedded optimizer for kernel weights; 2-layer Multilayer Kernel Machine in (Cho & Saul, 2009) (MKM); 2-Layer MKL (2LMKL) and Infinite 2-Layer MKL in (Zhuang et al., 2011) ($\text{2LMKL}^{\text{INF}}$).

Eleven binary classification data sets that have been widely used in MKL literature were split evenly for training and test and were all normalized to zero mean and unit variance prior to training. Twenty runs with identical settings but random weight initializations were repeated for each model. For each repetition, a new training-test split was selected randomly.

For kMLP, all results were achieved using a greedily-trained, one-hidden-layer model with the number of kernel machines ranging from 3 to 10 on the first layer for different data sets. The second layer was a single kernel machine. All kernel machines within one layer used the same Gaussian kernel ($k(\mathbf{x},\, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_2/\sigma^2}$), and the two kernels on the two layers differed only in kernel width σ. All hyperparameters were chosen via 5-fold cross-validation. As for the other models compared, for each data set, SVM used a Gaussian kernel. For the MKL algorithms, the base kernels contained Gaussian kernels with 10 different widths on all features and on each single feature and polynomial kernels of degree 1 to 3 on all features and on each single feature. For $\text{2LMKL}^{\text{INF}}$, one Gaussian kernel was added to the base kernels at each iteration. Each base kernel matrix was normalized to unit trace. For L^pMKL, p was selected from $\{2, 3, 4\}$. For MKM, the degree parameter was chosen from $\{0, 1, 2\}$. All hyperparameters were selected via

Table 1: Average test error (%) and standard deviation (%) from 20 runs. Results with overlapping 95% confidence intervals (not shown) are considered equally good. Best results are marked in bold. The average ranks (calculated using average test error) are provided in the bottom row. When computing confidence intervals, due to the limited sizes of the data sets, we pooled the twenty random samples.

	Size/Dimension	SVM	MKL$^{\text{LEVEL}}$	L^pMKL	GMKL	IKL	MKM	2LMKL	2LMKL$^{\text{INF}}$	kMLP-1
Breast	683/10	3.2± 1.0	3.5± 0.8	3.8± 0.7	3.0± 1.0	3.5± 0.7	2.9± 1.0	3.0± 1.0	3.1± 0.7	**2.4± 0.7**
Diabetes	768/8	**23.3± 1.8**	24.2± 2.5	27.4± 2.5	33.6± 2.5	**24.0± 3.0**	24.2± 2.5	**23.4± 1.6**	**23.4± 1.9**	**23.2± 1.9**
Australian	690/14	15.4± 1.4	15.0± 1.5	15.5± 1.6	20.0± 2.3	**14.6± 1.2**	14.7± 0.9	**14.5± 1.6**	**14.3± 1.6**	**13.8± 1.7**
Iono	351/33	7.2± 2.0	8.3± 1.9	7.4± 1.4	7.3± 1.8	6.3± 1.0	8.3± 2.7	7.7± 1.5	**5.6± 0.9**	**5.0± 1.4**
Ringnorm	400/20	**1.5± 0.7**	**1.9± 0.8**	3.3± 1.0	2.5± 1.0	**1.5± 0.7**	2.3± 1.0	2.1± 0.8	**1.5± 0.8**	**1.5± 0.6**
Heart	270/13	17.9± 3.0	17.0± 2.9	23.3± 3.8	23.0± 3.6	**16.7± 2.1**	17.6± 2.5	**16.9± 2.5**	**16.4± 2.1**	**15.5± 2.7**
Thyroid	140/5	6.1± 2.9	7.1± 2.9	6.9± 2.2	5.4± 2.1	5.2± 2.0	7.4± 3.0	6.6± 3.1	5.2± 2.2	**3.8± 2.1**
Liver	345/6	**29.5± 4.1**	37.7± 4.5	30.6± 2.9	36.4± 2.6	40.0± 2.9	**29.9± 3.6**	34.0± 3.4	37.3± 3.1	**28.9± 2.9**
German	1000/24	**24.8± 1.9**	28.6± 2.8	25.7± 1.4	29.6± 1.6	30.0± 1.5	**24.3± 2.3**	25.2± 1.8	25.8± 2.0	**24.0± 1.8**
Waveform	400/21	11.0± 1.8	11.8± 1.6	11.1± 2.0	11.8± 1.8	**10.3± 2.3**	**10.0± 1.6**	11.3± 1.9	**9.6± 1.6**	**10.3± 1.9**
Banana	400/2	**10.3± 1.5**	**9.8± 2.0**	12.5± 2.6	16.6± 2.7	**9.8± 1.8**	19.5± 5.3	13.2± 2.1	**9.8± 1.6**	11.5± 1.9
Rank	-	4.2	6.3	7.0	6.9	4.3	5.4	5.0	2.8	1.6

5-fold cross-validation.

From Table 1, kMLP compares favorably with other models, which validates our claim that kMLP learns its own kernels nonparametrically hence can work well even without excessive kernel parameterization. Performance difference among models can be small for some data sets, which is expected since these datasets are all rather small in size and not too challenging. Nevertheless, it is worth noting that only two Gaussian kernels were used for kMLP, whereas all other models except for SVM used significantly more kernels.

6.2 Comparing KNs with NNs

In this section, we provide empirical results on comparing KN with NN. In the first part, we demonstrate the competence of kernelized NNs and the effectiveness of the layer-wise learning method using kMLPs. We use the proposed layer-wise algorithm derived from our greedy learning framework and Adam (Kingma & Ba, 2014) as the underlying optimization algorithm. First, we show that this algorithm, albeit only having been certified under certain families of objectives, works well with most popular objective functions in practice. We then compare kMLPs trained with BP and the layer-wise algorithm to show the effectiveness of the latter. Finally, to further showcase the competence of the greedily-trained kernelized models, we compare kMLPs learned layer-wise with other popular deep architectures including MLPs, Deep Belief Networks (DBNs) (Hinton & Salakhutdinov, 2006) and Stacked Autoencoders (SAEs) (Vincent et al., 2010), with the last two trained using a combination of unsupervised greedy pre-training and standard BP (Hinton et al., 2006; Bengio et al., 2007). We also visualize the learning dynamics of greedy kMLPs and show that it is intuitive and simple to interpret. In the second part of the experiments, we partially kernelize the classic LeNet-5 (LeCun et al., 1998) and compare it with the original to validate our claim that the proposed kernelization and training algorithm is flexible in the sense that it works well with any given feedforward NN architecture and one can freely decide the degree of kernelization. The hidden representations learned from the two models are visualized. We show that the hidden representations learned by the kernelized model are much more discriminative than that from the original.

Figure 2: From left to right: example from *rectangles*, *rectangles-image*, *convex*, *mnist (50k test)* and *mnist (50k test) rotated*.

6.2.1 Part 1: Kernelizing MLPs

In terms of the datasets used. *rectangles, rectangles-image* and *convex* are binary classification datasets, *mnist (50k test)* and *mnist (50k test) rotated* are variants of MNIST. *fashion-mnist* is the Fashion-MNIST dataset (Xiao et al., 2017). These datasets all contain 28×28 grayscale images. In *rectangles, rectangles-image*, the model needs to learn if the height of the rectangle is longer than the width, and in *convex*, if the white region is convex. Examples from these datasets are shown in Fig 2. In actual training, no preprocessing method was used. As for the specific kernels used, we used Gaussian kernels ($k(\mathbf{x},\, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_2/\sigma^2}$) for the kernelized models for all our experiments. To ensure that the comparisons with other models are fair, we used the regularized (two-norm regularization on weights) cross-entropy loss as the objective function for the output layer of all models. More details can be found in Appendix A.

We first test the effect of using different hidden loss functions using a two-hidden-layer kMLP. The three hidden layer loss functions tested include the proposed SRS-1 loss, i.e., the SRS loss with $p = 1$, the SRS-2 loss and the empirical alignment (Cristianini et al., 2002) between $\mathbf{G}_i$ and $\mathbf{G}^\star$, where i is the hidden layer being optimized. $\mathbf{G}_i$ is the

kernel matrix of k_{i+1} computed on $\mathbf{F}_i(S_\mathbf{X})$ and $\mathbf{G}^\star$ is the kernel matrix induced by g on $S_\mathbf{X}$. The regularization term was always chosen to be the sum of the L^2 norms of the weights. On *convex*, this kMLP achieved a test error rate of 19.36%, 18.53% and 21.70% using alignment, SRS-2 and SRS-1 as the hidden losses, respectively. As a baseline, our best two-hidden-layer MLP achieved an error rate of 23.28% on this dataset. For the rest of our experiments, we use the best result from using these three hidden losses for our greedily-trained models.

