Network Degeneracy as an Indicator of Training Performance: Comparing Finite and Infinite Width Angle Predictions

Cameron Jakub CJAKUB@UOGUELPH.CA
University of Guelph, Ontario, Canada

Mihai Nica NICAM@UOGUELPH.CA
University of Guelph, Ontario, Canada

Abstract

Neural networks are powerful functions with widespread use, but the theoretical behaviour of these functions is not fully understood. Creating *deep* neural networks by stacking many layers has achieved exceptional performance in many applications and contributed to the recent explosion of these methods. Previous works have shown that depth can exponentially increase the expressibility of the network [3, 8]. However, as networks get deeper and deeper, they are more susceptible to becoming *degenerate*. We observe this degeneracy in the sense that on initialization, inputs tend to become more and more correlated as they travel through the layers of the network. If a network has too many layers, it tends to approximate a (random) constant function, making it effectively incapable of distinguishing between inputs. This seems to affect the training of the network and cause it to perform poorly, as we empirically investigate in this paper. We use a simple algorithm that can accurately predict the level of degeneracy for any given fully connected ReLU network architecture, and demonstrate how the predicted degeneracy relates to training dynamics of the network. We also compare this prediction to predictions derived using infinite width networks.

1. Introduction and Main Results

Our previous work *Depth Degeneracy in Neural Networks: Vanishing Angles in Fully Connected ReLU Networks* [6] theoretically studied the "large depth degeneracy" phenomenon for finite width ReLU networks. This workshop paper extends the work of that paper, and uses our previous theoretical results as an input into experiments that investigate how the level of degeneracy can influence training. Consider two inputs fed into an initialized feed-forward ReLU network with depth L and layer widths $n_\ell,\ 1 \leq \ell \leq L$ (see Appendix A.1 for a full definition of the network). We assume the network is initialized with independent Gaussian weights so that the network is on the "edge of chaos" [5, 10], and that the angle between inputs is defined using the inner product on $\mathbb{R}^{n_\ell}$ in the standard way. Given this setup, Algorithm 1 (established theoretically in [6]) provides us with a simple method to accurately predict the angle between those inputs after travelling through the layers of the network on network initialization up to an error of size $\mathcal{O}(n_\ell^{-2})$ in layer ℓ.

Algorithm 1 predicts the angle at the final layer on initialization based solely on the network architecture $n_1, n_2, \ldots n_L$. If all inputs into an initialized network tend to be highly correlated by the final layer, this could make it difficult for the network to distinguish the differences between inputs and therefore harder to train. Figure 1 demonstrates how networks which exhibit this type of degeneracy empirically tend to perform worse *after training*, and seem to train less consistently than networks which can better distinguish between inputs on initialization.

Algorithm 1: Angle prediction between inputs for a feed-forward ReLU network with depth L and layer widths $n_\ell,\ 1 \leq \ell \leq L$. The function $\mu(\theta, n)$ is given in Theorem 3.

```
$\theta^0 =$ angle between inputs ;
for $\ell = 0, \ldots, L-1$ do
    $x = \mu(\theta^\ell, n_\ell)$ ;                // $x$ represents $\mathbf{E}[\ln(\sin^2(\theta^{\ell+1}))]$
    $\theta^{\ell+1} = \arcsin(e^{\frac{x}{2}})$ ;
end
Final angle $= \theta^L$
```

