Deep Neural Networks as Gaussian Processes

Mathieu Chevalley[*]
Department of Computer Science
ETH Zurich
`mchevalley@student.ethz.ch`

Daniel Wälchli[*]
D - MAVT
ETH Zurich
`wadaniel@ethz.ch`

Abstract

During recent years there have been a lot of breakthroughs in the theoretical understanding of Neural Networks. One of the papers that paved the path to a range of discoveries is *Deep Neural Networks as Gaussian Processes* by Lee et al. [2017]. In this paper, the authors show that *deep* infinitely wide Neural Networks are equivalent to Gaussian Processes. In this report, we conduct a thorough review of this paper. We first review the background literature that leads to this new characterization of Neural Networks as well as the literature and discoveries that followed. Thereafter we present their main proof and the limitations of this result and the conclusions that Lee et al. [2017] draw from their experiments. We finally conduct four experiments to gain further insights following the shortcoming of the paper.

1 Introduction

In this report, we conduct a critical review of the paper *Deep Neural Networks as Gaussian Processes* by Lee et al. [2017]. Also, we review the associated literature and present the main proof of the paper. Lastly, we explore some potential limitations of results presented in the paper and we conduct some follow-up experiments to verify our claims.

Paper Main contributions The central contributions of the authors of *Deep Neural Networks as Gaussian Processes* [Lee et al., 2017] is the proof of the exact equivalence between the prior of infinitely wide *deep* Neural Network (NN) and Gaussian Process (GP). This result extends the seminal work of Neal [1994], who proved this widely known result for single-hidden-layer neural networks. This equivalence means that the prior function computed by a NN after initialization of the weights is drawn from a Neural Network Gaussian Process (NNGP). The authors additionally show the analytical form of the covariance matrix, respectively the kernel of the corresponding GP, as well as how to *efficiently* compute it.

This correspondence allows for the exact Bayesian inference for infinitely wide neural networks through the evaluation of the corresponding NNGP. Going beyond the context of Bayesian neural networks, Lee et al. [2017] conduct multiple experiments comparing NNGPs and finite width gradient-based trained neural networks on MNIST and CIFAR-10. From these comparisons, they aim to shed some light on commonly observed behaviors of NNs that are still not theoretically explained. They also observe potential new insights for NNs in practice.

Their main experimental results are the following: (i) NNGPs always outperform gradient-based trained NNs when the hyperparameters are tuned using the validation set (ii) finite width NNs performance approaches that of NNGPs as the layer width grows, suggesting that the NN converges to the NNGP and that Stochastic Gradient Descent (SGD) may implement approximate Bayesian training (iii) the GP uncertainty estimates are correlated to the prediction error (iv) the performance

[*]Joint work for the research project of the course *Guarantees for Machine Learning* at ETH Zurich

of the NNGP as a function of the variance of weight and bias follows the phase diagram predicted by the work on deep signal propagation of Samuel S Schoenholz and D. [2017].

Our contributions Our work includes three main contributions. In section 3 we present our literature review where we start by presenting related literature that preceded the paper Lee et al. [2017] and we give a broad overview of the context in which it was published. We also present a concurrent work by Matthews et al. [2018] that present the same result, which allows us to compare the approaches both papers used. We then review the literature that came after the paper. Notably, we review Neural Tangent Kernels (NTKs), which is a new kernel that can be used to study NNs. The work on NTKs gives new insights on NNs and thus allow us to better assess the relevance and importance of the result in Lee et al. [2017].

Second, we outline our considerations regarding the theoretical and practical limitations and shortcomings of given paper in section 5. We express some doubts on the conclusions drawn from their experiments which is also supported by more recent results presented in subsection 3.3.

Finally, we present our experimental results in which we unveil previously addressed limitations and where we explore new potential insights on NNs using NNGPs.

2 Motivation

Neural Networks have become the *de-facto* models for many machine learning tasks, and most recent breakthroughs come from training *huge* networks, parametrized by billions of variables [Brown et al., 2020]. Indeed, Deep Neural Networks (DNNs) are known to be easy to optimize [Choromanska et al., 2015, Goodfellow et al., 2014], highly expressive [Montufar et al., 2014] while keeping good generalization properties [Hardt et al., 2015]. However, most of these properties have been obtained through the inventions of new designs and architecture, as well as fine-grained hyper-parameter tuning. Many of these ideas came from intuition rather than theory. This motivates the search for more theoretical understanding in order to guide design and training decisions.

By proving a correspondence between DNNs and GPs, we are now able to perform Bayesian inference with DNNs, and we may also be able to obtain new insights on their properties and behaviors.