We now test the layer-wise learning algorithm against BP using the standard MNIST dataset (LeCun et al., 2010). Results from several MLPs were added as benchmarks. These models were trained with Adam or RMSProp (Tieleman & Hinton, 2012) and extra training techniques such as dropout (Srivastava et al., 2014) and batch normalization (BN) (Ioffe & Szegedy, 2015) were applied to boost performance. kMLPs accelerated using the proposed method (kMLP$^{\text{FAST}}$) were also tested, for which we randomly discarded some centers of each non-input layer before its training. Two popular acceleration methods for kernel machines were compared, including using a parametric representation (kMLP$^{\text{PARAM}}$), i.e., for each node in a kMLP, $f(\mathbf{x}) = \sum_{n=1}^{m} \alpha_n k(\mathbf{w}_n, \mathbf{x})$, α_n, $\mathbf{w}_n$ learnable and m a hyperparameter, and using random Fourier features (kMLP$^{\text{RFF}}$) (Rahimi & Recht, 2008).

Results in Table 2 validate the effectiveness of our layer-wise algorithm. For both the single-hidden-layer and the two-hidden-layer kMLPs, the layer-wise algorithm consistently outperformed BP. The layer-wise method is also much faster than BP. In fact, it is practically impossible to use BP to train kMLP with more than two hidden layers without any acceleration method due to the computational complexity involved.

Table 2: Testing the proposed layer-wise algorithm and acceleration method on MNIST. The numbers following the model names indicate the number of hidden layers used. For kMLP$^{\text{FAST}}$, we also include in parentheses the ratio between the number of training examples randomly chosen as centers for the kernel machines on the layer and the size of the training set. Apart from kMLP-2 (BP), the BP kMLP results are from (Zhang et al., 2017). For this and all following tables in this paper, the entries correspond to test errors ($\%$) and 95% confidence intervals ($\%$). Results with overlapping confidence intervals are considered equally good. Best results are marked in bold.

MLP-1 (RMSPROP+BN)	MLP-1 (RMSPROP+DROPOUT)	MLP-2 (RMSPROP+BN)	MLP-2 (RMSPROP+DROPOUT)	KMLP-1 (BP)	KMLP-1 (GREEDY)	KMLP-1$^{\text{RFF}}$ (BP)
2.05 ± 0.28	1.77 ± 0.26	$\mathbf{1.58 \pm 0.24}$	1.67 ± 0.25	3.44 ± 0.36	1.77 ± 0.26	2.01 ± 0.28

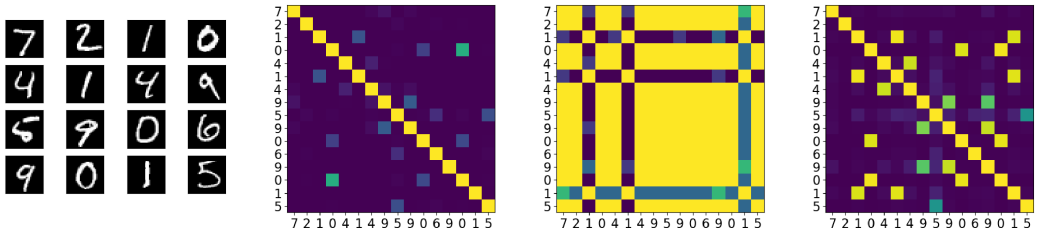
KMLP-1$^{\text{PARAM}}$ (BP)	KMLP-1$^{\text{FAST}}$ (GREEDY)	KMLP-2 (BP)	KMLP-2 (GREEDY)	KMLP-2$^{\text{RFF}}$ (BP)	KMLP-2$^{\text{PARAM}}$ (BP)	KMLP-2$^{\text{FAST}}$ (GREEDY)
1.88 ± 0.27	1.75 ± 0.26 (0.54)	3.66 ± 0.37	$\mathbf{1.56 \pm 0.24}$	1.92 ± 0.27	2.45 ± 0.30	$\mathbf{1.47 \pm 0.24}$ **(1/0.19)**

Table 3: Comparing kMLPs (trained fully layer-wise) with MLPs and other popular deep architectures trained with BP and BP enhanced by unsupervised greedy pre-training. The MLP-1 (SGD), DBN and SAE results are from (Larochelle et al., 2007). Note that in order to be consistent with (Larochelle et al., 2007), the MNIST results below were obtained using a train/test split (10k/50k) more challenging than what is commonly used in the literature.

	RECTANGLES	RECTANGLES-IMAGE	CONVEX	MNIST (50K TEST)	MNIST (50K TEST) ROTATED	FASHION-MNIST
MLP-1 (SGD)	7.16 ± 0.23	33.20 ± 0.41	32.25 ± 0.41	4.69 ± 0.19	18.11 ± 0.34	15.47 ± 0.71
MLP-1 (ADAM)	5.37 ± 0.20	28.82 ± 0.40	30.07 ± 0.40	4.71 ± 0.19	18.64 ± 0.34	12.98 ± 0.66
MLP-1 (RMSPROP+BN)	5.37 ± 0.20	23.81 ± 0.37	28.60 ± 0.40	4.57 ± 0.18	18.75 ± 0.34	14.55 ± 0.69
MLP-1 (RMSPROP+DROPOUT)	5.50 ± 0.20	23.67 ± 0.37	36.28 ± 0.42	4.31 ± 0.18	14.96 ± 0.31	12.86 ± 0.66
MLP-2 (SGD)	5.05 ± 0.19	$\mathbf{22.77 \pm 0.37}$	25.93 ± 0.38	5.17 ± 0.19	18.08 ± 0.34	12.94 ± 0.66
MLP-2 (ADAM)	4.36 ± 0.18	25.69 ± 0.38	25.68 ± 0.38	4.42 ± 0.18	17.22 ± 0.33	$\mathbf{11.48 \pm 0.62}$
MLP-2 (RMSPROP+BN)	4.22 ± 0.18	23.12 ± 0.37	23.28 ± 0.37	3.57 ± 0.16	13.73 ± 0.30	11.51 ± 0.63
MLP-2 (RMSPROP+DROPOUT)	4.75 ± 0.19	23.24 ± 0.37	34.73 ± 0.42	3.95 ± 0.17	13.57 ± 0.30	$\mathbf{11.05 \pm 0.61}$
DBN-1	4.71 ± 0.19	23.69 ± 0.37	19.92 ± 0.35	3.94 ± 0.17	14.69 ± 0.31	N/A
DBN-3	2.60 ± 0.14	$\mathbf{22.50 \pm 0.37}$	$\mathbf{18.63 \pm 0.34}$	3.11 ± 0.15	$\mathbf{10.30 \pm 0.27}$	N/A
SAE-3	2.41 ± 0.13	24.05 ± 0.37	$\mathbf{18.41 \pm 0.34}$	3.46 ± 0.16	$\mathbf{10.30 \pm 0.27}$	N/A
KMLP-1	$\mathbf{2.24 \pm 0.13}$	23.29 ± 0.37	19.15 ± 0.34	$\mathbf{3.10 \pm 0.15}$	11.09 ± 0.28	11.72 ± 0.63
KMLP-1$^{\text{FAST}}$	$\mathbf{2.36 \pm 0.13\ (0.05)}$	$23.86 \pm 0.37\ (0.01)$	$20.34 \pm 0.35\ (0.17)$	$\mathbf{2.95 \pm 0.15\ (0.1)}$	$12.61 \pm 0.29\ (0.1)$	$\mathbf{11.45 \pm 0.62\ (0.28)}$
KMLP-2	$\mathbf{2.24 \pm 0.13}$	23.30 ± 0.37	$\mathbf{18.53 \pm 0.34}$	3.16 ± 0.15	$\mathbf{10.53 \pm 0.27}$	$\mathbf{11.23 \pm 0.62}$
KMLP-2$^{\text{FAST}}$	$\mathbf{2.21 \pm 0.13\ (0.3/0.3)}$	$23.24 \pm 0.37\ (0.01/0.3)$	$19.32 \pm 0.35\ (0.005/0.03)$	$3.18 \pm 0.15\ (0.3/0.3)$	$10.94 \pm 0.27\ (0.1/0.7)$	$\mathbf{10.85 \pm 0.61\ (1/0.28)}$

Moreover, it is worth noting that the proposed acceleration trick is clearly very effective despite its simplicity and even produced models outperforming the original ones, which may be due to its regularization effect. This shows that kMLP together with the greedy learning scheme can be of practical interest even when dealing with the massive data sets in today's machine learning.

From Table 3, we see that the performance of kMLP is on par with some of the most popular and most mature deep architectures. In particular, the greedily-trained kMLPs compared favorably with their direct NN equivalents, i.e., the MLPs, even though neither batch normalization nor dropout was used for the former.