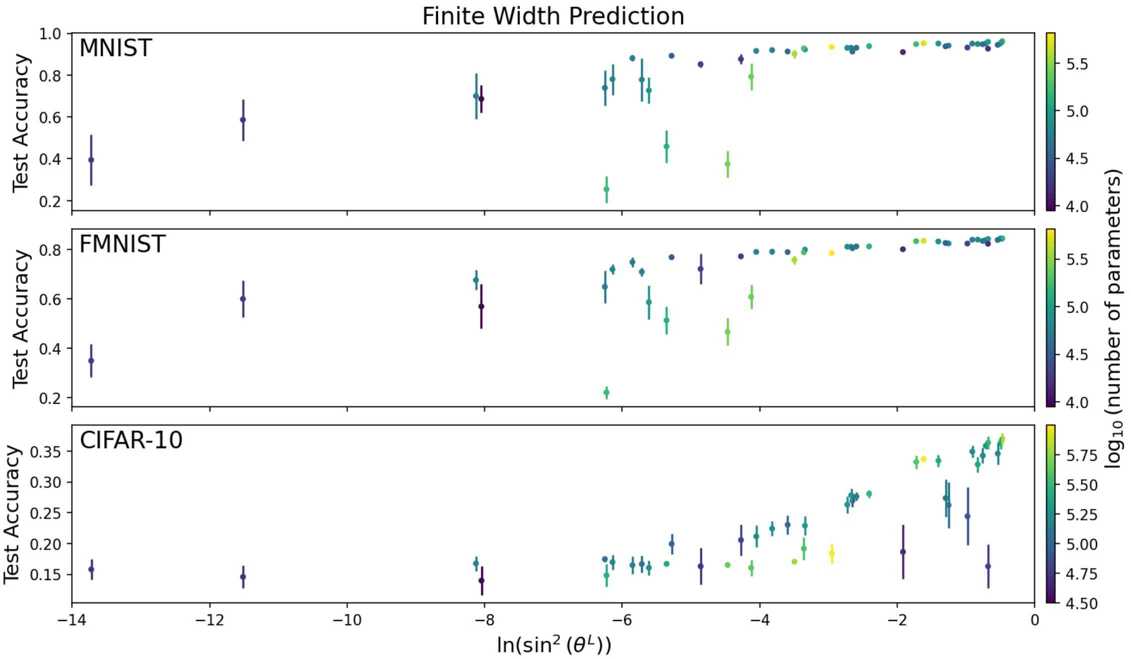


Figure 1: We compare 45 different network architectures trained on the MNIST [2], Fashion-MNIST [11], and CIFAR-10 [7] datasets 10 times each. Using the architecture of the network and Algorithm 1, we predict the angle between 2 orthogonal inputs at the final output layer of the network on initialization. We express the angle as $\ln(\sin^2(\theta^L))$, to follow the form used when developing the finite width approximations. The angle is plotted against the accuracy of each network on the test data after training, with error bars representing a 95% confidence interval across the 10 runs. All networks are trained using 1 epoch, batch size $= 100$, categorical cross-entropy loss, the ADAM optimizer, and default learning rate in the Keras module of TensorFlow [1]. See Appendix A.2 for details on all of the network architectures used.

When Algorithm 1 predicts that the network architecture forces inputs to become highly correlated on initialization, this serves a warning that the network may train poorly. Before going through the computationally expensive process of training many networks to assess their performance, this prediction could be used to quickly filter out network architectures that are unlikely to perform well. The simplicity and efficiency of the algorithm may lend itself well to applications in neural architecture search, and would be an interesting starting point for more detailed experiments and/or theoretical explanations about training.

1.1. Finite Width - Small Angle Evolution

Since the effect of each layer is independent of everything previous, θ^ℓ can be thought of as a Markov chain evolving as layer number ℓ increases. As expected by the aforementioned "large depth degeneracy" phenomenon, the angle θ^ℓ tends towards 0 as the current layer ℓ goes towards infinity. This indicates that the hidden layer representation of *any* two inputs in a deep neural network becomes closer and closer to co-linear as depth increases. We found a simple update rule in Jakub and Nica [6] which predicts how the angle between inputs evolves, given below in Approximation 1. The algorithm works well for finite sized networks because the errors are controlled up to size $\mathcal{O}(n^{-2})$ in the layer sizes.

Approximation 1 *(Finite width small angle update rule) For small angles $\theta^\ell \ll 1$ and large layer widths $n_\ell \gg 1$, the angle $\theta^{\ell+1}$ at layer $\ell+1$ is well approximated by*

$$\ln \sin^2(\theta^{\ell+1}) \approx \ln \sin^2(\theta^\ell) - \frac{2}{3\pi}\theta^\ell - \rho(n_\ell), \tag{1}$$

where $\rho(n_\ell)$ is a constant which depends on the width n_ℓ of layer ℓ, namely:

$$\rho(n) := \ln\left(\frac{n+5}{n-1}\right) - \frac{10n}{(n+5)^2} + \frac{6n}{(n-1)^2} = \frac{2}{n} + \mathcal{O}\left(n^{-2}\right). \tag{2}$$

Remark 1 *Approximation 1 comes from a simplification of more precise formulas for the mean and variance of the variable $\ln(\sin^2(\theta^\ell))$, which are stated in Theorem 3. Specifically, Approximation 1 is derived from a linear approximation of $\mu(\theta, n)$ in Theorem 3 about $\theta = 0$. For θ^ℓ sufficiently small, line 3 of Algorithm 1 could be replaced with the simpler linear update rule given in Approximation 1: $x = \ln \sin^2 \theta^\ell - \frac{2}{3\pi}\theta^\ell - \rho(n_\ell)$.*

