# DEEP KERNEL MACHINES VIA THE KERNEL REPARAMETRIZATION TRICK

**Jovana Mitrovic, Dino Sejdinovic & Yee Whye Teh**
Department of Statistics
University of Oxford
Oxford, OX1 3LB, UK
{mitrovic,dino.sejdinovic,y.w.teh}@stats.ox.ac.uk

## ABSTRACT

While deep neural networks have achieved state-of-the-art performance on many tasks across varied domains, they still remain black boxes whose inner workings are hard to interpret and understand. In this paper, we develop a novel method for efficiently capturing the behaviour of deep neural networks using kernels. In particular, we construct a hierarchy of increasingly complex kernels that encode individual hidden layers of the network. Furthermore, we discuss how our framework motivates a novel supervised weight initialization method that discovers highly discriminative features already at initialization.

## 1 INTRODUCTION

One of the first connections between kernels and neural networks was established by Neal (1996). There, it is shown that if the covariance kernel of a Gaussian process (GP) (Rasmussen, 2006) is chosen based on the weight prior of an infinite-width neural network, then these two models induce the same prior over functions. Recently, there has been a resurgence of interest in the connections between neural networks and kernel methods. For example, Cho & Saul (2009; 2010) construct kernels that mimic computations in neural networks with a particular form of non-linearity, while Montavon et al. (2011) analyze neural networks with kernels. Other related work includes drawing connections between convolutional neural networks and kernels (Mairal et al., 2014), constructing kernels for two-layer infinite-width neural networks with arbitrary non-linearities (Hazan & Jaakkola, 2015) and characterizing the duality in expressivity between compositional kernels and neural networks (Daniely et al., 2016).

In this paper, we extend the well-established connections between single layer neural networks and kernels to arbitrarily deep neural networks. In particular, we build upon the approach of (Hazan & Jaakkola, 2015). The main contributions of this paper are

- the derivation of a weight initialization scheme for infinite-width neural networks of arbitrary depth,
- the construction of a hierarchy of increasingly complex kernels that capture the behaviour of individual hidden layers, and
- the application of our framework to finite-width neural networks, in particular to the challenge of weight initialization in these networks.

## 2 DEEP INFINITE-WIDTH NEURAL NETWORKS

Infinite-width neural networks can be thought of as the limit of finite-width neural networks when the number of network weights between adjacent layers tends to infinity. Denote the first layer representation of a point $x \in \mathbb{R}^d$ with $\phi_{1,x}$, where $\phi_{1,x}(w) = f(\langle w, x\rangle_{\mathbb{R}^d})$ with weight vector $w \in \mathbb{R}^d$ connecting the input to a neuron in the first layer and $f$ the network non-linearity. For the distribution of the weights connecting the input and the first layer, we can choose any probability measure $\mu(w)$ that is defined on $\mathbb{R}^d$. By thinking of the first layer representations as a surrogate for the canonical feature mappings of a particular kernel, we can define that kernel as

$$k_1(x, x') = \langle \phi_{1,x}, \phi_{1,x'} \rangle_{L^2(\mathbb{R}^d, \mu)} = \int f(\langle w, x \rangle_{\mathbb{R}^d}) f(\langle w, x' \rangle_{\mathbb{R}^d}) \, d\mu(w)$$

with the dependence on the non-linearity $f$ suppressed when it does not lead to confusion.

In order for the second layer representation to be well defined, we need to ensure that the inner product between the first layer representation and the weights connecting the first and second layer is well defined. In particular, we need to construct a probability distribution over the space of the first layer representations. In our case, this is the space of real-valued functions defined on $\mathbb{R}^d$. We take a Gaussian process as the probability distribution over the weights connecting the first and second layer as it is a natural choice for a distribution over functions from $\mathbb{R}^d$ to $\mathbb{R}$. Thus, we define the second layer representation of $x$ as a function $\phi_{2,x}$ of draws from a GP $\nu_1$ with covariance function $C_1$. The corresponding feature maps and kernel are defined as follows

$$\phi_{2,x} : \{g : \mathbb{R}^d \to \mathbb{R}\} \to \mathbb{R} \quad \text{with} \quad \phi_{2,x}(u) = f(\langle u, \phi_{1,x} \rangle_{L^2(\mathbb{R}^d, \mu)}) \quad \text{and} \quad u \sim \nu_1 = \text{GP}(0, C_1),$$

$$k_2(x, x') = \langle \phi_{2,x}, \phi_{2,x'} \rangle_{L^2(\nu_1)} = \int f(\langle u, \phi_{1,x} \rangle_{L^2(\mathbb{R}^d, \mu)}) f(\langle u, \phi_{1,x'} \rangle_{L^2(\mathbb{R}^d, \mu)}) \, d\nu_1(u).$$