(a) Examples from test set.

(b) Kernel matrix of the first hidden layer (epoch 25).

(c) Kernel matrix of the second hidden layer (epoch 0).

(d) Kernel matrix of the second hidden layer (epoch 15).

Figure 3: Visualizing the learning dynamics in a two-hidden-layer kMLP. Each entry in the kernel matrices corresponds to the inner product between the learned representations of two examples in the RKHS. The labels are given on the two axes. The examples used to produce this figure are provided in Fig. 3a in the order of the labels plotted. The darker the entry, the more distant the learned representations are in the RKHS.

In Fig. 3, we visualize the learning dynamics within a two-hidden-layer kMLP learned layer-wise. Since by construction of the Gaussian kernel, the image vectors are all of unit norm in the RKHS, we can visualize the distance between two vectors by visualizing the value of their inner product. In Fig. 3d, we can see that while the image vectors are distributed randomly prior to training (see Fig. 3c), there is a clear pattern in their distribution after training that reflects the dynamics of training: the layer-wise algorithm squeezes examples from the same class closer together while pushes examples from different class farther apart. And it is easy to see that such a representation would be simple to classify. Fig. 3b and 3d suggest that this greedy, layer-wise algorithm still learns "deep" representations: the higher-level representations are more distinctive for different digits than the lower-level ones. Moreover, since learning becomes increasingly simple for the upper layers as the representations become more and more well-behaved, these layers are usually easy to set up and converge very fast during training.

6.2.2 Part 2: Kernelizing the Classic LeNet-5

We kernelize the output layer of the classic LeNet-5 (LeCun et al., 1998) architecture and train it layer-wise with all the layers but the output layer as one layer and the output layer as a second layer. The non-output layers are trained with BP. This is to demonstrate that our kernelization method and the layer-wise algorithm are flexible in the sense that the former can be applied to only a part of the network and that the latter works well with partly-kernelized models. Since we are interested in evaluating the layer-wise algorithm on partly-kernelized NNs instead of pursuing state-of-the-art performance, we use the original LeNet-5 without increasing the size of any layer or the number of layers.

Table 4: Kernelizing the output layer of the classic LeNet-5. The kernelized model (kLeNet-5) was trained layer-wise.

	MNIST	FASHION-MNIST	CIFAR-10
LENET-5	**0.76 ± 0.17**	9.34 ± 0.57	**36.42 ± 0.94**
KLENET-5	**0.75 ± 0.17**	**8.67 ± 0.55**	**35.87 ± 0.94**

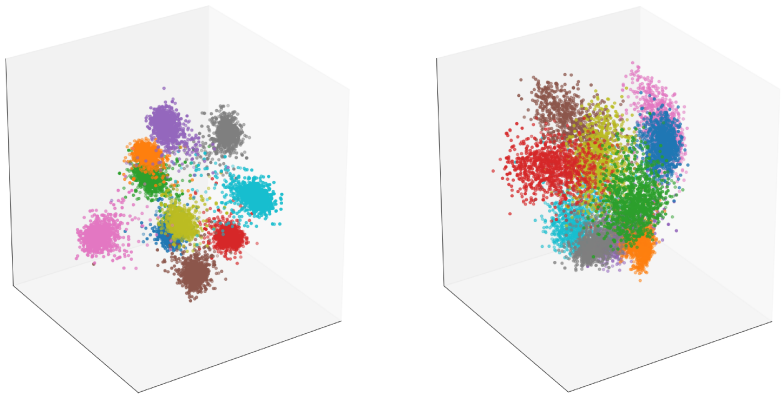


Figure 4: Visualizing the data representation of the MNIST test set in the last hidden layer of kLeNet-5 (left) and LeNet-5 (right). Each color corresponds to a digit. Representations learned by kLeNet-5 are more discriminative for different digits.

ReLU (Glorot et al., 2011) and max pooling were used as activations and pooling layers, respectively. Both models were optimized using Adam. The two networks were trained and tested on the unpreprocessed MNIST, Fashion-MNIST and CIFAR-10 (Krizhevsky & Hinton, 2009) datasets.

In Table 4, the results suggest that kernelization and the layer-wise algorithm resulted in marginal accuracy increase in all datasets. We emphasize that the layer-wise framework does not help the network learn intrinsically superior hypotheses compared to the

traditional end-to-end methods. In that regard, it offers the same optimality guarantee as that provided by an end-to-end method such as BP. We argue that the layer-wise framework is promising because it is more light-weight and returns more information on the training of the individual layers to the user, making possible new and more flexible model selection and hyperparameter-tuning paradigms. This could serve as a tentative step toward increasing the interpretability of deep architectures.

Fig. 4 provides more insights into the difference of kLeNet-5 and LeNet-5, in which we plotted the activations of the last hidden layer of the two models after PCA dimension reduction using the MNIST test set. In particular, we see that the representations in the last hidden layer of kLetNet-5 are much more discriminative for different digits than those in the corresponding layer of LeNet-5. Note that since the two models differed only in their output layers, this observation suggests that the layer-wise training algorithm turns deep architectures into more efficient representation learners, which may prove useful for computer vision tasks that build on convolutional features (Gatys et al., 2015; Gardner et al., 2015).

7 Conclusion

In this paper, we first proposed a family of connectionist models based on the kernel method and then presented a framework to train multilayer feedforward networks in a greedy, layer-by-layer fashion. Several realizations of the framework was provided and their optimality proven. Finally, we described a certified layer-wise training algorithm for deep feedforward architectures for classification based on the earlier realizations.

Empirical results were provided to supplement out theory, in which our proposed models and the layer-wise training algorithm compared favorably with classical kernel machines as well as other popular connectionist models.

References

Bach, F. R., Lanckriet, G. R., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on machine learning* (p. 6).

Balduzzi, D., Vanchinathan, H., & Buhmann, J. M. (2015). Kickback cuts backprop's red-tape: Biologically plausible credit assignment in neural networks. In *Aaai* (pp. 485–491).

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, *3*(Nov), 463–482.

Bengio, Y. (2014). How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153–160).

Carreira-Perpinan, M., & Wang, W. (2014). Distributed optimization of deeply nested systems. In *Artificial intelligence and statistics* (pp. 10–19).

Cho, Y., & Saul, L. K. (2009). Kernel methods for deep learning. In *Advances in neural information processing systems* (pp. 342–350).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. S. (2002). On kernel-target alignment. In *Advances in neural information processing systems* (pp. 367–373).

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in neural information processing systems* (pp. 524–532).

Gardner, J. R., Upchurch, P., Kusner, M. J., Li, Y., Weinberger, K. Q., Bala, K., & Hopcroft, J. E. (2015). Deep manifold traversal: Changing labels with convolutional features. *arXiv preprint arXiv:1511.06421*.

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

Gehler, P., & Nowozin, S. (2008). Infinite kernel learning.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323).

Gönen, M., & Alpaydın, E. (2011). Multiple kernel learning algorithms. *Journal of machine learning research*, *12*(Jul), 2211–2268.

Hermans, M., & Schrauwen, B. (2012). Recurrent kernel machines: Computing with infinite echo state networks. *Neural Computation*, *24*(1), 104–133.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Huang, F. J., & LeCun, Y. (2006). Large-scale learning with svm and convolutional for generic object categorization. In *Computer vision and pattern recognition, 2006 ieee computer society conference on* (Vol. 1, pp. 284–291).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., & Kavukcuoglu, K. (2016). Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kloft, M., Brefeld, U., Sonnenburg, S., & Zien, A. (2011). Lp-norm multiple kernel learning. *Journal of Machine Learning Research*, *12*(Mar), 953–997.

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images* (Tech. Rep.). Citeseer.

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

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on machine learning* (pp. 473–480).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., Cortes, C., & Burges, C. (2010). Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, *2*.

Lee, D.-H., Zhang, S., Fischer, A., & Bengio, Y. (2015). Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 498–515).

Mairal, J., Koniusz, P., Harchaoui, Z., & Schmid, C. (2014). Convolutional kernel networks. In *Advances in neural information processing systems* (pp. 2627–2635).

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Micchelli, C. A., Xu, Y., & Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, *7*(Dec), 2651–2667.

Park, J., & Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural computation*, *3*(2), 246–257.

Paszke, A., Gross, S., Chintala, S., & Chanan, G. (2017). *Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration.*

Pisier, G. (1999). *The volume of convex bodies and banach space geometry* (Vol. 94). Cambridge University Press.

Raghu, M., Gilmer, J., Yosinski, J., & Sohl-Dickstein, J. (2017). Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in neural information processing systems* (pp. 6076–6085).