2.1 Bayesian inference

One of the main advantages of Bayesian inference is that all predictions come with an uncertainty estimate. Indeed, we can treat the variance of the posterior distribution as a measure of how certain the model is about this prediction. This uncertainty estimate is usually well correlated with prediction error. In critical tasks, such as medical applications and autonomous driving, the capability to assess the quality of a model's prediction can be crucial for practical use. Highly uncertain prediction can be handed to the medical doctor or the pilot, to be manually treated in an *active learning* fashion, which would also gradually improve the model performance.

2.2 Guarantees on Neural Networks

Some examples of theoretical guarantees of great interest are: (i) why large and deep networks seem to perform better (ii) how the weights should be initialized and what kind of inductive bias it introduces on the learned function (iii) what functions they are able to model depending on their architecture (iv) why SGD works and how to perform optimization (v) whether we can obtain generalization guarantees.

3 Literature review

Even though the only actual work that precedes this paper is the work from Neal [1994], Lee et al. [2017] relies on different results in the domain of Neural Networks and Gaussian processes. We also review a concurrent paper by Matthews et al. [2018] that proves the same result and thus has some overlap. We also present some later works that follow Lee et al. [2017].

3.1 Before the paper

Early days As already mentioned, we can trace the related literature back to 1994, when Neal showed that infinite width single-hidden-layer NNs converge to a GP. The proof of Neal [1994] is only an existence proof. In 1997, Williams uncovers the analytical form of the covariance matrix, as well as how to perform Bayesian inference via simple matrix computations. The main idea is to replace the prior of the weights and biases with a prior over functions drawn from the corresponding GP. These results were believed to also hold for *deep* NNs, but a formal proof only materialized in 2018 [Lee et al., 2017, Matthews et al., 2018] and will be presented later in section 4.

Compositional Kernels In the kernel literature, Cho and Saul [2009] derives a compositional kernel aiming at *mimicking* the multi-layer structure of NNs. These kernels can be used in GPs [Krauth et al., 2016], but they do not produce GPs that are exactly equivalent to NNs.

Deep GPs Another related line of work are the attempts at stacking GPs, such as deep GPs. Here the idea is to try to match the expressiveness of DNNs for GPs by adopting the same layered architecture. This architecture may allow GPs to model more complex probability distributions. Deep Gaussian Process (DGP) were first introduced by Lawrence [2007], and then extended by Damianou and Lawrence [2013]. Duvenaud et al. [2014] perform an analysis that applies to DGPs, studying their behavior as the depth grows, exhibiting some pathologies, and how to correct them. In their paper, Duvenaud et al. also uses DGPs as a proxy to studying and improving DNNs.

Deep signal propagation The work of Lee et al. has similarities, particularly in the way of deriving the proof, to the study of NNs based on mean-field theory [Samuel S Schoenholz and D., 2017, Poole et al., 2016]. By studying whether and how information from the input is able to travel through an untrained network, they are able to gain insights on how the selection of the initial hyper-parameters may restrict the learning and expressiveness of a network. Samuel S Schoenholz and D. shows the existence of so called "depth scales", which bounds the maximum depth such that information can travel through the NN. Furthermore, they characterize ordered-chaotic *phase diagrams* for the weight-bias pair of hyper-parameters, arguing that a network may only be trained close to criticality. Another implication is that only certain pairs of weight-bias allow the network to be infinitely deep while remaining trainable.

SGD as approximate Bayesian training Recent work by Mandt et al. [2017] has shown that SGD can perform approximate Bayesian training under certain conditions. They demonstrate how SGD with a constant learning rate can be approximated by a multivariate Ornstein-Uhlenbeck process. This Bayesian view of SGD allows it to be used as a new variational Expectation Maximization (EM) algorithm. However, more investigation needs to be done in order to determine whether SGD, as used in practice, indeed implements approximate Bayesian inference.

3.2 Concurrent work

Published in the same conference (ICLR 2018) than Lee et al. [2017], the paper of Matthews et al. also proves the GP equivalence of DNNs. Obviously, there is some overlap between the two works, but they differ in the derivation of the proof as well as in their experimental studies. In Matthews et al. [2018], the authors prove the convergences by growing all the layer simultaneously to infinity, whereas Lee et al. take the layers to infinity one by one. Matthews et al. [2018] additionally shows that the convergence holds in distribution, whereas Lee et al. does not prove any type of convergence. More importantly, Matthews et al. focuses on comparing NNGPs to their corresponding finite Bayesian networks, using gold standard sampling and Maximum Mean Discrepancy (MMD) methods. On the other hand, Lee et al. compare NNGPs against finite NNs trained with SGD, such that new insights on DNNs can be unveiled and to suggest that SGD may implement approximate Bayesian training.

