Neural Networks: Trade-offs of Depth and Width

Andrew Searns
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623
abs2157@rit.edu

Abstract

Neural networks have been a topic of considerable interest in the past few years due to their ability to implement real-world functions in a systematic way. Despite their wide applications, the mathematical differences between deep and wide networks have only recently been explored. In this paper, we explore recent results in an effort to explain the differences between deep and wide models from a variety of perspectives.

1 Introduction

Neural networks are a topic of much interest in the field of artificial intelligence and computer science. Despite their popularity, the majority of conventional knowledge still treats them as black boxes or even magical techniques to allow a computer to "learn." While there have been thousands of papers studying many aspects of neural networks, there are many questions which remain either unanswered or left with only experimental evidence in place of rigorous proofs. One such problem that has started to gain significant interest in recent years is the relative power and trade-offs inherent in using different network structures.

Early analysis in machine learning focused on wide networks with only a single hidden layer. Early on in the mathematical development of neural networks, multiple papers emerged which proved that networks with only a single hidden layer are able to approximate any function under a variety of reasonable assumptions [1, 2, 3, 4, 5]. These papers leveraged different techniques, but they generally aimed to show density of various models of neural networks in different spaces of functions. While these papers established the existence of such neural networks, they did not discuss the necessary size of implementing such neural networks for approximating real functions.

For the majority of neural network history, wider networks of relatively few hidden layers were preferred for various reasons. The primary reason being the relative difficulty of training deep neural networks. Specifically, deep networks with the popular logistic sigmoid or hyperbolic tangent activation functions are hard to train due to the issue of vanishing gradient as network depth increases [6]. The ReLU activation function was proposed in 2011 as one technique to fix the vanishing gradient issue in deep networks [7]. Since its proposal, deep ReLU networks have gained much attention. Deep ReLU networks were shown to also be universal approximators [8]. In 2016, skip connections were proposed as an alternative for circumventing the vanishing gradient issue with general activation functions [9]. The corresponding ResNet architecture has been successfully applied to many real world problems.

As deep ReLU networks gained more popularity, experimental papers have repeatedly demonstrated that deeper networks often outperform their shallow counterparts. Surprisingly, this is true even for deep networks learning random labels on input data [10]. While the corresponding high parameter to test data ratio would seem to imply that deep networks exhibit over-fitting behavior due to their

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

over-parameterization, experimental research has also demonstrated that these same deep networks often generalize well on unseen test data.

In the past two years, there has been increased interest in a mathematical understanding of the differences between deep and wide neural networks. In this paper, we hope to summarize these recent results in order to better understand the distinction between different network structures both in training capacity, generalizability, and representation size.

2 Preliminaries

A neural network of L hidden layers with width $d_1, \ldots, d_L$ is a function $f : \mathbb{R}^d \to \mathbb{R}$ characterized by alternations of affine transformations $T^{(k)}$ and nonlinear activation functions σ which act elementwise. That is, $f(x) = (T^{(L+1)} \circ \sigma \circ T^{(L)} \circ \ldots \circ \sigma \circ T^{(1)})(x)$ where $T^{(k)} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ with $d_0 = d$.[1] Common activation functions include the logistic sigmoid ($\sigma(x) = \frac{1}{1+e^{-x}}$) and the ReLU activation function ($\sigma(x) = \max\{0, x\}$).

Let $\mathcal{N}(u, L)$ be the space of functions represented by neural networks of L layers with at most $u = \max_{i \in \{0,1,\ldots,L+1\}} d_i$ nodes per layer.

Function approximations can be quantified by the error with respect to different function norms. Function $f : \mathbb{R}^d \to R$ can be ϵ-approximated in $\mathcal{N}(u, L)$ with respect to the L^p norm on interval $[a, b] \subseteq \mathbb{R}$ if $\inf_{g \in \mathcal{N}(u,L)} \|f - g\|_p = \inf_{g \in \mathcal{N}(u,L)} (\int_a^b |f(x) - g(x)|^p)^{1/p} < \epsilon$. In particular, the L^1 norm corresponds to total error and is particularly useful for lower bounding error as a non-zero L^1 distance implies a set of positive measure on which f and g differ. Likewise, the L^∞ norm is useful for upper bounding error as it is sensitive to even a single point of difference. This definition of error slightly extends the notion presented by Chatziafratis et al. by using the infimum instead of the minimum as it is possible that no such function yields a minimum approximation (see Figure 1) [12]. Furthermore, we observe that under the L^∞ error, this approximation also generalizes the definition of ϵ-approximation (identically uniform-approximation) as posed by Rolnick and Tegmark [13].