Rahimi, A., & Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems* (pp. 1177–1184).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–538.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. In *Computational learning theory* (pp. 416–426).

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Sun, S., Chen, W., Wang, L., Liu, X., & Liu, T.-Y. (2016). On the depth of deep neural networks: A theoretical view. In *Aaai* (pp. 2066–2072).

Suykens, J. A. (2017). Deep restricted kernel machines using conjugate feature duality. *Neural computation*, *29*(8), 2123–2163.

Suykens, J. A., & Vandewalle, J. (1999). Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks*, *10*(4), 907–911.

Tang, Y. (2013). Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*.

Vapnik, V. (2000). The nature of statistical learning theory.

Varma, M., & Babu, B. R. (2009). More generality in efficient multiple kernel learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 1065–1072).

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*(Dec), 3371–3408.

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016). Deep kernel learning. In *Artificial intelligence and statistics* (pp. 370–378).

Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.*

Xu, Z., Jin, R., King, I., & Lyu, M. (2009). An extended level method for efficient multiple kernel learning. In *Advances in neural information processing systems* (pp. 1825–1832).

Zhang, S., Li, J., Xie, P., Zhang, Y., Shao, M., Zhou, H., & Yan, M. (2017). Stacked kernel network. *arXiv preprint arXiv:1711.09219*.

Zhou, Z.-H., & Feng, J. (2017). Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835*.

Zhuang, J., Tsang, I. W., & Hoi, S. C. (2011). Two-layer multiple kernel learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 909–917).

Appendix A Experimental Setup

The data set *rectangles* has 1000 training images, 200 validation images [3], and 50000 test images. The model is required to tell if a rectangle contained in an image has a larger width or length. The location of the rectangle is random. The border of the rectangle has pixel value 255 and pixels in the rest of an image all have value 0. *rectangles-image* is the same as *rectangles* except that the inside and outside of the rectangle are replaced by an image patch, respectively. *rectangles-image* has 10000 training images, 2000 validation images, and 50000 test images. *convex* consists of images in which there are white regions (pixel value 255) on black (pixel value 0) background. The model needs to tell if the region is convex. This data set has 6000 training images, 2000 validation images, and 50000 test images. *mnist (50k test)* contains 10000 training images, 2000 validation images, and 50000 test images taken from the standard MNIST. *mnist (50k test) rotated* is the same as the fourth except that the digits have been randomly rotated. For detailed descriptions of the data sets, see (Larochelle et al., 2007).

The experimental setup for the greedily-trained kMLPs is as follows, kMLP-1 corresponds to a one-hidden-layer kMLP with the first layer consisting of 15 to 150 kernel machines using the same Gaussian kernel and the second layer being a single or ten (depending on the number of classes) kernel machines using another Gaussian kernel. Hyperparameters were selected using the validation set. The validation set was then used in final training only for early-stopping based on validation error. For the standard MNIST and Fashion-MNIST, the last 5000 training examples were held out as validation set. kMLP-1^{FAST} is the same kMLP for which we accelerated by randomly

[3] The last 200 of the training set. Same for other datasets as well.

choosing a subset of the training set as centers for the second layer after the first had been trained. The kMLP-2 and kMLP-2^{FAST} are the two-hidden-layer kMLPs, the second hidden layers of which contained 15 to 150 kernel machines. Settings of all the kMLPs trained with BP can be found in (Zhang et al., 2017). Note that because it is extremely time/memory-consuming to train kMLP-2 with BP without any acceleration method, to make training possible, we could only randomly use 10000 examples from the entire training set of 55000 examples as centers for the kMLP-2 (BP) from Table 2.

In Table 3, we compared kMLP with a one/two-hidden-layer MLP (MLP-1/MLP-2), a one/three-hidden-layer DBN (DBN-1/DBN-3) and a three-hidden-layer SAE (SAE-3). For these models, hyperparameters were also selected using the validation set. For the MLPs, the sizes of the hidden layers were chosen from the interval [25, 700]. All hyperparameters involved in Adam, RMSProp and BN were set to the suggested default values in the corresponding papers. If used, dropout or BN was added to the hidden layers and the best probability for dropout was found using the validation set. For DBN-3 and SAE-3, the sizes of the three hidden layers varied in intervals [500, 3000], [500, 4000] and [1000, 6000], respectively. DBN-1 used a much larger hidden layer than DBN-3 to obtain comparable performance. A simple calculation shows that the total numbers of parameters in the kMLPs were fewer than those in the corresponding DBNs and SAEs by orders of magnitude in all experiments. Like in the training for the kMLPs, the validation set were also reserved for early-stopping in final training. The DBNs and SAEs had been pre-trained unsupervisedly before the supervised training phase, following the algorithms described in (Hinton et al., 2006; Bengio et al., 2007). More detailed settings for these models were reported in (Larochelle et al., 2007).

Appendix B Proofs

Lemma B.1. *Suppose $f_1 \in \mathbb{F}_1, \ldots, f_d \in \mathbb{F}_d$ are elements from sets of real-valued functions defined on $\mathbb{R}^p$ for some $p \geq 1$, $\mathbb{F} \subset \mathbb{F}_1 \times \cdots \times \mathbb{F}_d$ is a subset of their direct sum. For $\mathbf{f} \in \mathbb{F}$, define $\omega \circ \mathbf{f} : \mathbb{R}^p \times \cdots \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ as $(\mathbf{x}_1, \ldots, \mathbf{x}_m, \mathbf{y}) \mapsto \omega(f_1(\mathbf{x}_1), \ldots, f_d(\mathbf{x}_1), f_1(\mathbf{x}_2), \ldots, f_d(\mathbf{x}_m), \mathbf{y})$, where $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^p$, $\mathbf{y} \in \mathbb{R}^q$, and $\omega : \mathbb{R}^{md} \times \mathbb{R}^q \to \mathbb{R}$ is bounded and L-Lipschitz for each $\mathbf{y} \in \mathbb{R}^q$ with respect to the Euclidean metric on $\mathbb{R}^{md}$. Let $\omega \circ \mathbb{F} = \{\omega \circ \mathbf{f} : \mathbf{f} \in \mathbb{F}\}$.*

Define

$$\mathcal{G}_N^j(\mathbb{F}_i) = \mathbb{E}_{Z_n, \mathbf{X}_n^j}\left[\sup_{f \in \mathbb{F}_i} \frac{1}{N} \sum_{n=1}^{N} Z_n f\left(\mathbf{X}_n^j\right)\right], \; i = 1, \ldots, d, \; j = 1, \ldots, m,$$

where the $\mathbf{X}_n^j$ are i.i.d. random vectors defined on $\mathbb{R}^p$. We have

$$\mathcal{G}_N(\omega \circ \mathbb{F}) \leq 2L \sum_{i=1}^{d} \sum_{j=1}^{m} \mathcal{G}_N^j(\mathbb{F}_i). \tag{3}$$

In particular, if for all j, the $\mathbf{X}_n^j$ upon which the Gaussian complexities of the $\mathbb{F}_i$ are evaluated are sets of i.i.d. random vectors with the same distribution, we have $\mathcal{G}_N^1(\mathbb{F}_i) = \cdots = \mathcal{G}_N^m(\mathbb{F}_i) =: \mathcal{G}_N(\mathbb{F}_i)$ for all i and Eq. 3 becomes

$$\mathcal{G}_N(\omega \circ \mathbb{F}) \leq 2mL \sum_{i=1}^{d} \mathcal{G}_N(\mathbb{F}_i).$$

This lemma is a generalization of a result on the Gaussian complexity of Lipschitz functions on $\mathbb{R}^k$ from (Bartlett & Mendelson, 2002). And the technique used in the following proof is also adapted from there.

Proof. For the sake of brevity, we prove the case where $m = 2$. The general case uses exactly the same technique except that the notations would be more cumbersome.

Let $\mathbb{F}$ be indexed by $\mathcal{A}$. Without loss of generality, assume $|\mathcal{A}| < \infty$. Define

$$T_\alpha = \sum_{n=1}^{N} \omega(f_{\alpha,1}(\mathbf{X}_n),\, \ldots,\, f_{\alpha,d}(\mathbf{X}'_n),\, \mathbf{Y}_n) Z_n;$$
$$V_\alpha = L \sum_{n=1}^{N} \sum_{i=1}^{d} (f_{\alpha,i}(\mathbf{X}_n) Z_{n,i} + f_{\alpha,i}(\mathbf{X}'_n) Z_{N+n,i}),$$

where $\alpha \in \mathcal{A}$, $\{(\mathbf{X}_n,\, \mathbf{X}'_n) :\, n = 1,\, \ldots,\, N\}$ is a random sample of size N on $\mathbb{R}^p \times \mathbb{R}^p$ and $Z_1,\, \ldots,\, Z_N,\, Z_{1,1},\, \ldots,\, Z_{2N,d}$ are i.i.d. standard normal random variables.