To summarize, the authors of Matthews et al. focus on a practical comparison of what they have earlier proved to be comparable, whereas in the work of Lee et al. we find extensions of the theory to practical examples.

3.3 After the paper

The question now arises whether a correspondence to Gaussian processes exists for different types of neural networks, for example, convolutional or residual networks. A proved correspondence for Convolutional Neural Network (CNN) is in the paper *Bayesian Deep Convolutional Networks With Many Channels Are Gaussian Processes* [Novak et al., 2018], where they derive "an equivalence for multi-layer CNN both with and without pooling layers", as well as in Garriga-Alonso et al. [2018] for both residual and non-residual CNN. Yang [2019] shows that recurrent networks, but also feed-forward networks of *any* architectures (e.g with pooling, batch normalization) are Gaussian Processes.

NTK In a similar idea, Jacot et al. [2018] introduce the NTK, which is a tangent kernel that describes a DNN in function space. They show that this kernel is constant in the infinite limit and only depends on depth, non-linearity, and the variance of the parameter initialization. This kernel is a powerful method to study DNNs, as they also show that full batch gradient descent in parameter space is equivalent to kernel descent (with respect to the NTK) in function space in the infinite width limit. Arora et al. [2019] extends this work and introduce the NTK for CNN, as well as how to compute the kernel efficiently. Furthermore, they prove some non-asymptotic bounds for the convergence to the NTK at initialization and equivalence of the predictions between wide fully trained neural net and kernel regression with NTK.

Nonetheless, these kernel equivalences do not necessarily mean that the outputs of the gradient trained network follow a GP, and there are no direct relations between the NTK and the kernel of the corresponding NNGP. By studying DNNs via a linearized approximation, that is exact in the infinite limit, the authors of Lee et al. [2019] attempt to empirically verify the theory of Jacot et al. [2018]. Also they empirically find that the agreement between the predictions of the original network and of the linearized approximation is very high in the first few training steps, even if the width is finite. They empirically find very good correspondence between the empirical and analytical tangent kernels of linearized networks. They also compute the analytical form of the NTK for the ReLu and `erf` activation function. More interestingly, they show that the predictions of a full-batch gradient-trained neural network can be characterized by a GP as the width grows to infinity if the loss is the squared loss. They compare this GP with the NNGP and find significant discrepancies, which may suggest that the inductive bias of gradient descent differs from Bayesian inference. In fact, they find that "gradient descent does not generate samples from the posterior of any probabilistic model". This insight further reinforces our conviction that SGD may not implement approximate Bayesian training as suggested in Lee et al. [2017].

Lastly, as a caveat, let's note that all the works on NTK may not exactly describe the behavior of DNN in practice, as: they use a non-standard parameterization of the network calls NTK parameterization, that most results hold in the infinite or linearized case and that gradient descent is *continuous* time. As such, it should be further investigated if those theoretical results transpose to practical DNNs (i.e, LeCun parameterization, discrete-time small-batch gradient descent with possibly large and varying learning rate, finite width, and additional regularization techniques such as early stopping).

4 Main Proof

The correspondence between GP and deep, infinitely wide neural networks will be shown in the following two subsections. In section 4.1 we review the single-hidden layer case and in section 4.2 we expand the proof to the multi-layer case.

4.1 Single-Hidden Layer Network

The proof for the correspondence between single-hidden layer neural networks and GP has been first published by Neal [1994]. Here we consider single-hidden layer network with width N_1 and activation functions ϕ at the neurons. Weight and bias parameters for the l-th layer are denoted by W_{ij}^l and b_i^l, which are independently initialized from a normal distribution with zero mean and variances σ_W^2/N_l and σ_b^2, respectively. The activities in layer l are denoted by x_i^l and the neural network outputs are denoted by z_i^l. The input of the NN is given by x_i (omitting the subscript l=0).