3 Depth-Width Gaps For Smooth Activation Functions

Both deep and wide neural networks are universal approximators; however, these approximation theorems show the existence of networks for any ϵ-approximation only when the allowable width or depth is unbounded. For bounded size networks, there is an apparent difference between wide networks and deep networks. To provide intuition on the difference between the two network structures, we first highlight a result by Lin et al. [14] and an extension of this result by Rolnick and Tegmark [13]. Lin et al. proved that activation functions which have non-zero coefficients in their Taylor expansion around 0 can be used to represent multiplication over two network layers. Their construction, shown in Figure 1, uses 7 nodes for each multiplication. For the sake of intuition, we provide a proof of their construction.

Theorem 1 (Multiplication [14])**.** *If $f : \mathbb{R}^2 \to \mathbb{R}$ is a neural network with 4 nodes in a single hidden layer with a smooth non-linear activation function σ, then f can ϵ approximate multiplication arbitrarily well.*

Proof. By the smoothness of σ, we may write its Taylor expansion about 0: $\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 \frac{x^2}{2} + O(x^3)$. Since σ is non-linear, we may assume that $\sigma_2 \neq 0$.

[1] This definition parallels the structure presented by Rahaman et al. [11].

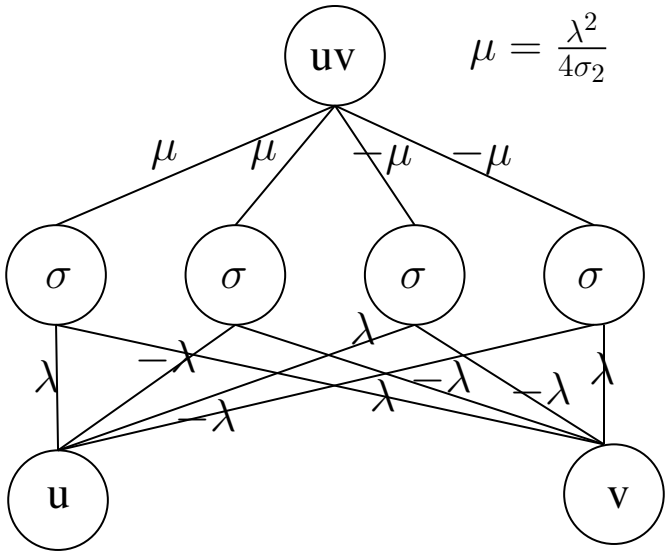

Figure 1: The multiplication gate of Lin et al. [14]

$$
\begin{aligned}
m(u,v) &\equiv \frac{\sigma(u+v)+\sigma(-u-v)-\sigma(u-v)-\sigma(-u+v)}{4\sigma_2} \\
&= \frac{\frac{\sigma_2}{2}(u^2+2uv+v^2+u^2+2uv+v^2-u^2+2uv-v^2-u^2+2uv-v^2)}{4\sigma_2} \\
&= \frac{\sigma_2(8uv)}{4\sigma_2} \\
&= uv[1+O(u^2+v^2)]
\end{aligned}
$$

When $|u|$ and $|v|$ are small, this gives a very good approximation of uv. By scaling the inputs to the hidden layer, $|u|$ and $|v|$ can be made arbitrarily small. The corresponding outputs can be multiplied by an appropriate scalar to correct the magnitude. □

The construction of Lin et al. was extended by Rolnick and Tegmark to compute x^2 with less nodes. Their extension observes that $(x-x)=0$ which means that both the $(u-v)$ and the $(-u+v)$ nodes can be removed in the hidden layer.

This multiplication construction generalizes both in width and in depth; however, the techniques required to do so are different. For deeper neural networks, it is possible to stack these multiplication gates together along with identity gates to create polynomial functions. For wider neural networks, higher degree polynomials require more nodes to implement than with deep networks. Intuitively, implementing a product of three variables requires a sum over each of the combinations of cubic terms of $\pm u$, $\pm v$, and $\pm w$. Lin et al. proved a lower bound on multiplying d inputs to an arbitrary accuracy of 2^d hidden nodes if only a single hidden layer is used [14]. While this proves that polynomials of d variables have a width-depth gap in the number of nodes required to implement them, it does not explain what happens in between 1 and $\log(d)$ hidden layers when approximating the product $\prod_{i=1}^{d} x_i$. Rolnick and Tegmark provide a construction to give an upper bound for the total number of nodes required as a function of the number layers. They show that it is possible to implement this product using $O(d^{(L-1)/L} \cdot 2^{d^{1/L}})$ [13]. As L approaches $\log_2(d)$ the total number of nodes becomes polynomial in d; however for $L=1$, the total number of nodes required is at least exponential in d.