Let arbitrary $\alpha,\, \beta \in \mathcal{A}$ be given, define $\|T_\alpha - T_\beta\|_2^2 = \mathbb{E}\,(T_\alpha - T_\beta)^2$, where the expectation is taken over the Z_n. Define $\|V_\alpha - V_\beta\|_2^2$ similarly and we have

$$\begin{aligned}
\|T_\alpha - T_\beta\|_2^2 &= \sum_{n=1}^{N} (\omega(f_{\alpha,1}(\mathbf{X}_n),\, \ldots,\, f_{\alpha,d}(\mathbf{X}'_n),\, \mathbf{Y}_n) - \omega(f_{\beta,1}(\mathbf{X}_n),\, \ldots,\, f_{\beta,d}(\mathbf{X}'_n),\, \mathbf{Y}_n))^2 \\
&\leq L^2 \sum_{n=1}^{N} \sum_{i=1}^{d} \Big((f_{\alpha,i}(\mathbf{X}_n) - f_{\beta,i}(\mathbf{X}_n))^2 + (f_{\alpha,i}(\mathbf{X}'_n) - f_{\beta,i}(\mathbf{X}'_n))^2 \Big) \\
&= \|V_\alpha - V_\beta\|_2^2.
\end{aligned}$$

By Slepian's lemma (Pisier, 1999),

$$\begin{aligned}
N\hat{\mathcal{G}}_N(\omega \circ \mathbb{F}) &= \mathbb{E}_{Z_n} \sup_{\alpha \in \mathcal{A}} T_\alpha \\
&\leq 2\mathbb{E}_{Z_{n,i},\, Z_{N+n,i}} \sup_{\alpha \in \mathcal{A}} V_\alpha \\
&\leq N2L \sum_{i=1}^{d} \Big(\hat{\mathcal{G}}_N(\mathbb{F}_i) + \hat{\mathcal{G}}'_N(\mathbb{F}_i) \Big).
\end{aligned}$$

Taking the expectation of the $\mathbf{X}_n,\, \mathbf{X}'_n,\, \mathbf{Y}_n$ on both sides proves the result. □

Lemma B.2. *Given kernel $k : \mathbb{R}^{d_1} \times \mathbb{R}^{d_1} \to \mathbb{R}$, let*

$$\mathbb{F}_2 = \left\{ f : \mathbb{R}^{d_1} \to \mathbb{R},\, f(\mathbf{x}) = \sum_{\nu=1}^{m} \alpha_\nu k(\mathbf{x}_\nu,\, \mathbf{x}) + b \,|\, \boldsymbol{\alpha} = (\alpha_1,\, \ldots,\, \alpha_m) \in \mathbb{R}^m,\, \|\boldsymbol{\alpha}\|_1 \leq A,\, b \in \mathbb{R} \right\},$$

where the $\mathbf{x}_\nu$ are an m-subset of $S_{\mathbf{X}}$.

Define $\mathbb{F}_1 = \big\{(f_1, \ldots, f_{d_1}) \colon \mathbf{x} \mapsto (f_1(\mathbf{x}), \ldots, f_{d_1}(\mathbf{x})) \,|\, \mathbf{x} \in \mathbb{R}^{d_0},\ f_j \in \Omega\big\}$*, where* Ω *is a given hypothesis class of real-valued functions on* $\mathbb{R}^{d_0}$*.*

Also, define

$$\mathbb{F}_2 \circ \mathbb{F}_1 = \left\{ h : \mathbf{x} \mapsto \sum_{\nu=1}^{m} \alpha_\nu k(\mathbf{F}(\mathbf{x}_\nu),\, \mathbf{F}(\mathbf{x})) + b \,|\, \mathbf{x} \in \mathbb{R}^{d_0},\ \|\boldsymbol{\alpha}\|_1 \le A,\ b \in \mathbb{R},\ \mathbf{F} \in \mathbb{F}_1 \right\}.$$

We have

$$\mathcal{G}_N(\mathbb{F}_2 \circ \mathbb{F}_1) \le 2ALd_1 \mathcal{G}_N(\Omega).$$

Proof. First, note that the bias b does not change $\mathcal{G}_N(\mathbb{F}_2 \circ \mathbb{F}_1)$.

$$\begin{aligned}
\hat{\mathcal{G}}_N(\mathbb{F}_2 \circ \mathbb{F}_1) &= \mathbb{E} \sup_{\boldsymbol{\alpha}, \mathbf{F}} \frac{1}{N} \sum_{n=1}^{N} \sum_{\nu=1}^{m} \alpha_\nu k(\mathbf{F}(\mathbf{x}_\nu),\, \mathbf{F}(\mathbf{x}_n)) Z_n \\
&\le \mathbb{E} \sup_{\boldsymbol{\alpha}, \mathbf{F}, \mathbf{y}_\nu \in \mathbb{R}^{d_1}} \frac{1}{N} \sum_{n=1}^{N} \sum_{\nu=1}^{m} \alpha_\nu k(\mathbf{y}_\nu,\, \mathbf{F}(\mathbf{x}_n)) Z_n.
\end{aligned}$$

Suppose the supremum over $\mathbf{y}_\nu$ is attained at $\mathbf{Y}_\nu$, the $\mathbf{Y}_\nu$ are random vectors as they are functions of the Z_n.

Write

$$\begin{aligned}
g_\nu \circ \mathbf{F}(\mathbf{x}) &= k(\mathbf{F}(\mathbf{x}),\, \mathbf{Y}_\nu), \\
\omega \circ \mathbf{F}(\mathbf{x}) &= \sum_{\nu=1}^{m} \alpha_\nu g_\nu \circ \mathbf{F}(\mathbf{x}) = \sum_{\nu=1}^{m} \alpha_\nu k(\mathbf{F}(\mathbf{x}),\, \mathbf{Y}_\nu).
\end{aligned}$$

Then we have

$$\begin{aligned}
\hat{\mathcal{G}}_N(\mathbb{F}_2 \circ \mathbb{F}_1) &\le \mathbb{E} \sup_{\boldsymbol{\alpha}, \mathbf{F}} \frac{1}{N} \sum_{n=1}^{N} \sum_{\nu=1}^{m} \alpha_\nu k(\mathbf{Y}_\nu,\, \mathbf{F}(\mathbf{x}_n)) Z_n \\
&= \mathbb{E} \sup_{\boldsymbol{\alpha}, \mathbf{F}} \frac{1}{N} \sum_{n=1}^{N} \omega \circ \mathbf{F}(\mathbf{x}) Z_n \\
&= \hat{\mathcal{G}}_N(\omega \circ \mathbb{F}_1).
\end{aligned}$$

We now prove a Lipschitz property for ω. For any $\boldsymbol{\xi}_1,\ \boldsymbol{\xi}_2 \in \mathbb{R}^{d_1}$, we have

$$\begin{aligned}
|\omega(\boldsymbol{\xi}_1) - \omega(\boldsymbol{\xi}_2)| &= \left| \sum_{\nu=1}^{m} \alpha_\nu (g_\nu(\boldsymbol{\xi}_1) - g_\nu(\boldsymbol{\xi}_2)) \right| \\
&\leq \sum_{\nu=1}^{m} |\alpha_\nu| |g_\nu(\boldsymbol{\xi}_1) - g_\nu(\boldsymbol{\xi}_2)| \\
&\leq A \max_\nu |g_\nu(\boldsymbol{\xi}_1) - g_\nu(\boldsymbol{\xi}_2)| \\
&= A \max_\nu |k(\boldsymbol{\xi}_1,\ \mathbf{Y}_\nu) - k(\boldsymbol{\xi}_2,\ \mathbf{Y}_\nu)| \\
&\leq A \max_\nu L_{\mathbf{Y}_\nu} \|\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2\|_2 \\
&\leq AL \|\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2\|_2.
\end{aligned}$$

Therefore, $\omega \circ \mathbf{F}(\mathbf{x})$, as a function of $\mathbf{F}(\mathbf{x})$, is Lipschitz w.r.t. the Euclidean metric on $\mathbb{R}^{d_1}$ with Lipschitz constant at most AL. It is easy to check that ω is bounded. Now the desired result follows from Lemma B.1. □

Proof of Proposition 3.2. The result follows from repeatedly applying Lemma B.2. □

Proof of Lemma 4.1. Let $\mathbf{G}_1 = \arg\min_{\mathbf{F}_1 \in \mathbb{F}_1'} \min_{\mathbf{F}_2 \in \mathbb{F}_2'} \tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1)$, $\mathbf{G}_2 = \arg\min_{\mathbf{F}_2 \in \mathbb{F}_2'} \tilde{R}(\mathbf{F}_2 \circ \mathbf{G}_1)$.