Now, as we increase the number of hidden layers in the network, we have to construct probability distributions over increasingly complex domains as the weights connecting layers $l$ and $l + 1$ have to come from the same space as the representations at layer $l$. This implies that we have to define distributions over functions of functions of functions and so on depending on the number of hidden layers previous to that layer. This is by no means a trivial undertaking. Furthermore, it is often not clear what a natural choice for these distributions should be if we go beyond two hidden layers.

In order to work with deep neural networks with infinite-width layers, we develop a novel approach for the construction of weights that can be used in networks of arbitrary depth. The main idea of our approach is to take advantage of the structure of the induced reproducing kernel Hilbert spaces (RKHS) in order to facilitate the choice of weight distributions in networks with more than two hidden layers. In particular, we reparametrize the $L_2$ inner product between hidden layer representations into an RKHS inner product between the canonical feature mappings of the induced kernel. The intuition behind the *kernel reparametrization trick* is that it allows us to identify the $L_2$ functions encoding the hidden layer representations with their smoother representers in the RKHS.

First, owing to a general result from RKHS theory (Aronszajn, 1950), we can reparametrize $k_l$ in the corresponding RKHS using canonical feature mappings $k_l(\cdot, x)$, i.e.

$$k_l(x, x') = \langle \phi_{l,x}, \phi_{l,x'} \rangle_{L^2(\nu_{l-1})} = \langle k_l(\cdot, x), k_l(\cdot, x') \rangle_{\mathcal{H}_{k_l}}. \tag{1}$$

Thus, we can identify $\phi_{l,x} \in L^2(\nu_{l-1})$ ($\nu_0 = \mu$) with a smoother function $k_l(\cdot, x) \in \mathcal{H}_{k_l}$. Second, we choose the covariance function $C_l$ in a principled way as

$$C_l(x, x') = \int k_l(x, \xi) k_l(x', \xi) \, d\mu(\xi).$$

Now, in order to extend the neural network beyond two hidden layers, we just need to sample the weights connecting layers $l$ and $l + 1$ from a GP $\nu_l$ with covariance function $C_l$. The special covariance structure we posit ensures that the weights are in the appropriate RKHS, which, in turn, ensures that the inner product between the representations at layer $l$ and the weights connecting layers $l$ and $l + 1$ is well defined. At layer $l + 1$, we again perform the *kernel reparametrization trick* on the induced kernel $k_{l+1}$. Furthermore, we identify the hidden representations at layer $l + 1$ with the canonical feature mappings induced by the kernel $k_{l+1}$. Thus, at layer $l + 1$, we have

$$\phi_{l+1,x}(u) = f(\langle u, k_l(\cdot, x) \rangle_{\mathcal{H}_{k_l}}) = f(u(x)), \quad \text{for} \quad u \sim \nu_l(u) = \text{GP}(0, C_l),$$

$$k_{l+1}(x, x') = \langle \phi_{l+1,x}, \phi_{l+1,x'} \rangle_{L^2(\nu_l)} = \langle k_{l+1}(\cdot, x), k_{l+1}(\cdot, x') \rangle_{\mathcal{H}_{k_{l+1}}} = \int f(u(x)) f(u(x')) \, d\nu_l(u),$$

where we used the reproducing property of the kernel $k_l$ for the last equality in the first line. After we have repeated the above process for all hidden layers in the neural network, we can use the resulting representation of the data in any algorithm where a feature-based representation is needed.

## 3 Application: Supervised Weight Initialization

While our contribution is theoretical in nature, we show how the insights gained from the study of deep infinite-width neural networks can be applied to standard finite-width deep neural networks. In particular, we discuss how our framework motivates a novel supervised weight initialization method.