Following this, the activities in the single-hidden layer NN are given by

$$x_j^1(x) = \phi\Big(b_j^0 + \sum_{k=1}^{d_{in}} W_{jk}^0 x_k\Big) \tag{1}$$

and the output is computed as

$$z_i^1(x) = b_i^1 + \sum_{j=1}^{N_1} W_{ij}^1 x_j^1(x). \tag{2}$$

Because the weight and bias parameters are taken to be i.i.d., the activities x_j^1, $x_j'^1$ are independent for $j \neq j'$. Moreover, since $z_i^1(x)$ is a sum of i.i.d. terms, it follows from the Central Limit Theorem (CLT) that in the limit of infinite width, i.e. $N_1 \to \infty$, $z_i^1(x)$ will be Gaussian distributed with mean zero and some variance. Likewise, from a multi-dimensional CLT, any finite collection of outputs $\{z_i^1(x^{\alpha=1}), ..., z_i^1(x^{\alpha=k})\}$ will have a joint multivariate Gaussian distribution, which is exactly the definition of a GP. Therefore we conclude that $z_i^1 \sim \mathcal{GP}(\mu^1, K^1)$, a GP with mean μ^1 and covariance K^1. As previously mentioned, the parameters are sampled from a Gaussian distribution with zero mean, and hence

$$\mu^1(x) = \mathbb{E}[z_i^1(x)] = \mathbb{E}[b_i^1] + \sum_{j=1}^{N_1} \mathbb{E}[W_{ij}]E[x_j(x)] = 0 \tag{3}$$

and

$$K^1(x, x') = \mathbb{E}[z_i^1(x), z_i^1(x')] = \sigma_b^2 + \sigma_W^2 \mathbb{E}[x_i^1(x)x_i^1(x')] = \sigma_b^2 + \sigma_W^2 C(x, x'), \tag{4}$$

using the fact that the parameter of the NN are independently distributed and hence $\mathbb{E}[b_i, W_{ij}] = \mathbb{E}[W_{kl}, W_{ij}] = 0$. The term $C(x, x')$ can be computed by integrating against the distribution of W^0 and b^0. Note that any two outputs $z_i^1(x)$ and $z_i^1(x')$ for $i \neq j$ are joint Gaussian with zero mean and zero covariance despite utilizing the same features produced by the hidden layer.

4.2 Deep Networks

In this subsection we extrapolate the results for single hidden layer networks to deep neural networks by induction, iteratively taking the layer widths to infinity $(N_1 \to \infty, N_2 \to \infty, ...)$.

Suppose that the output of layer $l-1$ z_j^{l-1} is a GP, identical and independent for every j. The activities $x_j^l(x) = \phi(z_j^{l-1})$ are therefore independent and identically distributed and the output at layer l can be computed by

$$z_j^l(x) = b_j^l + \sum_{j=1}^{N_l} W_{ij}^l x_j^l(x). \tag{5}$$

Analogously to the single layer case, $z_j^l(x)$ is a sum of i.i.d. random terms, such that, as the width of layer l grows $N_l \to \infty$, any finite collection $\{..\}$ will follow a joint multivariate Gaussian distribution and $z_j^l \sim \mathcal{GP}(0, K^l)$. The covariance is given by

$$K^l(x, x') = \mathbb{E}[z_j^l(x), z_j^l(x')] = \sigma_b^2 + \sigma_W^2 \mathbb{E}_{z_i^{l-1} \sim \mathcal{GP}(0, K^{l-1})}[\phi(z_i^{l-1}(x))\phi(z_i^{l-1}(x'))]. \tag{6}$$

The expectation in eq. (6) is over the random variable z_i^{l-1}, which is equivalent to integrating against the joint distribution of $z_i^{l-1}(x)$ and $z_i^{l-1}(x')$. By induction, the joint distribution follows a two-dimensional Gaussian with zero mean whose covariance matrix has entries $K^{l-1}(x, x)$, $K^{l-1}(x, x')$ and $K^{l-1}(x', x')$. And hence we can rewrite eq. (6)

$$K^l(x, x') = \sigma_b^2 + \sigma_W^2 F_\phi\big(K^{l-1}(x, x), K^{l-1}(x, x')K^{l-1}(x', x')\big), \tag{7}$$

in order to highlight the recursive relationship between K^l and K^{l-1} via a deterministic function F whose form depends on the activation function ϕ.

The authors of Lee et al. [2017] present a computationally efficient approach to compute $K^l(x, x')$ through a series of iterative computations, exploiting that many interim solutions can be reused and stored in a lookup table.

5 Limitations

In this section, we briefly outline the theoretical and experimental limitations of the work of Lee et al. [2017].

Proof Limitations In the proof presented in section 4, one takes the hidden layer widths to infinity in succession, but in practice, one usually fixes the depth of the NN and then simultaneously increases the width of the layers, which is in contrast to the derivation of the NNGP. Also, no proof of the type of convergence from DNN to NNGP is presented by the authors.