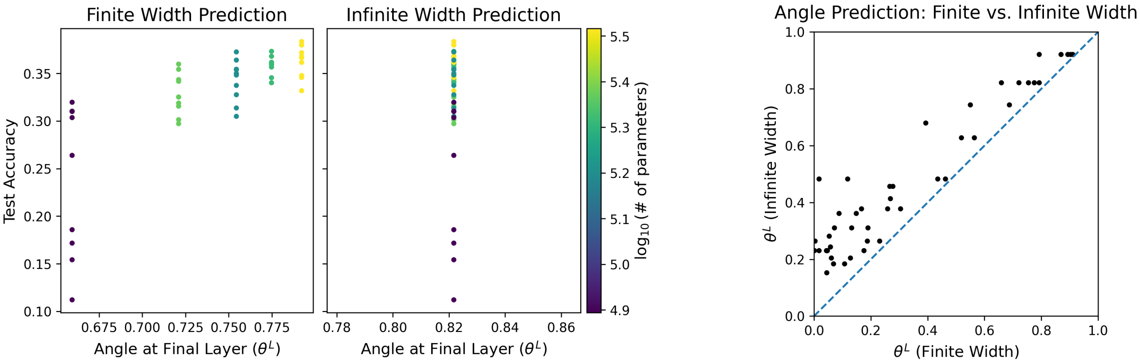
1.2. Comparison to Infinite Width Networks

The angle degeneracy phenomenon has been studied in previous works for networks in the limit of infinite width [4, 5, 9, 10]. The infinite width case uses the law of large numbers and thereby disregards any random fluctuations in $\theta^{\ell+1}$ given θ^ℓ. These random fluctuations, though small, can accumulate over many layers leading to inaccurate predictions for finite width networks (see Figure 3). The infinite width update rule is given below in Approximation 2.

Approximation 2 *(Infinite width update rule) In the limit that the width of each layer tends to infinity, the* infinite width approximation *for the angle $\theta^{\ell+1}$ given θ^ℓ is*

$$\cos\left(\theta^{\ell+1}\right) = \frac{\sin(\theta^\ell) + (\pi - \theta^\ell)\cos(\theta^\ell)}{\pi}. \tag{3}$$

Another issue with using the infinite width prediction to study finite width networks is that all networks with the same depth are treated exactly the same, since it does not take into account the width of each layer. Both the depth of the network and the width of each layer affect how the angle between inputs propagates layer-by-layer through the network. Figure 2-Left illustrates how our method yields different angle predictions for different architectures with the same depth, while the infinite width method does not. Figure 2-Right shows the how the infinite width predictions differ from our “finite width” method which takes into account fluctuations of size $\mathcal{O}(n^{-1})$ in each layer.


Finite Width Prediction

Infinite Width Prediction

Angle Prediction: Finite vs. Infinite Width

Figure 2: Left: Comparison of the finite and infinite width predictions for 5 network architectures with a depth of $L = 3$ trained 10 times each on the CIFAR-10 dataset [7]. The infinite width predicts the same final angle for all networks, since it only depends on network depth. Right: Using the same 45 network architectures as in Figure 1, we plot a comparison of the predicted angle θ^L using Algorithm 1 (finite width) versus the infinite width prediction. We see that the infinite width prediction tends to underestimate the rate at which θ^ℓ tends towards 0.

2. Mathematical Theory

Derivations of Approximation 1 and 2 rely on calculating the joint moments of the ReLU function applied to correlated Gaussian variables (Approximations 1 and 2 are derived in Section 2 and Appendix A.9 of Jakub and Nica [6], respectively). We provide a very brief overview of the main results of that theory here. A core ingredient in those calculations is the joint moments defined below, which we think of as a family of “J” functions.

