Gaussian Predictions from Gradient Descent Training of Wide Neural Networks

Jaehoon Lee*, **Lechao Xiao***, **Jascha Sohl-Dickstein, Jeffrey Pennington**
Google Brain
{jaehlee, xlc, jaschasd, jpennin}@google.com

1 Introduction

Deep neural networks define complex systems for which many types of theoretical analyses are intractable. As in the physical sciences, investigating the extreme limits of such systems can often shed light on these hard problems. For neural networks, one such limit is that of infinite width, in which case the output of the network at initialization or after fully Bayesian training may be understood as a draw from a Gaussian process (GP) [1, 2, 3, 4, 5].

The correspondence between wide neural networks and GPs is useful because it allows the properties of neural networks to be fully specified through a mean and covariance function over inputs, and because GP inference is equivalent to exact Bayesian training of deep neural networks. In practice however, neural networks are usually trained with gradient-based methods. A natural question, therefore, is how faithfully training by gradient descent mirrors Bayesian inference. Some preliminary observations in [2] indicate that wide neural networks perform similarly to the corresponding GPs as the width becomes large, while [4] found the opposite in the case of convolutional networks.

In this note, we explore the inductive bias introduced by gradient descent training in the limit of infinitely wide networks, building upon the work of [6]. We first show that when only the readout layer is optimized, the outputs of the network correspond to a GP throughout training. Next, we show that when all layers are optimized simultaneously, the deep neural network can be replaced by a first order Taylor expansion with respect to its parameters at initialization, and the dynamics of gradient descent become *analytically tractable*.

More concretely, assuming sufficiently small learning rates, the full time evolution of gradient flow can be exactly analyzed. Taking the view of sample-then-optimize posterior sampling [7], the predictive distribution of ensembles of trained networks becomes tractable. We show that this predictive distribution is Gaussian with a specified mean and covariance during the whole training process. In general, this predictive distribution is **different** from the GP posterior, while the GP posterior corresponds to the exact Bayesian posterior.

The variance in the predictive distribution at the end of training can be interpreted as the output uncertainty of gradient descent trained neural networks. We remark that uncertainty estimates for neural network predictions are frequently obtained by ensembling many network predictions trained with different instantiations. Our results provide a theoretical probabilistic basis for such heuristics.

2 Theoretical Background

Notation

Let $\mathcal{X} \subseteq \mathbb{R}^{n_0}$ and $\mathcal{Y} \subseteq \mathbb{R}$ denote the training inputs and labels. We also identify $\mathcal{X}$ as a $|\mathcal{X}| \times n_0$ matrix and $\mathcal{Y}$ as a $|\mathcal{Y}| \times 1$ vector. Consider a fully-connected feed-forward network with L *hidden* layers with width n_l, for $l = 1, ..., L$ and a readout layer with $n_{L+1} = 1$. For each $x \in \mathbb{R}^{n_0}$, we use $h^l(x), x^l(x) \in \mathbb{R}^{n_l}$ to represent the pre- and post-activation functions at layer l with input x. The

*Both authors contributed equally to this work. Work done as a member of the Google AI Residency program (https://g.co/airesidency).

Preprint. Work in progress.

recurrence relation is defined as

$$\begin{cases} h^{l+1} &= x^l W^{l+1} + b^{l+1} \\ x^{l+1} &= \phi(h^{l+1}) \end{cases} \quad \text{and} \quad \begin{cases} W^l_{i,j} &= \frac{\sigma_\omega}{\sqrt{n_l}} \omega^l_{ij}, \quad \omega^l_{ij} \sim \mathcal{N}(0,1) \\ b^l_j &= \sigma_b \beta^l_j, \quad \beta^l_j \sim \mathcal{N}(0,1) \end{cases} \tag{1}$$

where ϕ is a point-wise activation function, $W^{l+1} \in \mathbb{R}^{n_l \times n_{l+1}}$ and $b^{l+1} \in \mathbb{R}^{n_{l+1}}$ are the weight and bias with σ_w^2 and σ_b^2 being the weight and bias variances at initialization. Note that ω^l_{ij} and β^l_j (rather than $W^l_{i,j}$ and b^l_j) are the training parameters of the network. Here we have adopted a non-standard parameterization (with a self-normaling factor $\frac{1}{\sqrt{n_l}}$ in front of the weights), referred to as the NTK-parameterization [6]. Unlike the standard parameterization, which only normalizes the forward dynamics of the network, the NTK-parameterization also normalizes its backward dynamics. Not shown, analogous results to those below hold for the standard parameterization, with more complex coefficients on several terms.