The approach discussed in the previous section can easily be adapted to the setting of finite-width neural networks by approximating integrals with their Monte Carlo estimates and RKHS inner products with inner products between random feature expansions. While the weights construction via GPs is mathematically appealing, we focus on a more direct approach, while still preserving the required covariance structure of the weights. In particular, we set the weight vector connecting layer $l$ with neuron $i$ from layer $l+1$ to

$$u_{l,i} = \sum_{m=1}^{M_l} \alpha_{im} \hat{k}_l(\cdot, \xi_{im}^{(l)}) \quad \text{with} \quad \alpha_i \sim \mathcal{N}\left(0, \frac{1}{M_l} I\right),$$

where $\xi_{im}^{(l)} \in \mathbb{R}^d$ and $\hat{k}_l(\cdot, x)$ is the random features representation of $x$ at layer $l$. Thus, for the representation of point $x$ at layer $l+1$, we have

$$\hat{k}_{l+1}(\cdot, x) = \frac{1}{\sqrt{P_{l+1}}} \left[ f(\alpha_1^T \hat{k}_l(x, \xi_1^{(l)})), \dots f(\alpha_{P_{l+1}}^T \hat{k}_l(x, \xi_{P_{l+1}}^{(l)})) \right],$$

with $\xi_i^{(l)} = \{\xi_{im}^{(l)}\}_m$ and $P_{l+1}$ the number of neurons at layer $l+1$. From this we see that the neuron $i$ encodes the alignment of $x$ to the subspace spanned by $\xi_i^{(l)}$. Guided by the idea of disentangling factors of variation, we choose the sets $\{\xi_{im}^{(l)}\}_m$ for each neuron $i$ at layer $l$ in a supervised fashion.

### 3.1 Results

We perform some initial experiments with our supervised weight initialization scheme and compare it to four commonly used initialization schemes - Xavier (Glorot & Bengio, 2010), Xavier-Caffe ($w \sim \mathcal{N}(0, (\text{fan in})^{-1})$), Kaiming (He et al., 2015) and Heuristic (LeCun et al., 2012). In particular, we train a single layer neural network with 800 hidden units on the MNIST dataset (LeCun et al., 1998). We use the RELU non-linearity and optimize with Adam (Kingma & Ba, 2014) using the default parameters. From Table 1, we can see that our initialization method discovers highly discriminative features already at initialization. Furthermore, our method is competitive after the neural network has been fully trained.

Table 1: Classification accuracy on the MNIST test set at initialization and after training averaged over 10 runs.

| Init Method | At Initialization | After Training |
|---|---|---|
| Heuristic | $9.40 \pm 3.1$ | $96.53 \pm 0.51$ |
| Xavier | $9.53 \pm 3.5$ | $96.16 \pm 0.43$ |
| Xavier-Caffe | $9.31 \pm 2.7$ | $96.08 \pm 0.45$ |
| Kaiming | $10.19 \pm 2.6$ | $96.23 \pm 0.42$ |
| Ours | $\mathbf{75.74 \pm 2.2}$ | $96.37 \pm 0.47$ |

## 4 Discussion

In this paper, we have presented a novel method for efficiently capturing the behaviour of deep neural networks using a hierarchy of increasingly complex kernels. In particular, each kernel encodes a single layer of an infinite-width neural network. Furthermore, we apply our framework to finite-width neural networks and presented a novel supervised weight initialization method that discovers highly discriminative features already at initialization.

## Acknowledgments

JM acknowledges the financial support of The Clarendon Fund of the University of Oxford. DS's and YWT's research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) ERC grant agreement no. 617071.

## References

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

Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pp. 342–350, 2009.

Youngmin Cho and Lawrence K Saul. Large-margin classification in infinite neural networks. *Neural computation*, 22(10):2678–2697, 2010.

Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pp. 2253–2261, 2016.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

T Hazan and T Jaakkola. Steps Toward Deep Kernel Methods from Infinite Neural Networks. *arXiv preprint arXiv:1508.05133*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional Kernel Networks. *arXiv preprint arXiv:1406.3332*, 2014.

Grégoire Montavon, Mikio L. Braun, and Klaus-Robert Müller. Kernel Analysis of Deep Networks. *The Journal of Machine Learning Research*, 12:2563–2581, feb 2011.

Radford M Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pp. 29–53. Springer, 1996.

Carl Edward Rasmussen. Gaussian processes for machine learning. *MIT Press*. 2006.