Definition 2 *Let $G,\ \hat{G}$ be marginally $\mathcal{N}(0,1)$ random variables with correlation $\mathbf{E}[G\hat{G}] = \cos\theta$, and let $\varphi(x) = \max\{0,x\}$ be the ReLU activation function. Then, we define an infinite family of J functions as*

$$J_{a,b}(\theta) = \mathbf{E}[\varphi^a(G)\varphi^b(\hat{G})].$$

With this definition of $J_{a,b}(\theta)$, Approximation 2 is first derived by the law of large numbers as $\cos(\theta^{\ell+1}) = 2J_{1,1}(\theta^\ell)$, and then $J_{1,1}$ is explicitly evaluated to obtain Approximation 2. By using combinatorial expansions, one can get more accurate expansions for θ^ℓ which involve even higher order $J_{a,b}$ (i.e. $a, b \geq 2$) appearing as $\mathcal{O}(n^{-1})$ corrections to the infinite width update rule. Solving for the higher order, mixed J functions thereby allows us to further correct the infinite width update rule to an update rule that is more accurate for finite width networks. With this approach, we can not only predict the expected value of θ^ℓ at each layer, but we can also study its variance, as shown in Theorem 3 below. Consequently, the resulting normal approximation Approximation 3 matches *both* the mean and *the variance* of actual neural networks remarkably well; see Figure 3 for Monte Carlo simulations.

Theorem 3 *Conditionally on the angle θ^ℓ in layer ℓ, the mean and variance of $\ln\sin^2(\theta^{\ell+1})$ obey the following limit as the layer width $n_\ell \to \infty$*

$$\begin{aligned}\mathbf{E}[\ln\sin^2(\theta^{\ell+1})] =& \mu(\theta^\ell, n_\ell) + \mathcal{O}(n_\ell^{-2}), \quad \mathbf{Var}[\ln\sin^2(\theta^{\ell+1})] = \sigma^2(\theta^\ell, n_\ell) + \mathcal{O}(n_\ell^{-2}), && (4)\\ \mu(\theta, n) =& \ln\sin^2\theta - \frac{2}{3\pi}\theta - \rho(n) - \frac{8\theta}{15\pi n} - \left(\frac{2}{9\pi^2} - \frac{68}{45\pi^2 n}\right)\theta^2 + \mathcal{O}(\theta^3), && (5)\\ \sigma^2(\theta, n) =& \frac{8}{n} - \frac{64}{15\pi}\frac{\theta}{n} - \left(8 + \frac{296}{45\pi}\right)\frac{\theta^2}{n} + \mathcal{O}\left(\theta^3\right), && (6)\end{aligned}$$

where $\rho(n)$ is as defined in (2).

Approximation 3 *Conditional on the value of θ^ℓ, the angle at layer $\ell+1$ is well approximated by a Gaussian random variable*

$$\ln\sin^2(\theta^{\ell+1}) \overset{d}{\approx} \mathcal{N}(\mu(\theta^\ell, n_\ell), \sigma^2(\theta^\ell, n_\ell)). \qquad (7)$$