We set $\theta^l \equiv [w^l_{i,j}, b^l_j]_{i,j}$ the collection of parameters mapping *to* the l-th layers, whose cardinality is $(n_{l-1}+1) \times n_l$. Define $\theta = \cup_{l=1}^{L+1} \theta^l$ and similarly $\theta^{\leq l}$ or $\theta^{>l}$. We will also use θ_t to indicate the time dependence and in particular, θ_0 denote the initial value of the parameters. Lastly, we use $f_\theta(x) \equiv h^{L+1}(x)$ to denote the output of the neural network.

2.1 Neural Network as a Gaussian Process (NNGP)

As the width of the hidden layers approaches infinity, the Central Limit Theorem (CLT) implies that the outputs at initialization $\{f_{\theta_0}(x)\}_{x \in \mathcal{X}}$ converge weakly to a multivariate Gaussian distribution. Informally, this can be proved by induction. Conditioning on activations at layer l, each pre-activation $(h^{l+1}_j(x))$ at layer $l+1$ is a sum of n^l i.i.d. properly normalized random variables (the weights) and a Gaussian distribution (the bias term). One can apply the CLT to conclude that $\{h^{l+1}_j(x)\}_j$ are i.i.d. Gaussian; see [8, 2, 9] for more details and [7, 4] for a formal treatment.

This connection induces a correspondence between randomly initialized neural networks and Gaussian process, facilitating a fully Bayesian treatment of neural networks. More precisely, let $\mathcal{K}^l$ denote the sample-to-sample kernel function (of the pre-activation) at layer l in the infinite width setting,

$$\mathcal{K}^l(x, x') = \lim_{n_l \to \infty} \cdots \lim_{n_1 \to \infty} \frac{1}{n^l} \mathbb{E}[h^l(x) \cdot h^l(x')], \quad x, x' \in \mathbb{R}^{n_0} \tag{2}$$

then $f_{\theta_0}(\mathcal{X}) \sim \mathcal{G}P(0, \mathcal{K}^{L+1}(\mathcal{X}, \mathcal{X}))$, where the covariance kernel $\mathcal{K} \equiv \mathcal{K}^{L+1}$ can be computed recursively [2] (see Section A.1). For an unobserved test input x^*, the joint output distribution $f([x^*, \mathcal{X}])$ is also a Gaussian process. Conditioning on the training samples $f(\mathcal{X}) = \mathcal{Y}$, the posterior predictive distribution of $f(x^*)$ is also Gaussian

$$f(x^*)\big|\mathcal{X}, \mathcal{Y} \sim \mathcal{N}\left(\mathcal{K}(x^*, \mathcal{X})\mathcal{K}^{-1}\mathcal{Y},\, \mathcal{K}(x^*, x^*) - \mathcal{K}(x^*, \mathcal{X})\mathcal{K}^{-1}\mathcal{K}(x^*, \mathcal{X})^T\right). \tag{3}$$

This is the test distribution resulting from exact Bayesian inference in an infinitely wide neural network.

2.2 Gradient flow dynamics for training only the readout-layer

The connection between Gaussian processes and Bayesian wide neural networks can be extended to the setting when only the readout layer parameters are being optimized. More precisely, we show that when training only the readout layer, the outputs of the network form a Gaussian process (over an ensemble of draws from the parameter prior) throughout training, where that output is an interpolation between the GP prior and GP posterior.

Note that for any $x, x' \in \mathbb{R}^{n_0}$, in the infinite width limit $\bar{x}(x) \cdot \bar{x}(x') \to \mathcal{K}(x, x')$, where for notational simplicity we assign $\bar{x}(x) = [\frac{\sigma_w x^L(x)}{\sqrt{n_L}}, \sigma_b]$. The regression problem is specified with mean-squared loss

$$\mathcal{L} = \frac{1}{2|\mathcal{X}|} \|f(\mathcal{X}) - \mathcal{Y}\|_2^2 = \frac{1}{2|\mathcal{X}|} \|\bar{x}(\mathcal{X})\theta^{L+1} - \mathcal{Y}\|_2^2, \tag{4}$$

and applying gradient flow to optimize the readout layer (and freezing all other parameters),