Suppose $\mathbf{F}_1^\star \neq \mathbf{G}_1$,

$$\begin{aligned}
\tilde{R}(\mathbf{G}_2 \circ \mathbf{G}_1) &= \min_{\mathbf{F}_2 \in \mathbb{F}_2'} \tilde{R}(\mathbf{F}_2 \circ \mathbf{G}_1) && \text{(definition of } \mathbf{G}_2\text{)} \\
&< \min_{\mathbf{F}_2 \in \mathbb{F}_2'} \tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1^\star) && \text{(definition of } \mathbf{G}_1 \text{ and } \mathbf{F}_1^\star \neq \mathbf{G}_1\text{)} \\
&= \tilde{R}(\mathbf{F}_2^\star \circ \mathbf{F}_1^\star). && \text{(definition of } \mathbf{F}_2^\star\text{)}
\end{aligned}$$

However, this contradicts the optimality of $\mathbf{F}_2^\star \circ \mathbf{F}_1^\star$. □

Proof of Theorem 4.2. Let $\mathbb{F}_1'$ be the class of all $\mathbf{F}_1'$ such that for any $f_2 \in \arg\min_{f_2 \in \mathbb{F}_2} \tilde{R}(f_2 \circ \mathbf{F}_1')$, there exist $(\mathbf{x}_+,\ y_+),\ (\mathbf{x}_-,\ y_-) \in S$ such that $\ell(f_2 \circ \mathbf{F}_1',\ (\mathbf{x}_n,\ y_n)) = 0$, $n = +,\ -$.

Observe that $\mathbf{F}_1^\star \in \mathbb{F}_1'$ since any $f_2 \in \arg\min_{f_2 \in \mathbb{F}_2} \tilde{R}(f_2 \circ \mathbf{F}_1^\star)$ is easily shown to be $f_2^\star$.

Now, suppose $\mathbf{F}_1^\circ$ satisfies Eq. 1, if we show $\mathbf{F}_1^\circ \in \mathbb{F}_1'$ and that for any $\mathbf{F}_1' \in \mathbb{F}_1'$, we have

$$\min_{f_2 \in \mathbb{F}_2} \tilde{R}(f_2 \circ \mathbf{F}_1') \geq \min_{f_2 \in \mathbb{F}_2} \tilde{R}(f_2 \circ \mathbf{F}_1^\circ),$$

then by Lemma 4.1, $\mathbf{F}_1^\circ = \mathbf{F}_1^\star$.

We now start the formal proof. Note that we drop the layer indices 1 and 2 for brevity, which will cause no confusion since the output layer will be denoted by f and the input layer $\mathbf{F}$. We assume that $\mathbf{F}^\circ$ satisfies Eq. 1. Let $f^\circ \in \arg\min_{f \in \mathbb{F}_2} \tilde{R}(f \circ \mathbf{F}^\circ)$. Let $\mathbf{F}' \in \mathbb{F}_1'$ be given and also let $f' \in \arg\min_{f \in \mathbb{F}_2} \tilde{R}(f \circ \mathbf{F}')$.

Claim 1.

$$\|\phi(\mathbf{x})\|_H = \sqrt{c},\ \forall \mathbf{x} \in \mathbb{R}^{d_1}.$$

Proof of Claim 1.

$$c = k(\mathbf{x},\, \mathbf{x}) = \langle \phi(\mathbf{x}),\, \phi(\mathbf{x}) \rangle_H = \|\phi(\mathbf{x})\|_H^2,$$

which implies $\|\phi(\mathbf{x})\|_H = \sqrt{c}$. ■

Claim 2.

$$\phi(\mathbf{F}^\circ(\mathbf{x})) = \phi(\mathbf{F}^\circ(\mathbf{x}')), \quad \forall \mathbf{x},\, \mathbf{x}' \in S_{\mathbf{X}} \textit{ with } y = y';$$

$$\phi(\mathbf{F}^\circ(\mathbf{x}_+)) = \phi(\mathbf{F}^\circ(\mathbf{x}_-)), \quad \forall \mathbf{x}_+,\, \mathbf{x}_- \in S_{\mathbf{X}}.$$

Proof of Claim 2. By Cauchy-Schwarz inequality and Claim 1,

$$0 < c = k(\mathbf{F}^\circ(\mathbf{x}),\, \mathbf{F}^\circ(\mathbf{x}')) = \langle \phi(\mathbf{F}^\circ(\mathbf{x})),\, \phi(\mathbf{F}^\circ(\mathbf{x}')) \rangle_H \leq \|\phi(\mathbf{F}^\circ(\mathbf{x}))\|_H \|\phi(\mathbf{F}^\circ(\mathbf{x}'))\|_H = c.$$

So the equality holds in Cauchy-Schwarz and we have $\phi(\mathbf{F}^\circ(\mathbf{x})) = p\phi(\mathbf{F}^\circ(\mathbf{x}'))$ for some $p > 0$. Again by Claim 1, $p = 1$.

The second part of this claim follows from $k(\mathbf{F}^\circ(\mathbf{x}_+),\, \mathbf{F}^\circ(\mathbf{x}_-)) = a \neq c$. ■

Claim 3. *For any* $\mathbf{x}_+,\, \mathbf{x}_- \in S_{\mathbf{X}}$ *and any* $\mathbf{F} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}$,

$$\sqrt{2(c-a)} = \|\phi(\mathbf{F}^\circ(\mathbf{x}_+)) - \phi(\mathbf{F}^\circ(\mathbf{x}_-))\|_H \geq \|\phi(\mathbf{F}(\mathbf{x}_+)) - \phi(\mathbf{F}(\mathbf{x}_-))\|_H.$$

Proof of Claim 3.

$$\begin{aligned}
&\|\phi(\mathbf{F}^\circ(\mathbf{x}_+)) - \phi(\mathbf{F}^\circ(\mathbf{x}_-))\|_H^2 \\
&\quad = \|\phi(\mathbf{F}^\circ(\mathbf{x}_+))\|_H^2 + \|\phi(\mathbf{F}^\circ(\mathbf{x}_-))\|_H^2 - 2\langle \phi(\mathbf{F}^\circ(\mathbf{x}_+)),\, \phi(\mathbf{F}^\circ(\mathbf{x}_-))\rangle_H \\
&\quad = c + c - 2k(\mathbf{F}^\circ(\mathbf{x}_+),\, \mathbf{F}^\circ(\mathbf{x}_-)) \\
&\quad = 2c - 2a \\
&\quad \geq 2c - 2k(\mathbf{F}(\mathbf{x}_+),\, \mathbf{F}(\mathbf{x}_-)) \\
&\quad = \|\phi(\mathbf{F}(\mathbf{x}_+))\|_H^2 + \|\phi(\mathbf{F}(\mathbf{x}_-))\|_H^2 - 2\langle \phi(\mathbf{F}(\mathbf{x}_+)),\, \phi(\mathbf{F}(\mathbf{x}_-))\rangle_H \\
&\quad = \|\phi(\mathbf{F}(\mathbf{x}_+)) - \phi(\mathbf{F}(\mathbf{x}_-))\|_H^2
\end{aligned}$$

■

Claim 4.

$$\tilde{R}(f^\circ \circ \mathbf{F}^\circ) = \tau\|\mathbf{w}_{f^\circ}\|_H = \frac{2\tau}{\|\phi(\mathbf{F}^\circ(\mathbf{x}_+)) - \phi(\mathbf{F}^\circ(\mathbf{x}_-))\|_H},\ \forall \mathbf{x}_+,\, \mathbf{x}_- \in S_{\mathbf{X}}.$$

Proof of Claim 4.

$$\begin{aligned}
&\tilde{R}(f^\circ \circ \mathbf{F}^\circ) \\
&\quad = \frac{1}{N}\sum_{n=1}^{N} \max(0,\, 1 - y_n f^\circ(\mathbf{F}^\circ(\mathbf{x}_n))) + \tau \|\mathbf{w}_{f^\circ}\|_H \\
&\quad = \kappa \max(0,\, 1 - y_+ f^\circ(\mathbf{F}^\circ(\mathbf{x}_+))) + (1-\kappa)\max(0,\, 1 - y_- f^\circ(\mathbf{F}^\circ(\mathbf{x}_-))) + \tau \|\mathbf{w}_{f^\circ}\|_H,
\end{aligned}$$

for any pair of $\mathbf{x}_+,\, \mathbf{x}_- \in S_{\mathbf{X}}$. Let

$$\zeta_{f^\circ} = y_+ f^\circ(\mathbf{F}^\circ(\mathbf{x}_+)) + y_- f^\circ(\mathbf{F}^\circ(\mathbf{x}_-)) = \|\mathbf{w}_{f^\circ}\|_H \|\phi(\mathbf{F}^\circ(\mathbf{x}_+)) - \phi(\mathbf{F}^\circ(\mathbf{x}_-))\|_H \cos\theta_{f^\circ}.$$

We have

$$\tilde{R}(f^\circ \circ \mathbf{F}^\circ) = \kappa \max(0,\, 1 - t_{f^\circ}) + (1-\kappa)\max(0,\, 1 - (\zeta_{f^\circ} - t_{f^\circ})) + \tau \|\mathbf{w}_{f^\circ}\|_H,$$

where $t_{f^\circ} = f^\circ(\mathbf{F}^\circ(\mathbf{x}_+))$.