Also, it would be interesting to look at non-asymptotic guarantees for the convergence of NNs to NNGP. Having only convergence in the limiting case does not allow us to confidently draw conclusions of the behavior of the NNGPs in practice, as the dynamic of a NN with infinite width is very different from the finite case.

Experiments Limitations One practical limitation is based on the fact that the correspondence between NNs and NNGPs hold only for infinitely wide neural nets trained in a *fully* Bayesian manner. Therefore, all parallels drawn between finite width SGD trained neural networks do not follow the theory but are purely based on experimental evidence (see Subsection 3.1 of the original work).

First, we find that the experiments do not effectively show that the NN converges to a NNGP as the width grows. The authors draw this conclusion by comparing the accuracy and the Mean Squared Error (MSE) loss on a test set in a range of classification tasks, rather than directly comparing the outputs of a NN against the outputs of a NNGP (in our opinion, calculating the MSE between the outputs of the NN and the NNGP would be more insightful). If a large SGD trained finite NN indeed converges to the NNGP, this would mean two things: that a NN is close to the NNGP even in the non-asymptotic case, and more importantly, that SGD implements some kind of approximate Bayesian training, which is conjectured by the authors. We think that this claim is not supported strongly enough by the experiments conducted, and more recent work seems to show that the inductive bias of SGD training is different from Bayesian inference (see subsection 3.3).

Furthermore, we question that the NNGPs performance always surpasses the performance of neural networks (see Table 3 in the original work), as the authors try to experimentally show. One of our hypotheses is that their claim is related to some optimization issues of the NN when the width is too large or the network too deep, a problem that is usually solved by adding skip connections between the layers. In their experiments, the authors perform a large hyper-parameter search using the validation set for the NN and the NNGP separately for different training set sizes, and it seems to indicate that the NNGP usually has the best performance. First, this is surprising because the authors of [Novak et al., 2018] show that CNNs perform better than their GP equivalent. Secondly, the parameters that are search for the NN, the optimization algorithms used as well as how it is trained (we assume until zero training error) are not made clear by the authors. The paper also seems to imply that if NNGPs were scalable, they would be a better option, which we think is probably a bit overblown.

6 Follow-up experiments

Mean Squared Output Differences (MSOD) To address of our limitation pointed out earlier in section section 5, namely that the authors did not effectively show the convergence of a NN towards an NNGP as the width grows, we will use an additional measure (besides the accuracy and the MSE) for these three classification tasks. We will measure the MSOD of the NNGP and the NN. This will help us identifying if the NN indeed converges to an NNGP as the layer width grows. Solely based on converging values for the accuracy and the MSE, one cannot tell if they truly produce the same outputs. E.g. we would like to test if they both misclassify the same images. If this holds, one can support the conjecture that SGD may implement approximate Bayesian training, which is one key statement made in the original work.

Experiment 1 Our follow-up experiments will be based on the experimental limitations described in the previous section. To start with, we aim to reproduce the classification results on the CIFAR:10k dataset using the ReLu activation function as presented in Table 1 of the original work. Thereby we can also test our experimental setup and verify no mistakes are being made for our later studies.

Experiment 2 In a second experiment, we would like to verify the classification capabilities of the NNGP using the STL-10:1k dataset. We may uncover a trend that NNGPs perform better than NNs on more complex data (here we assume that the complexity of the data is directly linked to the size[2] of the images). Note that the outperformance of the NNGP is larger on the CIFAR dataset than on the MNIST-10 as shown in Figure 1 of the original work. For this analysis, we consider the ReLu activation functions and optimize σ_b and σ_W for network widths 5, 50, 500, and 5000 for the NN, and network depths 1, 3, 5 and 7 for the NN and NNGP.

Experiment 3 In our third experiment, we will use the CIFAR:10k dataset to investigate the convergence of the NN towards a NNGP. For this, we measure the MSOD while growing the NN width and keeping the depth and the initialization parameter σ_W and σ_b fixed. Hereby we would like to address one of the limitations pointed out earlier, that the convergence of the NN was not effectively shown by the authors.

Experiment 4 And last, we will investigate if the reported outperformance of the NNGP in the classification tasks is an artifact of the chosen optimization method for the NN. It is well known that larger networks become more difficult to optimize, and hence the choice of the optimization method used during training can lead to a large impact on the results. Therefore we will investigate if other optimization algorithms than SGD or Adam [Kingma and Ba, 2014] with fixed learning rate for the NN yield better results in the classification of the CIFAR:10k dataset. Also, we may try optimization techniques (such as dropout or early stopping) that help to regularize the DNN and prevent over-fitting, which is crucial when dealing with large networks.