$$\frac{d\theta^{L+1}}{dt} = -\frac{\eta}{|\mathcal{X}|}\bar{x}(\mathcal{X})^T\left(\bar{x}(\mathcal{X})\theta^{L+1} - \mathcal{Y}\right), \tag{5}$$

where η is the learning rate. The solution to this ODE gives the evolution of the output of an arbitrary x^*. So long as the empirical kernel $\bar{x}(\mathcal{X})\bar{x}(\mathcal{X})^T$ is invertible, it is

$$f(x^*) = \bar{x}(x^*)\bar{x}(\mathcal{X})^T(\bar{x}(\mathcal{X})\bar{x}(\mathcal{X})^T)^{-1}\left(\mathcal{Y} + \exp\left(-\frac{\eta}{|\mathcal{X}|}\bar{x}(\mathcal{X})\bar{x}(\mathcal{X})^T t\right)(\bar{x}(\mathcal{X})\theta_0^{L+1} - \mathcal{Y})\right) \tag{6}$$

For any $x, x' \in \mathbb{R}^{n_0}$, letting $n_l \to \infty$ for $l = 1, \ldots, L$, one has the convergence in distribution [4]

$$\bar{x}(x)\bar{x}(x') \to \mathcal{K}(x, x') \quad \text{and} \quad \bar{x}(\mathcal{X})\theta_0^{L+1} \to \mathcal{N}(0, \mathcal{K}(\mathcal{X}, \mathcal{X})). \tag{7}$$

Moreover θ_0^{L+1} is the only stochastic terms over the ensemble of network initializations, therefore the output $f(x^*)$ throughout training is also Gaussian distributed, with

$$\mathbb{E}[f(x^*)](t) = \mathcal{K}(x^*, \mathcal{X})\mathcal{K}^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\mathcal{K}t})\mathcal{Y}, \tag{8}$$

$$\mathrm{Var}[f(x^*)](t) = \mathcal{K}(x^*, x^*) - \mathcal{K}(x^*, \mathcal{X})\mathcal{K}^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\mathcal{K}t})\mathcal{K}(x^*, \mathcal{X})^T. \tag{9}$$

Thus the output of the neural network is also a GP and the optimal asymptotic solution (i.e. $t \to \infty$) is identical to the posterior of the NNGP (Equation 3). Therefore, in the infinite width case, the optimized neural network is performing posterior sampling if only the readout layer is being trained.

2.3 Linearization and neural tangent kernel

Now we consider that case that all layers of a wide neural network are trained. Building upon the work of [6], we show that the outputs are also Gaussian distributed throughout training and are tractable in the infinite width setting. To proceed, for each l and $x \in \mathbb{R}^{n_0}$ let $J_x^l \equiv \frac{\partial h^l(x)}{\partial \theta^{\leq l}}$, which is a $n_l \times |\theta^{\leq l}|$ vector. We denote $J = J^{L+1}$ and $J_{\mathcal{X}}^l \equiv [J_x^l]_{x\in\mathcal{X}}$ which is a $|n_l||\mathcal{X}| \times |\theta^{\leq l}|$ matrix. For each l, taking $n_j \to \infty$ for $j = 1, \ldots, l-1$ *sequentially*, the neural tangent kernel [6] converges to a deterministic quantity $J_{\mathcal{X}}^l(J_{\mathcal{X}}^l)^T \to \Theta^l \otimes \mathbf{Id}_{n_l}$, where Θ^l can be computed recursively (see Appendix A.2). For notational simplicity, we use $\Theta \equiv \Theta^{L+1}(\mathcal{X}, \mathcal{X})$ and $\hat{\Theta}$ to denote the empirical (i.e. finite width) version of Θ, which is a random variable that converges to Θ in probability as $n_1, \ldots, n_L \to \infty$. We further use $\hat{\Theta}_t$ to emphasize the dependence on time during training. The evolution of the output function f during gradient flow can be written in terms of $\hat{\Theta}_t$

$$\left.\frac{df_\theta}{dt}\right|_{\mathcal{X}} = \left.\frac{\partial f_\theta}{\partial \theta}\right|_{\mathcal{X}}\frac{d\theta}{dt} = -\frac{\eta}{|\mathcal{X}|}J_{\mathcal{X}}J_{\mathcal{X}}^T\left(f_\theta(\mathcal{X}) - \mathcal{Y}\right) = -\frac{\eta}{|\mathcal{X}|}\hat{\Theta}_t\left(f_\theta(\mathcal{X}) - \mathcal{Y}\right). \tag{10}$$