Note that by definition of f°,

$$\begin{aligned}
\tilde{R}(f^\circ \circ \mathbf{F}^\circ) &= \min_f \tilde{R}(f \circ \mathbf{F}^\circ) \\
&= \min_{\zeta_f,\, t_f,\, \|\mathbf{w}_f\|_H} \kappa \max(0,\, 1 - t_f) + (1-\kappa)\max(0,\, 1 - (\zeta_f - t_f)) + \tau \|\mathbf{w}_f\|_H.
\end{aligned}$$

There are four possible cases that the terms inside the minimum operator can be simplified to:

(1) If $1 \geq t_f \geq \zeta_f - 1,\, \zeta_f \leq 2$, to $1 - (1-\kappa)\zeta_f + (1-2\kappa)t_f + \tau \|\mathbf{w}_f\|_H$;

(2) If $t_f \geq \max(1,\, \zeta_f - 1)$, to $(1-\kappa)(1 - \zeta_f + t_f) + \tau \|\mathbf{w}_f\|_H$;

(3) If $t_f \leq \min(1,\, \zeta_f - 1)$, to $\kappa(1 - t_f) + \tau \|\mathbf{w}_f\|_H$;

(4) If $1 \leq t_f \leq \zeta_f - 1$, $\zeta_f \geq 2$, to $\tau \|\mathbf{w}_f\|_H$.

If $\zeta_f \geq 2$, for each fixed ζ_f, $\|\mathbf{w}_f\|_H$, t_f, we have that the values of $\tilde{R}$ in (2), (3) are no less than that in (4) and that their minima agree. Therefore, when $\zeta_f \geq 2$, $\tilde{R}(f^\circ \circ \mathbf{F}^\circ) = \tau \|\mathbf{w}_{f^\circ}\|_H$.

On the other hand, if $\zeta_f \leq 2$, then for each fixed ζ_f, $\|\mathbf{w}_f\|_H$, first note $t_f \in \mathbb{R}$ can be chosen freely by adjusting b. Also, since $\max(1, \zeta_f - 1) = 1$ and $\min(1, \zeta_f - 1) = \zeta - 1$, by working out the minima over t_f in (1), (2), and (3), respectively, we have $\tilde{R}(f^\circ \circ \mathbf{F}^\circ) = \min(\kappa, 1-\kappa)(2 - \zeta_{f^\circ}) + \tau \|\mathbf{w}_{f^\circ}\|_H$.

Note that we have $\zeta_{f^\circ} = \|\mathbf{w}_{f^\circ}\|_H \psi^\circ \cos\theta_{f^\circ}$, where $\psi^\circ = \|\phi(\mathbf{F}^\circ(\mathbf{x}_+)) - \phi(\mathbf{F}^\circ(\mathbf{x}_-))\|_H$, we can rewrite the earlier result in terms of ζ_{f° and $\cos\theta_{f^\circ}$. Consequently, we now determine the minimum over ζ_f and $\cos\theta_f$ of the resulting expression.

To this end, first observe that for each ζ_f, one can choose $\cos\theta_f \in [-1, 1]$ freely by adjusting $\|\mathbf{w}_f\|_H$ under the constraint that the two quantities must be of the same sign, if both are nonzero. Therefore, for each $\zeta_f \geq 2$,

$$\min_{\cos\theta_f} \tilde{R}(f \circ \mathbf{F}^\circ) = \tilde{R}(f \circ \mathbf{F}^\circ) \mid_{\cos\theta_f = 1} = \frac{\tau \zeta_f}{\psi^\circ}.$$

Similarly, for each $\zeta_f \leq 2$, we have $\min_{\cos\theta_f} \tilde{R}(f \circ \mathbf{F}^\circ) = \min(\kappa, 1-\kappa)(2 - \zeta_f) + \tau|\zeta_f|/\psi^\circ$.

Combining these two cases and using the assumption on τ, it is easy to see that $\tilde{R}(f^\circ \circ \mathbf{F}^\circ) = \min_{\zeta_f} \tilde{R}(f \circ \mathbf{F}^\circ) = \tilde{R}(f \circ \mathbf{F}^\circ) \mid_{\zeta_f = 2} = 2\tau/\psi^\circ$. This proves the claim.

■

Remark B.2.1. *By Claim 4,* $\mathbf{F}^\circ \in \mathbb{F}_1'$.

Claim 5. *For any* $\mathbf{F}' \in \mathbb{F}_1'$, $\min_{f \in \mathbb{F}_2} \tilde{R}\big(f \circ \mathbf{F}'\big) \geq \min_{f \in \mathbb{F}_2} \tilde{R}(f \circ \mathbf{F}^\circ)$.

Proof of Claim 5. By Claim 4, it amounts to prove

$$\tilde{R}\Big(f' \circ \mathbf{F}^{'}\Big) \geq \frac{2\tau}{\|\phi(\mathbf{F}^{\circ}(\mathbf{x}_{+})) - \phi(\mathbf{F}^{\circ}(\mathbf{x}_{-}))\|_{H}},$$

for an arbitrary pair of $\mathbf{x}_{+},\ \mathbf{x}_{-} \in S_{\mathbf{X}}$. Suppose $(\mathbf{x}_{+}^{'},\ y_{+}^{'})$, $(\mathbf{x}_{-}^{'},\ y_{-}^{'})$ are a pair of examples with $\mathbf{x}_{+}^{'},\ \mathbf{x}_{-}^{'} \in S_{\mathbf{X}}$, and $\ell\big(f' \circ \mathbf{F}^{'},\ (\mathbf{x}_{n}^{'},\ y_{n}^{'})\big) = 0,\ n = +,\ -$, then we have $y_{+}^{'} f^{'}\big(\mathbf{x}_{+}^{'}\big) + y_{-}^{'} f^{'}\big(\mathbf{x}_{-}^{'}\big) \geq 2$.

Since $y_{+}^{'} f^{'}\big(\mathbf{x}_{+}^{'}\big) + y_{-}^{'} f^{'}\big(\mathbf{x}_{-}^{'}\big) = \|\mathbf{w}_{f'}\|_{H} \big\|\phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{+}^{'}\big)\big) - \phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{-}^{'}\big)\big)\big\|_{H} \cos\theta_{f'}$, it is implied that $\cos\theta_{f'} \in (0,\ 1]$, $\big\|\phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{+}^{'}\big)\big) - \phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{-}^{'}\big)\big)\big\|_{H} > 0$ and $\|\mathbf{w}_{f'}\|_{H} \geq 2/\big\|\phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{+}^{'}\big)\big) - \phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{-}^{'}\big)\big)\big\|_{H}$. Therefore,

$$\begin{aligned}
\tilde{R}\Big(f' \circ \mathbf{F}^{'}\Big) &\geq \tau\|\mathbf{w}_{f'}\|_{H} \\
&\geq \frac{2\tau}{\big\|\phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{+}^{'}\big)\big) - \phi\big(\mathbf{F}^{'}\big(\mathbf{x}_{-}^{'}\big)\big)\big\|_{H}} \\
&\geq \frac{2\tau}{\big\|\phi\big(\mathbf{F}^{\circ}\big(\mathbf{x}_{+}^{'}\big)\big) - \phi\big(\mathbf{F}^{\circ}\big(\mathbf{x}_{-}^{'}\big)\big)\big\|_{H}} \\
&= \tilde{R}(f^{\circ} \circ \mathbf{F}^{\circ}).
\end{aligned}$$

■

This concludes the proof of the theorem. □

Proof of Theorem 4.3. Denote with $\mathbb{F}_{2}^{\prime}$ the set of all $\mathbf{F}_{2}^{\prime}$ such that for all j, $\Big\|\mathbf{w}_{f_{2}^{j'}}\Big\|_{H} > 0$.