Univariate polynomials suffer a similar issue despite the reduction in size of the product gates. In order to implement a univariate polynomial of degree d using a single hidden layer, a linear combination of each $\sigma(x^i)$ for $i=1$ through d is necessary [13]. In turn, univariate polynomials can be implemented linearly in the number of nodes. However, using the square gate and the binary representation of d, it

is possible to compute x^d using only a logarithmic number of nodes in a deep network. These bounds were proved tight by Rolnick and Tegmark showing a depth-width gap for univariate polynomials.

While multiplication can be done more efficiently in deeper networks using chains of multiplication gates, it is not obvious that deeper networks are universally more efficient than wider networks. To this end, we show that in very small networks it is possible for a deeper network with three parameters to represent every function that can be represented by a wider network with four parameters. The corresponding models are shown in Figure 2.

Theorem 2. *For smooth, invertible, and non-linear activation functions,* $\mathcal{N}(2,1) \subseteq \mathcal{N}(1,2)$.

Proof. Let $f(x) \in \mathcal{N}(2,1)$ be such that $f(x) = \sigma(c\sigma(ax)+d(\sigma(bx))$. We show that $f(x) \in \mathcal{N}(1,2)$ by proving the existence of a network $g(x) = \sigma(\hat{c}\sigma(\hat{b}\sigma(\hat{a}x)))$ for which $f(x) = g(x)+O(x^3)$ where the x^3 term can be made arbitrarily small by scaling the parameter $\hat{a}$ downwards and correspondingly scaling the parameter $\hat{c}$ upwards.

Since σ is invertible, we may ignore the last activation function in both networks yielding a necessary equation that $c\sigma(ax) + d\sigma(bx) = \hat{c}\sigma(\hat{b}\sigma(\hat{a}x)) + O(x^3)$. Since σ is nonlinear and smooth, we may use its Taylor expansion $\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + O(x^3)$.

We first analyze the left hand side of the equation:

$$c\sigma(ax) + d\sigma(bx) = (c+d)\sigma_0 + (ac+bd)\sigma_1 x + (a^2c + b^2d)\sigma_2 x^2 + O(x^3)$$

Likewise, the right hand side of the equation can be expanded

$$\begin{aligned}\hat{c}\sigma(\hat{b}\sigma(\hat{a}x)) = \hat{c}\sigma_0 + \hat{b}\hat{c}\sigma_0\sigma_1 + \hat{b}^2\hat{c}\sigma_0^2\sigma_2 + \\ \hat{a}\hat{b}\hat{c}\sigma_1^2 x + 2\hat{a}\hat{b}^2\hat{c}\sigma_0\sigma_1\sigma_2 x + \\ \hat{a}^2\hat{b}\hat{c}\sigma_1\sigma_2 x^2 + \hat{a}^2\hat{b}^2\sigma_1^2\sigma_2 x^2 + 2\hat{a}^2\hat{b}^2\hat{c}\sigma_0\sigma_2^2 x^2 + O(x^3)\end{aligned}$$

Since our goal is to compute a function $g(x)$ which is identical to $f(x)$ up to cubic order terms, we have a system of equations in terms of the coefficients of the powers of x:

$$\begin{aligned}c + d &= \hat{c}\sigma_0 + \hat{b}\hat{c}\sigma_0\sigma_1 + \hat{b}^2\hat{c}\sigma_0^2\sigma_2 \\ ac + bd &= \hat{a}\hat{b}\hat{c}\sigma_1^2 + 2\hat{a}\hat{b}^2\hat{c}\sigma_0\sigma_1\sigma_2 \\ a^2c + b^2d &= \hat{a}^2\hat{b}\hat{c}\sigma_1\sigma_2 + \hat{a}^2\hat{b}^2\sigma_1^2\sigma_2 + 2\hat{a}^2\hat{b}^2\hat{c}\sigma_0\sigma_2^2\end{aligned}$$

Using the first equation, we can solve for $\hat{c}$ in terms of $\hat{b}$:

$$\hat{c} = \frac{c+d}{1 + \hat{b}\sigma_1 + \hat{b}^2\sigma_0\sigma_2}$$

Substituting this into the second equation lets us compute $\hat{a}$ in terms of $\hat{b}$:

$$\hat{a} = \frac{(ac+bd)(1 + \hat{b}\sigma_1 + hatb^2\sigma_0\sigma_2)}{\hat{b}(c+d)(\sigma_1 + 2\hat{b}\sigma_0\sigma_2)}$$