This ODE is complicated in general as $\hat{\Theta}_t$ evolves with t. Remarkably, [6] showed that

$$\sup_{t\in[0,T]} \|\hat{\Theta}_t - \hat{\Theta}_0\|_F, \quad \|\hat{\Theta}_0 - \Theta\|_F \to 0, \quad \text{as} \quad n_1, \ldots, n_L \to \infty \quad \text{sequentially} \tag{11}$$

under a technical assumption $\int_0^T \|f_\theta(\mathcal{X}) - \mathcal{Y}\|_2\, dt < \infty$. This indicates that, in the large width regime, the dynamics of the linearization of the neural network are a good approximation to those of the original network. Define the linearized neural network to be

$$f_\omega^{\text{lin}}(x) \equiv f_{\theta_0}(x) + J_x\omega_t, \quad x \in \mathbb{R}^{n_0} \tag{12}$$

where $\omega_t \equiv \theta_t - \theta_0$ is the total change in the parameters since initialization. We will sometimes drop the subscript t to simplify notation.

Note that this is just the first order approximation of the neural network at initialization. Therefore we observe that deep neural networks are well approximated by their linearization in the limit of infinite width. The output-parameter Jacobian J_x (and $J_{\mathcal{X}}$) is determined when the network is initialized and remains unchanged throughout training. We make an assumption that $\hat{\Theta}_0$ and Θ are invertible.

Associating the linearized network with the same loss $\mathcal{L}^{\text{lin}} = \frac{1}{2|\mathcal{X}|}\|f_{\theta_0}(\mathcal{X}) + J_{\mathcal{X}}\omega_t - \mathcal{Y}\|_2^2$, the gradient flow of the parameters of the linearized network are

$$\frac{d\omega}{dt} = -\eta\nabla_\omega\mathcal{L}^{\text{lin}} = -\frac{\eta}{|\mathcal{X}|}J_{\mathcal{X}}^T(f_\omega^{\text{lin}}(\mathcal{X}) - \mathcal{Y}) \tag{13}$$

$$\left.\frac{df_\omega^{\text{lin}}}{dt}\right|_{\mathcal{X}} = -\frac{\eta}{|\mathcal{X}|}J_{\mathcal{X}}J_{\mathcal{X}}^T(f_\omega^{\text{lin}}(\mathcal{X}) - \mathcal{Y}) = -\frac{\eta}{|\mathcal{X}|}\hat{\Theta}_0(f_\omega^{\text{lin}}(\mathcal{X}) - \mathcal{Y}) \tag{14}$$

respectively. These dynamics can be solved analytically,

$$\omega = -J_{\mathcal{X}}^T\hat{\Theta}_0^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\hat{\Theta}_0 t})(f_0^{\text{lin}}(\mathcal{X}) - \mathcal{Y})\,, \tag{15}$$

$$f_\omega^{\text{lin}}(\mathcal{X}) = (I - e^{-\frac{\eta}{|\mathcal{X}|}\hat{\Theta}_0 t})\mathcal{Y} + e^{-\frac{\eta}{|\mathcal{X}|}\hat{\Theta}_0 t}f_0^{\text{lin}}(\mathcal{X})\,. \tag{16}$$

The output of an arbitrary point x^* is

$$f_\omega^{\text{lin}}(x^*) = f_{\theta_0}(x^*) + J_{x^*}\omega = f_{\theta_0}(x^*) - \hat{\Theta}_0(x^*, \mathcal{X})\hat{\Theta}_0^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\hat{\Theta}_0 t})(f_0^{\text{lin}}(\mathcal{X}) - \mathcal{Y}). \tag{17}$$

In the infinite width limit, $\hat{\Theta}_0$ approaches a deterministic quantity Θ, and $f_{\theta_0}(x^*)$ and $f_0^{\text{lin}}(\mathcal{X}) = f_{\theta_0}(\mathcal{X})$ are thus the only stochastic terms over the ensemble of network initializations. $f_{\theta_0}(\mathcal{X})$ is a Gaussian with mean zero and covariance $\mathcal{K}$. Therefore the output $f_\omega^{\text{lin}}(x^*)$ throughout training is also Gaussian distributed, with

$$\mathbb{E}[f_\omega^{\text{lin}}(x^*)](t) = \Theta(x^*, \mathcal{X})\Theta^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\Theta t})\mathcal{Y}\,, \tag{18}$$

