Deep Neural Networks as Gaussian Processes

Jaehoon Lee[*†], **Yasaman Bahri**[*†], **Roman Novak**, **Samuel S. Schoenholz**, **Jeffrey Pennington**, **Jascha Sohl-Dickstein**

Google Brain

{jaehlee, yasamanb, romann, schsam, jpennin, jaschasd}@google.com

1 Introduction

Deep neural networks have emerged in recent years as flexible parametric models which can fit complex patterns in data. As a contrasting approach, Gaussian processes have long served as a traditional nonparametric modeling tool. In fact, a correspondence due to [11] equates these two models in the limit of infinite width and parameters drawn from a suitable prior.

Consider a deep fully-connected neural network with i.i.d. random parameters. Each scalar output of the network, an affine transformation of the final hidden layer, will be a sum of i.i.d. terms, as we discuss briefly in Sec. 2. In the limit of infinite width, the Central Limit Theorem[3] implies that the function computed by the neural network (NN) *is* a function drawn from a Gaussian process (GP). This correspondence implies that if we choose the hypothesis space to be the class of infinitely wide neural networks, an i.i.d. prior over parameters can be replaced with a corresponding GP prior over functions. As noted by [16], this substitution enables *exact* Bayesian inference for regression using neural networks. The computation requires building the necessary covariance matrices over the training and test sets and straightforward linear algebra computations.

In the case of single hidden-layer networks, the form of the kernel of this GP is well known ([11, 16]). Recently, kernel functions for multi-layer random neural networks have been developed, but only outside of a Bayesian framework. As such, previous work has not identified the correspondence between using these kernels as the covariance function for a GP and performing fully Bayesian prediction with a deep neural network.

Our contributions:[4] We describe this correspondence and develop a computationally efficient pipeline to compute the kernel. We then use the resulting GP – hereafter referred to as the Neural Network GP (NNGP) – to perform Bayesian inference for deep neural networks on MNIST and CIFAR-10 across different hyperparameters including network depth, nonlinearity, training set size (up to and including the full dataset consisting of tens of thousands of images), and weight and bias variance. To utilize the exact Bayesian results for regression, we treat classification as regression on one-hot targets; our intention here is to provide a first proof of concept for our method, and we leave a fully Bayesian approach to classification for later work. We find that the GP-based predictions are competitive and can outperform neural networks trained with stochastic gradient descent (Table 1). We further observe that, with increasing network width, the performance of neural networks with gradient-based training approaches that of the NNGP computation (Figure 1), and that the GP uncertainty is strongly correlated with prediction error (Figure 2). Our observations can also be connected to the recent understanding of signal propagation in random neural networks [15].

[*] Both authors contributed equally to this work.

[†] Work done as a member of the Google AI Residency program (g.co/airesidency).

[3] Throughout this work, we assume the conditions on the parameter distributions and nonlinearities are such that the Central Limit Theorem will hold.

[4] We note that, this is extended abstract for the paper [10]. More details and results are available there.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

Related work: Our work touches on aspects of GPs, Bayesian learning, and compositional kernels. We are not aware of prior work treating deep neural networks and GPs in the manner we do. There is related work on compositional kernels and the underlying recurrence relation and can be found in ([2, 7, 4, 13, 15]). Moreover, a body of work on other constructions of GP analogs for multilayer networks can be found in [9, 3, 8, 5, 1, 17].

2 Formulation

The correspondence between single-hidden layer neural networks and GPs can be found in [11, 12, 16]. We formulate the multi-layer correspondence, which proceeds by induction on the previous layer, starting with the base case discussed in [11]. Let an L-layer deep fully-connected network of width N and with pointwise nonlinearity ϕ have pre- and post-activations denoted z_i^l and x_i^l, respectively, for the ith component in the lth layer. The input is simply denoted by $x^0 \equiv x$. Weight and bias parameters are random and independent and taken to have zero mean with variances σ_w^2/N and σ_b^2. A GP with mean μ and covariance K is denoted $\mathcal{GP}(\mu, K)$. In layer l, the network computes

$$z_i^l(x) = b_i^l + \sum_{j=1}^{N} W_{ij}^l x_j^l(x), \quad x_j^l(x) = \phi(z_j^{l-1}(x)), \tag{1}$$

where we have emphasized the dependence on the input. By the induction hypothesis, z_j^{l-1} is a GP, identical and independent for every j. $z_i^l(x)$ is hence a sum of i.i.d. random terms so that, as $N \to \infty$, due to the multivariate Central Limit Theorem, any collection evaluated for a finite set of inputs will have joint multivariate Gaussian distribution, $z_i^l \sim \mathcal{GP}(0, K^l)$. The joint distribution of preactivations across index i will also be a multivariate Gaussian, and since the preactivation for i, i' with $i \neq i'$ has vanishing covariance, this implies independence, as noted in the single-layer argument due to [11].