Using both the definitions of $\hat{a}$ and $\hat{c}$ in terms of $\hat{b}$, we can solve for $\hat{b}$ in terms of a, b, c, d, σ_0, σ_1, and σ_2 using the third equation.

$$\begin{aligned}(\hat{b}^3(4\sigma_0^2\sigma_2^2(c+d)^2) + \hat{b}^2(4\sigma_0\sigma_1\sigma_2(c+d)^2) + \hat{b}\sigma_1^2(c+d)^2)\frac{a^2c+b^2d}{(ac+bd)^2} \\ = \hat{b}^2\sigma_0\sigma_2(c+d) + \hat{b}\sigma_1(c+d) + \hat{b}^5\sigma_0^2\sigma_1^2\sigma_2^2 + \hat{b}^4 2\sigma_0\sigma_1^3\sigma_2 + \hat{b}^3\sigma_1^4 + \hat{b}^3 2\sigma_0\sigma_1^3\sigma_2 + \hat{b}^3 2\sigma_0\sigma_1^2\sigma_2 + \\ \hat{b}^2 2\sigma_1^3 + \hat{b}2\sigma_1^3 + \hat{b}\sigma_1^2 + \hat{b}^3\sigma_0^2\sigma_2^2(c+d) + \hat{b}^2\sigma_0\sigma_1\sigma_2(c+d) + \hat{b}\sigma_0\sigma_2(c+d) + (c+d)\end{aligned}$$

While this equation might seem unwieldy at first, we observe that it is a fifth degree polynomial in $\hat{b}$ with non-zero roots. Observing $\lim_{\hat{b}\to-\infty} p(\hat{b}) = -\infty$ and that $\lim_{\hat{b}\to\infty} p(\hat{b}) = \infty$ and using the intermediate value theorem for continuous functions, we see that this equation has a real root. Checking by hand shows that this root does not correspond to a value which makes the denominator of either $\hat{c}$ or $\hat{a}$ 0. Lastly, we observe that when $c + d = 0$, it is possible to make a perturbation $\tilde{c}$ such that $\tilde{c} + d \neq 0$ and $|\tilde{c} - c| << \epsilon$. □

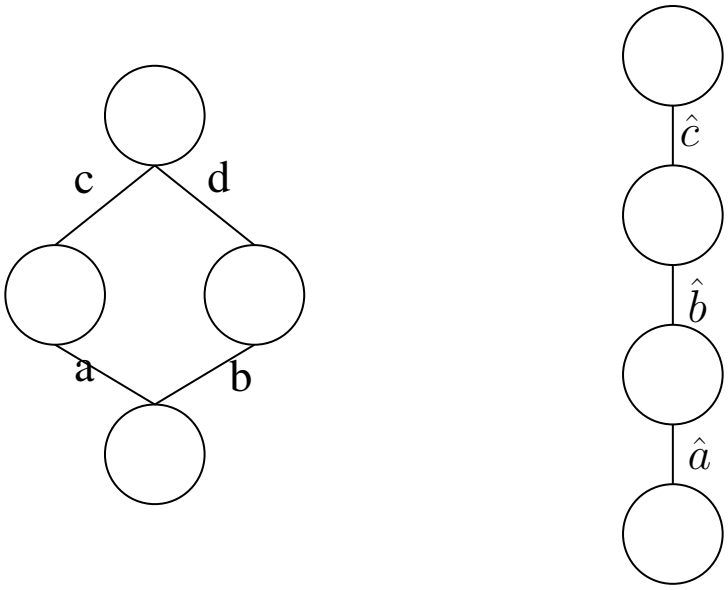

Figure 2: Two small network models

3.1 Depth-Width Gaps for ReLUs

Up to this point, we have only discussed activation functions which are smooth. In practice many deep networks are trained using ReLU activation functions. In this setting, it is well known that the output functions of ReLU networks are piecewise linear and networks in $\mathcal{N}(u, L)$ can represent at most $(2u)^L$ linear pieces [15]. In this setting, it is comparatively simple to demonstrate a depth-width gap between deep and wide networks. In order to well approximate a function which has many segments $\Omega((2u)^L)$, one necessarily either increases u or L. The number of nodes needed per layer decreases exponentially as the number of layers increases.