$$\begin{aligned}\text{Var}[f_\omega^{\text{lin}}(x^*)](t) = {}& \mathcal{K}(x^*, x^*) - 2\Theta(x^*, \mathcal{X})\Theta^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\Theta t})\mathcal{K}(x^*, \mathcal{X})^T \\ & + \Theta(x^*, \mathcal{X})\Theta^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\Theta t})\mathcal{K}\Theta^{-1}(I - e^{-\frac{\eta}{|\mathcal{X}|}\Theta t})\Theta(x^*, \mathcal{X})^T\,.\end{aligned} \tag{19}$$

Unlike the case when only the readout layer is being optimized, when all layers are being trained the neural network is no longer performing posterior sampling in general (one possible exception is when the NNGP kernel and NTK are the same up to a scalar multiplication.)

3 Experiments

In this section we illustrate that the exact dynamics of gradient descent training described in the previous section approximate those of sufficiently wide networks well. We focus on training fully connected neural networks with constant width for every hidden layer with full batch gradient descent. We consider binary classification task on two classes (horses and planes) of the CIFAR-10 dataset. We treat the binary classification task as a regression task with one class regressing to $+1$ and the other to -1. Learning rates are chosen sufficiently small that the gradient flow continuum limit holds well.

Predictive output distribution

In Figure 1, the predictive output distribution of input points interpolating between two training points are shown for an ensemble of neural networks and their corresponding GPs. The interpolation is given by $x(\alpha) = \alpha x^{(1)} + (1-\alpha)x^{(2)}$ where $x^{(1,2)}$ are two training inputs with different classes. The trained network has 3 hidden layers of width 8,192, a $\tanh$ activation function, $\sigma_w^2 = 1.5$, no bias, and $\bar{\eta} = 0.5$. We observe that the mean and variance dynamics of neural network outputs during gradient descent training follow the analytic dynamics from linearization well (Equation 18, Equation 19). Also the NNGP posterior which corresponds to exact Bayesian inference is similar but noticeably different from predictive distribution at the end of gradient descent training (t=65,536).

Analytic dynamics vs Simulation

For a particular initialized networks, one can analytically predict their dynamics of weights and outputs. In Figure 2, we compare dynamics predicted for a linearized network (Equation 15, Equation 16) and from training the actual network. We see remarkably good agreement on both output values corresponding to training and test inputs. The weight change dynamics is in good agreement as well, though not as close as for the outputs.

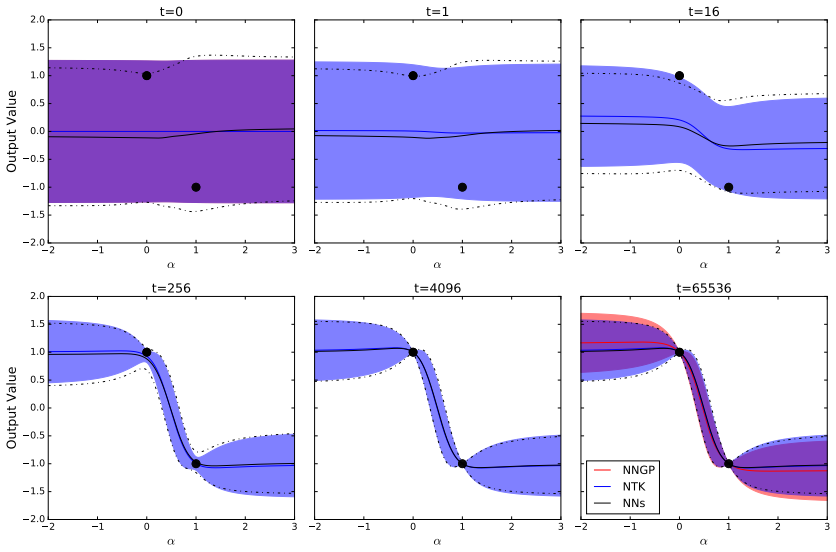
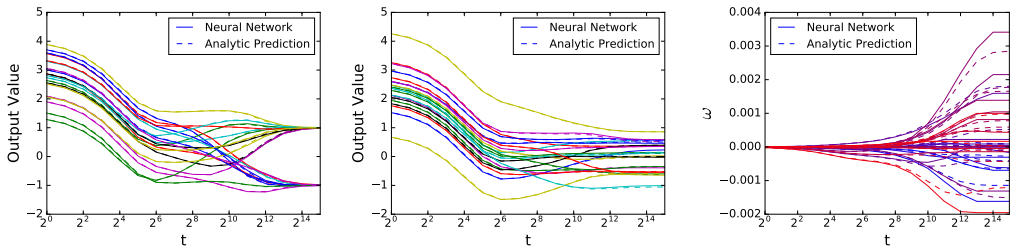