Denote with $\mathbb{F}_{1}^{\prime}$ the set of all $\mathbf{F}_{1}^{\prime}$ such that for any $\mathbf{F}_{2} \in \arg\min_{\mathbf{F}_{2} \in \mathbb{F}_{2}^{\prime}} \tilde{R}(\mathbf{F}_{2} \circ \mathbf{F}_{1}^{\prime})$, $\mathbf{F}_{2}$ satisfies:

$$\exists(\mathbf{x}_{+},\ y_{+}),\ (\mathbf{x}_{-},\ y_{-}) \in S_{\mathbf{X}} \times S_{Y} \text{ s.t. } \ell(\mathbf{F}_{2} \circ \mathbf{F}_{1}^{\prime},\ (\mathbf{x}_{+},\ y_{+}),\ (\mathbf{x}_{-},\ y_{-})) = 0.$$

Using the same argument as in the beginning of the proof of Theorem 4.2, we have

$\mathbf{F}_1^\star \in \mathbb{F}_1'$. Let $\mathbf{F}_1' \in \mathbb{F}_1'$ be given and suppose $\mathbf{F}_1^\circ$ satisfies Eq. 2. Let

$$\mathbf{F}_2' \in \operatorname*{arg\,min}_{\mathbf{F}_2 \in \mathbb{F}_2'} \tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1'), \text{ and } \mathbf{F}_2^\circ \in \operatorname*{arg\,min}_{\mathbf{F}_2 \in \mathbb{F}_2'} \tilde{R}(\mathbf{F}_2 \circ \mathbf{F}_1^\circ).$$

Then by Lemma 4.1, the proof is complete if we can show $\tilde{R}(\mathbf{F}_2' \circ \mathbf{F}_1') \geq \tilde{R}(\mathbf{F}_2^\circ \circ \mathbf{F}_1^\circ)$.

To this end, first note that Claims 1, 2, 3 from the proof of Theorem 4.2 evidently hold here as well. Define $\psi = 1/N^2 \sum_{n,\,m=1}^{N} \mathbb{1}_{\{y_m \neq y_n\}}$.

Claim 6. $\mathbf{F}_2^\circ(\mathbf{x}) = \mathbf{F}_2^\circ(\mathbf{x}'), \forall \mathbf{x},\, \mathbf{x}' \in S_{\mathbf{X}}$ *with* $y = y'$.

Proof of Claim 6. $\forall \mathbf{x},\, \mathbf{x}' \in S_{\mathbf{X}}$,

$$\begin{aligned}\mathbf{F}_2^\circ(\mathbf{x}) - \mathbf{F}_2^\circ(\mathbf{x}') &= \big(f_2^{1\circ}(\mathbf{x}) - f_2^{1\circ}(\mathbf{x}'), \ldots\big) \\ &= \Big(\Big\langle \mathbf{w}_{f_2^{1\circ}},\, \phi(\mathbf{F}_1^\circ(\mathbf{x})) - \phi(\mathbf{F}_1^\circ(\mathbf{x}'))\Big\rangle_H, \ldots\Big) = (0, \ldots),\end{aligned}$$

where we have used Claim 2 for the last equality. ■

Combining this claim with our earlier assumptions on h, we can simplify the objective function

$$\tilde{R}(\mathbf{F}_2^\circ \circ \mathbf{F}_1^\circ) = \psi\Big(h\Big(\|\mathbf{F}_2^\circ(\mathbf{x}_+) - \mathbf{F}_2^\circ(\mathbf{x}_-)\|_q\Big) - b\Big)^p + \tau t\Big(\Big\|\mathbf{w}_{f_2^{1\circ}}\Big\|_H, \ldots, \Big\|\mathbf{w}_{f_2^{d_2\circ}}\Big\|_H\Big),$$

where $\mathbf{x}_+,\, \mathbf{x}_- \in S_{\mathbf{X}}$ are arbitrary.

Rewrite the above expression as

$$\begin{aligned}&\tilde{R}(\mathbf{F}_2^\circ \circ \mathbf{F}_1^\circ) \\ &= \psi\left(h\left(\left(\sum_j^{d_2} \Big\|\mathbf{w}_{f_2^{j\circ}}\Big\|_H^q \|\phi(\mathbf{F}_1^\circ(\mathbf{x}_+)) - \phi(\mathbf{F}_1^\circ(\mathbf{x}_-))\|_H^q \Big(\cos\theta_{f_2^{j\circ}}\Big)^q\right)^{1/q}\right) - b\right)^p \\ &+ t\Big(\Big\|\mathbf{w}_{f_2^{1\circ}}\Big\|_H, \ldots, \Big\|\mathbf{w}_{f_2^{d_2\circ}}\Big\|_H\Big)\end{aligned}$$

Claim 7. $\Big(\cos\theta_{f_2^{j\circ}}\Big)^2 = 1,\, \forall j$.

Proof of Claim 7. This claim follows from noting that for each $\left\|\mathbf{w}_{f_2^j}\right\|_H$, $\left(\cos\theta_{f_2^j}\right)^2$ may be chosen freely and since the $\left\|\mathbf{w}_{f_2^j}\right\|_H$ are nonzero by the definition of $\mathbb{F}'_2$, it is easy to see that the unique minimizers of the $\left(\cos\theta_{f_2^j}\right)^2$ are $\left(\cos\theta_{f_2^j}\right)^2 = 1,\ \forall j$. ■

Using Claim 3 and the above claim, we further simplify the objective function into

$$
\begin{aligned}
&\tilde{R}(\mathbf{F}_2^\circ \circ \mathbf{F}_1^\circ) \\
&\quad = \psi\left(h\left(\sqrt{2(c-a)}\left(\sum_j^{d_2}\left\|\mathbf{w}_{f_2^{j\circ}}\right\|_H^q\right)^{1/q}\right) - b\right)^p + \tau t\left(\left\|\mathbf{w}_{f_2^{1\circ}}\right\|_H, \ldots, \left\|\mathbf{w}_{f_2^{d_2\circ}}\right\|_H\right) \\
&\quad = \min_{\mathbf{w}_{f_2^j}} \psi\left(h\left(\sqrt{2(c-a)}\left(\sum_j^{d_2}\left\|\mathbf{w}_{f_2^j}\right\|_H^q\right)^{1/q}\right) - b\right)^p + \tau t\left(\left\|\mathbf{w}_{f_2^1}\right\|_H, \ldots, \left\|\mathbf{w}_{f_2^{d_2}}\right\|_H\right)
\end{aligned}
$$

Now, let $\left(\mathbf{x}'_+, \mathbf{x}'_-\right) \in \left\{\arg\max_{\mathbf{x}_+, \mathbf{x}_- \in S_\mathbf{X}} \|\mathbf{F}'_2(\mathbf{x}_+) - \mathbf{F}'_2(\mathbf{x}_-)\|_q\right\}$, we have

$$
\begin{aligned}
&\tilde{R}(\mathbf{F}'_2 \circ \mathbf{F}'_1) \\
&\quad \geq \psi\left(h\left(\left\|\mathbf{F}'_2\left(\mathbf{x}'_+\right) - \mathbf{F}'_2\left(\mathbf{x}'_-\right)\right\|_q\right) - b\right)^p + \tau t\left(\left\|\mathbf{w}_{f_2^{1\prime}}\right\|_H, \ldots, \left\|\mathbf{w}_{f_2^{d_2\prime}}\right\|_H\right) \\
&\quad \geq \psi\left(h\left(\left\|\phi\left(\mathbf{F}'_1\left(\mathbf{x}'_+\right)\right) - \phi\left(\mathbf{F}'_1\left(\mathbf{x}'_-\right)\right)\right\|_H\left(\sum_{j=1}^{d_2}\left\|\mathbf{w}_{f_2^{j\prime}}\right\|_H^q\right)^{1/q}\right) - b\right)^p \\
&\qquad + \tau t\left(\left\|\mathbf{w}_{f_2^{1\prime}}\right\|_H, \ldots, \left\|\mathbf{w}_{f_2^{d_2\prime}}\right\|_H\right) \\
&\quad \geq \psi\left(h\left(\sqrt{2(c-a)}\left(\sum_{j=1}^{d_2}\left\|\mathbf{w}_{f_2^{j\prime}}\right\|_H^q\right)^{1/q}\right) - b\right)^p + \tau t\left(\left\|\mathbf{w}_{f_2^{1\prime}}\right\|_H, \ldots, \left\|\mathbf{w}_{f_2^{d_2\prime}}\right\|_H\right) \\
&\quad \geq \tilde{R}(\mathbf{F}_2^\circ \circ \mathbf{F}_1^\circ).
\end{aligned}
$$

□