Following tables give a detailed description of the setups we need to run in order to conduct experiment 1 - 4.

Experiment 1		
MEASURE	DATASET	CONFIGURATION
MSE, Accuracy	CIFAR:1k	NN-5-500-1.29-0.28
MSE, Accuracy	CIFAR:1k	GP-7-1.28-0.00
MSE, Accuracy	CIFAR:10k	NN-5-2000-1.60-1.07
MSE, Accuracy	CIFAR:10k	GP-5-2.97-0.28

Table 1: Runs for experiment 1. We only consider the ReLu activation function as it yielded the best result more often (see Table 2 of the original work).

Experiment 2		
MEASURE	DATASET	CONFIGURATIONS
MSE, Accuracy, MSOD	STL-10:10k	NN widths: 5, 50, 500, 5000 depths: 1, 3, 5, 7
MSE, Accuracy, MSOD	STL-10:10k	NNGP depths: 1, 3, 5, 7

Table 2: In experiment 2 we search for the params σ_b and σ_W that optimize the classification accuracy of the NNs and NNGPs in the STL-10 dataset under given configuration. We will also measure the MSE and the MSOD. We only consider the ReLu activation function.

Experiment 3		
MEASURE	DATASET	CONFIGURATION
MSE, Accuracy, MSOD	CIFAR:10k	NN widths: 5, 50, 500, 5000 depth 5, $\sigma_W = 1.60$, $\sigma_b = 1.07$
MSE, Accuracy, MSOD	CIFAR:10k	NNGP depth 5 $\sigma_W = 1.60$, $\sigma_b = 1.07$

Table 3: Experiment 3: convergence study for increasing width while keeping depth and variances constant. Configuration based on original work.

[2]MNIST-10 consists of 28x28 pixel images, CIFAR of 3x32x32 pixel images and STL-10 3x96x96 of pixel images

Experiment 4		
MEASURE	DATASET	CONFIGURATION
MSE, Accuracy	CIFAR:10k	NN-5-2000-1.60-1.07

Table 4: In experiment 4 we use one case from experiment 1, but using different optimization techniques for the NN to investigate if we can achieve better results. Configuration based on original work.

7 Results

All relevant codes to produce our results can be found in this GitHub repository `https://github.com/wadaniel/nngp/blob/master/nngp.py`.

7.1 Experiment 1: Reproducibility

The following tables show our reproduced results based on the configurations given in the original work. We compare our obtained accuracy on the test sets with the accuracy presented in the original work. Found differences are printed in braces. As you can see, we obtained a higher accuracy for the NNGP while using 1000 and 10000 training images.

NN For the NN, reproducing the results was difficult as the authors do not provide their source code, and do not give a highly precise description of their experimental setting. Similar to the paper, we use Adam with a constant learning rate, which the authors say produce results similar to SGD, as well as weight decay. This technique, which penalizes high weights, is known to reduce overfitting even when the number of training steps is large. This allows the network to be trained until convergence without overfitting the training set. We defined convergence as the satisfaction of at least one of the three following criteria: either the network reached 100% training accuracy on the validation set, the training loss did not improve for 50 epochs, or the training reached 500 epochs. With this setup, we were able to closely reproduce results presented in the paper.

NNGP Reproducing the NNGP result was fairly easy as the authors made their code public, except that their code was based on older python and Tensorflow versions. Surprisingly, we consistently got better results for a fixed configuration than the one presented in the original work. We have not been able to explain this discrepancy.

Experiment 1: Neural Network			
DATASET	CONF. (depth-width-σ_W-σ_b)	MSE	ACC.
CIFAR:1k	5-500-1.29-0.28	0.0954	0.3304 (+79e-4)
CIFAR:10k	5-2000-1.60-1.07	0.0889	0.4529 (-16e-4)

Table 5: Measured MSE and accuracy of the neural network.

Experiment 1: NNGP			
DATASET	CONFIG. (depth-σ_W-σ_b)	MSE	ACC.
CIFAR:1k	7-1.28-0.00	0.0712	0.3612 (+4e-4)
CIFAR:10k	5-2.97-0.28	0.0707	0.4979 (+194e-4)