Figure 1: **Dynamics of mean and variance of trained neural network outputs follows analytic dynamics from linearization**. Black lines indicate the time evolution of the predictive output distribution from an ensemble of 100 trained neural networks (NNs). The blue region indicates the analytic prediction of the output distribuion throughout training (Equation 18, Equation 19). Finally, the red region indicates the prediction that would result from training only the top layer (Equation 8, Equation 9). The trained network has 3 hidden layers of width 8,192, $\tanh$ activation functions, $\sigma_w^2 = 1.5$, no bias, and $\bar{\eta} = 0.5$. The output is computed for inputs interpolated between two training points (denoted with black dots) $x(\alpha) = \alpha x^{(1)} + (1-\alpha)x^{(2)}$. The shaded region and dotted lines denote 2 standard deviations ($\sim 95\%$ quantile) from the mean denoted in solid lines. Training was performed with full-batch gradient descent with dataset size $m = 128$.



(a) Train set output dynamics (b) Test set output dynamics (c) Weight difference dynamics

Figure 2: **Exact and experimental dynamics are nearly identical for network outputs, and are similar for individual weights.** Experiment is for a ReLU network with 3 hidden layers of width n =4,096, $\bar{\eta} = 0.01$ $m = 256$, $\sigma_w^2 = 2.0$, and $\sigma_b^2 = 0.1$. All three panes show dynamics for a randomly selected subset of datapoints or parameters.

4 Conclusion

In this work, we have derived the distribution over test point predictions and exact weight dynamics which occur during gradient descent training of infinitely wide neural networks. If only the readout layer is trained, the resulting distribution from gradient descent training is identical to that which would result from fully Bayesian neural network training, and is described by a Gaussian process. If all the layers in the network are trained, the distribution over test point predictions corresponds to a different Gaussian process, but is no longer equivalent to fully Bayesian training of the neural network. We finally verified the predictions for output distribution and weight dynamics experimentally.

References

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

[2] Jaehoon Lee, Yasaman Bahri, Roman Novak, Sam Schoenholz, Jeffrey Pennington, and Jascha Sohl-dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[3] Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[4] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian convolutional neural networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018.

[5] Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow gaussian processes. *arXiv preprint arXiv:1808.05587*, 2018.

[6] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[7] Alexander G. de. G Matthews, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for bayesian linear models. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

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

[9] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5393–5402, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

A Iterative formula for Kernels

A.1 NNGP Kernel

The NNGP kernel can be computed recursively:

$$\mathcal{K}^{l+1}(x,x') = \sigma_w^2 \mathcal{T}(\mathcal{K}^l(x,x')) + \sigma_b^2, \quad \mathcal{K}^1(x,x') = \frac{\sigma_w^2}{n_0} x x'^T + \sigma_b^2, \quad x, x' \in \mathbb{R}^{n_0}\,, \tag{20}$$

where $\mathcal{T}$ is first defined as a function on the set of 2×2 positive semi-definite matrices (denoted by PSD_2) as follows

$$\mathcal{T}(\Sigma) = \mathbb{E}[\phi(u)\phi(v)], \quad (u,v) \sim \mathcal{N}(0,\Sigma), \quad \Sigma \in \mathrm{PSD}_2 \tag{21}$$

and then extended to a mapping from the set of $m \times m$ positive semi-definite matrices PSD_m to itself in a straightforward fashion for all $m \geq 1$.

A.2 NTK

Defined recursively

$$\Theta^l(x,x') = \sigma_w^2 \dot{\mathcal{T}}(\mathcal{K}^{l-1}(x,x'))\Theta^{l-1}(x,x') + \mathcal{K}^l(x,x')\,, \quad x, x' \in \mathbb{R}^{n_0} \tag{22}$$

with $\Theta^0 = 0$ as a base case. Here $\dot{\mathcal{T}}$ is defined exactly the same as in Equation 21 except ϕ replaced by its derivative $\dot{\phi}$.