The covariance K^l is

$$K^l(x, x') \equiv \mathbb{E}\left[z_i^l(x) z_i^l(x')\right] = \sigma_b^2 + \sigma_w^2 \, \mathbb{E}_{z_i^{l-1} \sim \mathcal{GP}(0, K^{l-1})}\left[\phi(z_i^{l-1}(x))\phi(z_i^{l-1}(x'))\right]. \tag{2}$$

The expectation requires only the joint distribution of $z_i^{l-1}(x)$ and $z_i^{l-1}(x')$, which is a zero mean, 2D Gaussian with covariance matrix built out of $K^{l-1}(x, x')$, $K^{l-1}(x, x)$, and $K^{l-1}(x', x')$. Hence, these are the only quantities that appear in the result. We summarize this as

$$K^l(x, x') = \sigma_b^2 + \sigma_w^2 \, F_\phi\Big(K^{l-1}(x, x'),\, K^{l-1}(x, x),\, K^{l-1}(x', x')\Big) \tag{3}$$

to emphasize the recursive relationship between K^l and K^{l-1} via a deterministic function F whose form depends only on the nonlinearity ϕ. This gives an iterative series of computations which can be performed to obtain K^L for the GP describing the network's final output. The function F_ϕ is known in closed form for certain nonlinearities; for instance, for ReLU it is given in [2].

We develop a computational recipe to compute the covariance matrix for the NNGP corresponding to any well-behaved nonlinearity ϕ. All steps can be implemented using accelerated tensor operations, and computation of K^L is typically faster than solving the system of linear equations in Equation (4).

The full computational pipeline is *deterministic* and *differentiable*. The shape and properties of a deep network kernel are purely determined by hyperparameters of the deep neural network. Since GPs provide exact closed form likelihoods, this kernel construction may allow principled hyperparameter selection or nonlinearity design, e.g. by gradient ascent on the log likelihood w.r.t. the hyperparameters.

Bayesian Inference: Consider making a Bayesian prediction at test point x^*, given dataset $\mathcal{D} = \{(x^1, t^1), ..., (x^n, t^n)\}$ of input-target pairs (x, t), using the GP correspondence as prior (see e.g. [14]). The result can be computed exactly and reduces to matrix algebra: namely, $z^*|\mathcal{D}, x^* \sim \mathcal{N}(\bar{\mu}, \bar{K})$ with

$$\bar{\mu} = K^L_{x^*,\mathcal{D}}(K^L_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1}\boldsymbol{t}, \qquad \bar{K} = K^L_{x^*,x^*} - K^L_{x^*,\mathcal{D}}(K^L_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1} K^{L,T}_{x^*,\mathcal{D}} \tag{4}$$

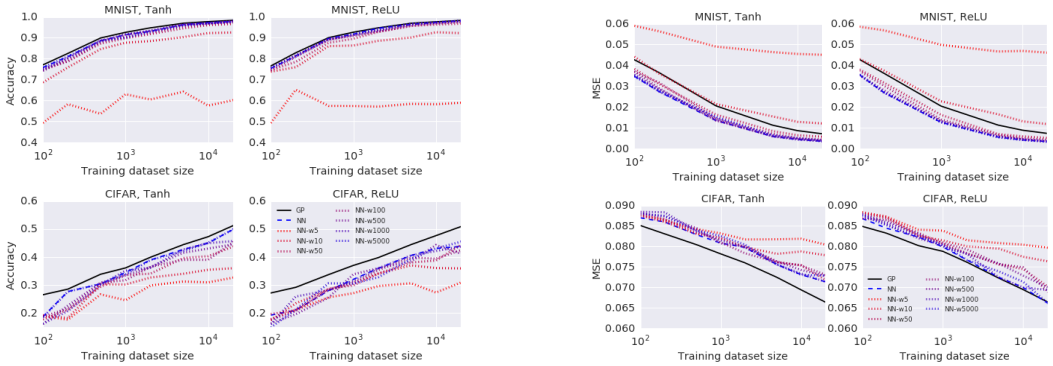
where $\mathbb{I}_n$ is the $n \times n$ identity. $K_{\mathcal{D},\mathcal{D}}, K_{x^*,\mathcal{D}}$ are $n \times n, 1 \times n$ matrices constructed by recursion of Equation 3.

3 Experimental Results

We compare NNGPs with SGD[5] trained neural networks on the permutation invariant MNIST and CIFAR-10 datasets. The baseline neural network is a fully-connected network with identical width at each hidden layer. Training is on the mean squared error (MSE) loss, chosen so as to allow direct comparison to GP predictions. We constructed the covariance kernel numerically for ReLU and Tanh nonlinearities following the formulation described in Section 2.

Performance: We find that the NNGP often outperforms trained finite width networks, and that trained neural network performance becomes more similar to that of the NNGP with increasing width. See Table 1 and Figure 1.

Uncertainty: For conventional neural networks, capturing the uncertainty in a model's predictions is challenging [6]. In the NNGP, every test point has an explicit estimate of prediction variance associated with it (Equation 4), due to its Bayesian nature. We observe that the NNGP uncertainty estimate is highly correlated with prediction error (Figure 2).