Perhaps one surprising result is that very little information is needed about a function f to generate a function which cannot be well approximated by small neural networks (subexponential in width or sublinear in depth). In fact, if f is a Lipschitz continuous function with a point of odd period ($f^{2k+1}(x) = x$ for $k > 1$), then as long as the Lipschitz constant is tight to its osciliations when iterated, it f^t will generate a function which exhibits the depth-width gap [12]. This may seem like many requirements on f, but on $[-1, 1]$ the functions $f(x) = k|x|$ for $k \in (\sqrt{2}, \phi)$ satisfy these requirements where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

4 Training Deep Versus Wide Networks

Despite the apparent benefits of deep networks over wide networks in representation capabilities, deep networks have still be notoriously harder to train. However, deep neural networks trained via gradient descent have repeatedly shown to converge to zero training loss given sufficient training time. In fact, it it was recently shown by Zou et al. that deep neural networks will converge in polynomial time with respect to d, L, ϕ, and $\log(1/\epsilon)$ where ϕ represents the minimum separation distance $\|x - y\|$ between input points of different classes and ϵ is the allowed training error [16]. While the time is polynomial in these terms, the polynomials are rather large. For example, in order for gradient descent to converge to zero training loss, they show a lower bound on u as $\tilde{\Omega}(n^{26}L^{38}/\phi^8) \cdot \Omega(\log(\frac{1}{\epsilon})$. Fortunately, when these very wide networks are trained for a minimum of $\tilde{\Omega}(n^4/m\phi) \cdot \Omega(\log(\frac{1}{\epsilon})$ under certain starting assumptions, the training error will decrease below ϵ where m is the minimum number of nodes in any hidden layer.

Conventional wisdom would imply that zero training loss is a sign of overfitting; however, this is often not the case with deep neural networks in practice. These networks historically generalize very well for unseen test data. This is largely due to the fact that deep networks have natural implicit

regularization when trained through gradient descent. Rahaman et al. recently demonstrated that deep ReLU networks trained via gradient descent learn low-frequency components of the underlying functions before learning high-frequency components [11]. This work is not the first to study Fourier spectrum analysis of neural networks, but they provide experimental evidence that the low-frequency bias of deep networks implies that these networks are less subject to misclassification due to high-frequency noise.

Despite the relative advantages of deep neural networks, it is actually possible to train infinitely wide neural networks making them theoretically feasible use. A paper by Radford Neal showed that infinitely wide neural networks can be viewed as a Gaussian process [17]. Using Bayesian inference, it is possible to directly compute the posterior distribution corresponding to a training step of the infinite neural network. This concept was extended by Jacot, Gabriel, and Hongler to train infinite width neural networks using kernel approximation [18]. This technique results in both an efficient training procedure for infinite width networks as well as a mathematical explanation for why early-stopping can act as a regularizer. While this process can be done for infinite width networks, it is still mathematically challenging to convert a neural network model to its corresponding infinite width form as a Gaussian process. Recent work has started to combine the approaches of simultaneously deeper and wider neural networks. Lee et al. prove that deep neural networks also can be modeled by Gaussian processes in the infinite width form [19]. They show experimentally that the Gaussian processes outperform the finite width network models and they use this model to correlate Gaussian process uncertainty with network prediction error.

5 Conclusions

As deep networks become more computationally feasible to train, they have gained significant interest for their theoretical properties. While practical training time is still a significant issue for deep neural networks, their convergence results as well as representation power still make them very powerful tools in the machine learning toolbox. As computing power continues to increase, it is very likely that neural networks will continue to grow deeper and wider in order to tackle increasingly complex problems. While deep infinite width networks are still relatively new, it is very likely that they will gain more traction as tools become available to automatically compute the effective Gaussian processes. In fact, at the time of this writing, there is currently a paper under review at ICLR describing a Python implementation of automatic Neural Tangent Kernel methods [20].

References

[1] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[2] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[3] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[4] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14(1):115–133, 1994.

[5] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[7] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[8] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[11] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[12] Vaggos Chatziafratis, Sai Ganesh Nagarajan, and Ioannis Panageas. Better depth-width trade-offs for neural networks through the lens of dynamical systems. *arXiv preprint arXiv:2003.00777*, 2020.

[13] David Rolnick and Max Tegmark. The power of deeper networks for expressing natural functions. In *International Conference on Learning Representations*, 2018.

[14] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

[15] Matus Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

[16] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 109(3):467–492, Oct 2019.

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

[18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[19] 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.

[20] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A Alemi, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. *arXiv preprint arXiv:1912.02803*, 2019.