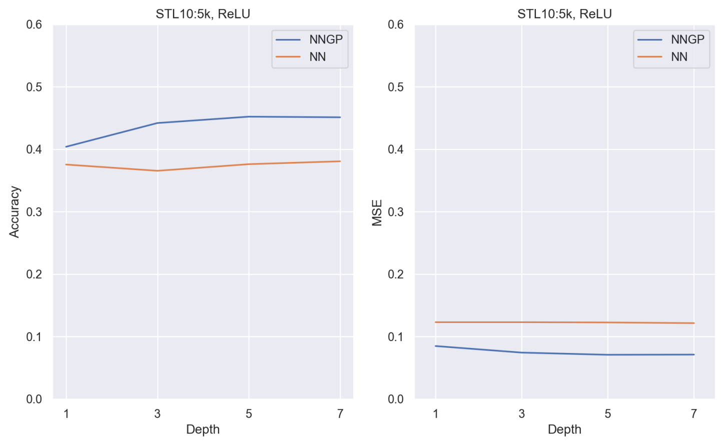
Table 6: Measured MSE and accuracy of the NNGP.

7.2 Experiment 2

In experiment 2 we investigate and compare the classification capabilities of the NN and the NNGP on the STL-10 dataset. For this, we do a grid-search for the optimal parameter (σ_W, σ_b) as well as the width of the NN at increasing depths. We found that the network with largest width (5000) always yielded the best results. For the training we use the full STL-10 dataset consisting of 5000 images, whereas the accuracy and the MSE have been measured on the test set (8000 images). In fig. 1 we plot the performances corresponding to the best parameters found. It is evident that the NNGPs outperform the NNs in terms of accuracy and MSE. The relative outperformance of the best found NNGP compared to the best found NN is 19%, which is also larger than the measured outperformance on the CIFAR dataset found in section 7.1 (10%). Based on our findings, we conclude that gradient based methods seem to struggle as the input size grows, whereas the NNGP, trained by exact Bayesian inference, does not have this limitation.

The best parameter combinations found to create the plot are listed in the appendix (section 9.1).

Figure 1: Classification performance (MSE and accuracy) comparison between NNs and NNGPs on the STL10 dataset.



7.3 Experiment 3: MSOD for convergence estimation

Here, we try to investigate if neural networks with increasing width converge against its corresponding NNGP. For this, we compare the MSOD of the NNs and the NNGP with the MSE of the NNs (see fig. 2). The MSOD has been calculated from networks with increasing width (5, 50, 500, 5000) and fixed NNGP (depth 5, $\sigma_W = 1.60$, $\sigma_b = 1.07$). For reference we plot the MSE of the NNGP. Contrary to the paper, we compare NNs and NNGPs with the exact same parameters, which we think makes more sense if we are trying to test convergence. We train the NN with Adam as in the paper, but also with SGD with a fixed learning rate, which is a setting closed to the theory in Mandt et al. [2017] (see subsection 3.3).

First, we find that the MSOD is lower than the MSE, which shows that on average the NN and the NNGP do the same classification mistakes. The MSOD of the NN trained with vanilla SGD and with Adam also seems to decrease with network width, which shows that the NN is converging in a certain way as the width grows. Unfortunately, it becomes numerically unstable to train very large NNs, which limits the study of convergence through experiments. Lastly, there does not seem to be a significant difference between SGD and Adam.

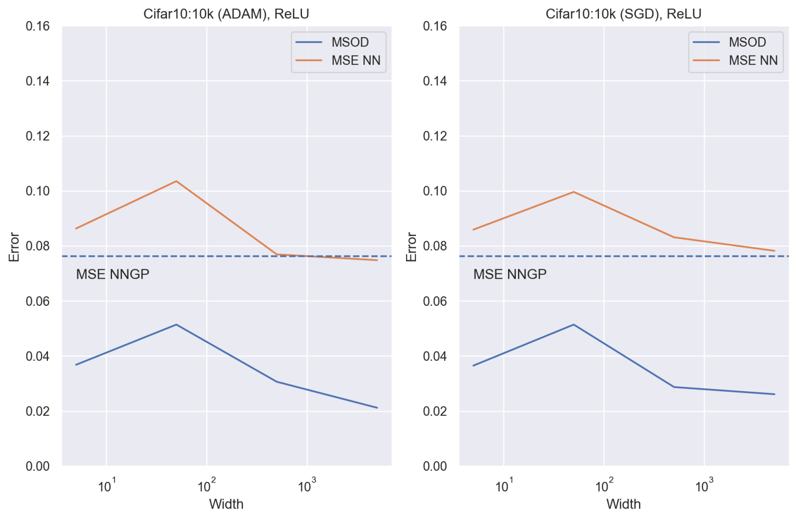
Even though our conclusion is the same as the one of the authors, we think using MSOD to directly compare the outputs of NNs and NNGPs with the *exact* same configuration is more convincing than the experiment conducted by Lee et al..

Results in tabular form can be found in the appendix (section 9.2).