(a) Accuracy

(b) Mean squared error

Figure 1: The NNGP often outperforms finite width networks, and neural network performance more closely resembles NNGP performance with increasing width. Accuracy and mean squared error on MNIST and CIFAR-10 dataset are shown for the best performing NNGP and best performing SGD trained neural networks for given width.

Table 1: The NNGP often outperforms finite width networks. Test accuracy on MNIST and CIFAR-10 datasets. The reported NNGP results correspond to the best performing depth, σ_w^2, and σ_b^2 values on the validation set. The traditional NN results correspond to the best performing depth, width and optimization hyperparameters. Best models for a given training set size are specified by (depth-width-σ_w^2-σ_b^2) for NNs and (depth–σ_w^2-σ_b^2) for GPs.

Num training	Model (ReLU)	Test accuracy	Model (tanh)	Test accuracy
MNIST:1k	NN-2-5000-3.19-0.00	0.9252	NN-2-1000-0.60-0.00	0.9254
	GP-20-1.45-0.28	**0.9279**	GP-20-1.96-0.62	0.9266
MNIST:10k	NN-2-2000-0.42-0.16	0.9771	NN-2-2000-2.41-1.84	0.9745
	GP-7-0.61-0.07	0.9765	GP-2-1.62-0.28	**0.9773**
MNIST:50k	NN-2-2000-0.60-0.44	0.9864	NN-2-5000-0.28-0.34	0.9857
	GP-1-0.10-0.48	0.9875	GP-1-1.28-0.00	**0.9879**
CIFAR:1k	NN-5-500-1.29-0.28	0.3225	NN-1-200-1.45-0.12	0.3378
	GP-7-1.28-0.00	0.3608	GP-50-2.97-0.97	**0.3702**
CIFAR:10k	NN-5-2000-1.60-1.07	0.4545	NN-1-500-1.48-1.59	0.4429
	GP-5-2.97-0.28	**0.4780**	GP-7-3.48-2.00	0.4766
CIFAR:45k	NN-3-5000-0.53-0.01	0.5313	NN-2-2000-1.05-2.08	0.5034
	GP-3-3.31-1.86	**0.5566**	GP-3-3.48-1.52	0.5558

[5]For all presented results, the variant of SGD used is Adam. Although not shown, we found vanilla SGD produced qualitatively similar results, with slightly higher MSE.

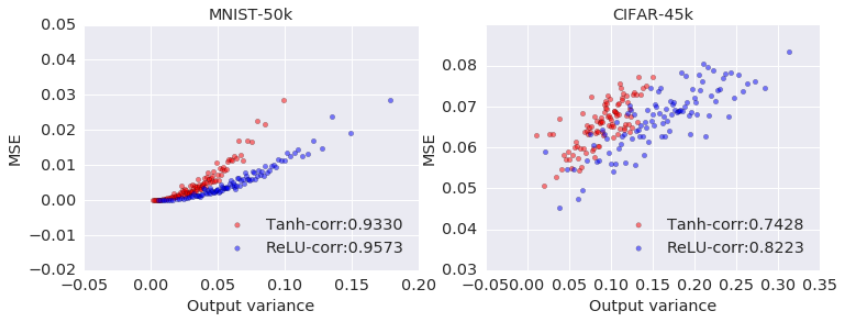


Figure 2: The Bayesian nature of NNGP allows it to assign a prediction uncertainty to each test point. This prediction uncertainty is highly correlated with the empirical error on test points. The $x-$axis shows the predicted MSE for test points, while the $y-$axis shows the realized MSE. To allow comparison of *mean* squared error, each plotted point is an average over 100 test points, binned by predicted MSE. The hyperparameters for the NNGP are depth$= 3$, $\sigma_w^2 = 2.0$, and $\sigma_b^2 = 0.2$.

References

[1] Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning*, pp. 1472–1481, 2016.

[2] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pp. 342–350, 2009.

[3] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[4] 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.

[5] David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In *Artificial Intelligence and Statistics*, pp. 202–210, 2014.

[6] Yarin Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

[7] Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. *arXiv preprint arXiv:1508.05133*, 2015.

[8] James Hensman and Neil D Lawrence. Nested variational compression in deep gaussian processes. *arXiv preprint arXiv:1412.1370*, 2014.

[9] Neil D Lawrence and Andrew J Moore. Hierarchical gaussian process latent variable models. In *Proceedings of the 24th international conference on Machine learning*, pp. 481–488. ACM, 2007.

[10] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[11] Radford M. Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

[12] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, Dept. of Computer Science, 1994.

[13] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances In Neural Information Processing Systems*, pp. 3360–3368, 2016.

[14] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[15] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *ICLR*, 2017.

[16] Christopher KI Williams. Computing with infinite networks. In *Advances in neural information processing systems*, pp. 295–301, 1997.

[17] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378, 2016.