7.4 Experiment 4: NNGP performance evaluation

In this experiment, we wanted to verify whether it was possible to improve the performance of the NN using other learning methods. To this end, we used Adam with a learning rate scheduling based on the validation accuracy evolution and an Early Stopping triggered when the validation accuracy

Figure 2: Comparison of MSE and MSOD at increasing network widths and different optimization methods (ADAM left, SGD right).



stops improving. At the end of the training, we return the networks from the best epoch, also based on the validation accuracy. With this setup, we were able to increase the NN performance and surpass the accuracy of the NNGP results in the paper (but lower than the results that we got for the NNGP). See the table for Experiment 4 and compare it to the one of Experiment 1. These results clearly show that the reported underperfomance of the NN in the paper is due to overfitting and can easily be circumvented by following state-of-the-art training techniques. We thus argue that the reported overperformance of the NNGP only holds in a restricted setting, and that state-of-the-art neural networks model clearly remain the model of choice, especially for large datasets.

Experiment 4: improved NN performance on CIFAR:10k		
CONFIG.	MSE	ACC.
NN-5-2000-1.6-1.07	0.0750	0.4750
NN-7-6000-1.6-1.07	0.0694	0.4907

Table 7: Improved performance of the NN using alternative training methods.

8 Conclusion and Future Work

In this report,we conducted a thorough review of the paper *Deep Neural Networks as Gaussian Processes* by Lee et al. [2017]. We reviewed the literature published before and after the paper. We also presented some limitations in the proof, and in the experiments conducted, which, for us, were mostly not convincing. We implemented a more convincing experiment to show the convergence of the gradient trained NN to the corresponding NNGP. In our preliminary results, gradient based optimization methods seem to struggle as the input width grows, contrary to the exact Bayesian training of the NNGP. Both, conducting a more consistent study of this behavior, as well as coming up with potential remedies, are interesting avenues of future work. Also, further comparing the possible convergence to the NNGP of SGD and Adam (and other additional algorithms like RmsProp or Adagrad) may lead to interesting insights.

9 Appendix

9.1 Experiment 2: Best configurations found for STL-10 classification

Experiment 2: NNGP			
Depth	CONFIG. (σ_W^2-σ_b^2)	MSE	ACC.
1	3.14-0.00	0.0848	0.4040
3	1.28-0.00	0.0743	0.4420
5	3.14-0.00	0.0708	0.4522
7	2.21-0.00	0.0711	0.4512

Table 8: Best NNGP configurations found for different depths.

Experiment 2: NN			
Depth	CONFIG. (width-σ_W^2-σ_b^2)	MSE	ACC.
1	5000-1.28-1.57	0.1233	0.3756
3	5000-1.28-1.57	0.1230	0.3755
5	5000-2.36-0.00	0.1226	0.3761
7	5000-2.36-0.00	0.1215	0.3807

Table 9: Best NN configurations found for different depths.

9.2 Experiment 3: Results

Experiment 3: NN (ADAM)			
Depth	CONFIG. (width-σ_W^2-σ_b^2)	MSE	MSOD.
5	5-1.60-1.07	0.0863	0.0368
5	50-1.60-1.07	0.1035	0.0584
5	500-1.60-1.07	0.0769	0.0306
5	5000-1.60-1.07	0.07478	0.0211

Table 10: Performance of an ADAM trained NN with different widths.

Experiment 3: NN (SGD)			
Depth	CONFIG. (width-σ_W^2-σ_b^2)	MSE	MSOD.
5	5-1.60-1.07	0.0859	0.0365
5	50-1.60-1.07	0.0996	0.0514
5	500-1.60-1.07	0.0831	0.0287
5	5000-1.60-1.07	0.0782	0.02611

Table 11: Performance of an SGD trained NN with different widths.

References

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8141–8150. Curran Associates, Inc., 2019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

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

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204, 2015.

Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

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

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

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8571–8580. Curran Associates, Inc., 2018.

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

Karl Krauth, Edwin V. Bonilla, Kurt Cutajar, and Maurizio Filippone. Autogp: Exploring the capabilities and limitations of gaussian process models, 2016.

Neil D Lawrence. Learning for larger datasets with the gaussian process latent variable model. In *Artificial Intelligence and Statistics*, pages 243–250, 2007.

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.

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

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

Radford M. Neal. Priors for inifinite networks. *Department of Computer Science Toronto - Technical Report CRG-TR-94-1*, 1994.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018.

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*, pages 3360–3368, 2016.

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

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

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9951–9960. Curran Associates, Inc., 2019.