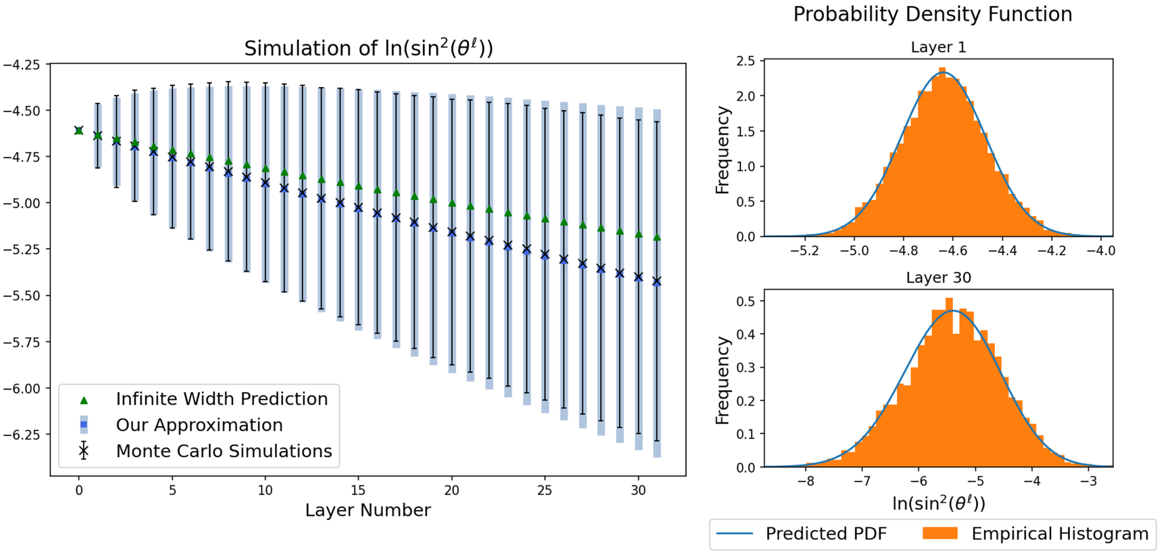


Figure 3: Simulations generated using 5000 independently initialized networks with uniform hidden layer widths $n_\ell = 256$. We generate Monte Carlo samples by feeding 2 inputs with initial angle $\theta^0 = 0.1$ into these networks on initialization. Left: We compare the mean and standard deviation of Approximation 3 vs Monte Carlo samples vs the infinite width prediction as in Approximation 2 (which predicts 0 variance and is less accurate at predicting the mean of $\mathbf{E}[\ln(\sin^2(\theta^\ell))]$). Right: Using Approximation 3, we compare the predicted probability density function of $\ln(\sin^2(\theta^\ell))$ to the Monte Carlo simulations.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[3] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Annual Conference Computational Learning Theory*, 2015.

[4] Boris Hanin. Random fully connected neural networks as perturbatively solvable hierarchies, 2023. URL https://arxiv.org/abs/2204.01058.

[5] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In *International Conference on Machine Learning*, 2019.

[6] Cameron Jakub and Mihai Nica. Depth degeneracy in neural networks: Vanishing angles in fully connected ReLU networks on initialization, 2023.

[7] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[8] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/148510031349642de5ca0c544f31b2ef-Paper.pdf.

[9] Daniel A. Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge University Press, 2022. doi: 10.1017/9781009023405.

[10] Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=H1W1UN9gg.

[11] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. URL http://arxiv.org/abs/1708.07747.

Appendix A.

A.1. Definition of the Network

Symbol	**Definition**
$x \in \mathbb{R}^{n_{in}}$	Input (e.g. training example) in the input dimension $n_{in} \in \mathbb{N}$
$\ell \in \mathbb{N}$	Layer number. $\ell = 0$ is the input
$n_\ell \in \mathbb{N}$	Width of hidden layer ℓ (i.e. number of neurons in layer ℓ)
$W^\ell \in \mathbb{R}^{n_{\ell+1} \times n_\ell}$	Weight matrix for layer ℓ. Initialized with iid standard Gaussian entries $W^\ell_{a,b} \sim \mathcal{N}(0,1)$
$\varphi : \mathbb{R}^n \to \mathbb{R}^n$	Entrywise ReLU activation function $\varphi(x)_i = \varphi(x_i) = \max\{x_i, 0\}$
$z^\ell(x) \in \mathbb{R}^{n_\ell}$	Pre-activation vector in the ℓ^{th} layer for input x (a.k.a logits of layer ℓ) $z^1(x) := W^1 x, \qquad z^{\ell+1}(x) := \sqrt{\frac{2}{n_\ell}} W^{\ell+1}\varphi(z^\ell(x)).$
$\theta^\ell \in [0, \pi]$	Angle between φ^ℓ_α and φ^ℓ_β defined by $\cos(\theta^\ell) := \frac{\langle \varphi^\ell_\alpha, \varphi^\ell_\beta \rangle}{\|\varphi^\ell_\alpha\| \|\varphi^\ell_\beta\|}$

Table 1: Definition and notation used for fully connected ReLU neural networks.

Given the notation in Table 1, a feed-forward ReLU network with L layers is defined as follows:

$$z^1 = W^1 x, \qquad z^{\ell+1} = \sqrt{\frac{2}{n_\ell}} W^{\ell+1} \varphi(z^\ell), \qquad f_L(x) = z^L. \tag{8}$$

A.2. Network Architectures

This section details the architectures of the 45 different neural networks used to produce Figure 1.

#	Depth	Avg. Width	# Parameters		Avg. Test Accuracy ± Standard Deviation		
			(F)MNIST	CIFAR	MNIST	FMNIST	CIFAR
1	2	50	58880	165790	0.924 ± 0.007	0.79 ± 0.02	0.211 ± 0.029
2	2	85	57350	135510	0.837 ± 0.051	0.709 ± 0.028	0.276 ± 0.011
3	2	200	19930	54250	0.878 ± 0.009	0.721 ± 0.098	0.163 ± 0.048
4	2	25	138300	201600	0.94 ± 0.004	0.812 ± 0.009	0.229 ± 0.025
5	2	125	31725	88925	0.89 ± 0.005	0.768 ± 0.013	0.199 ± 0.027
6	3	25	43990	114550	0.928 ± 0.008	0.812 ± 0.013	0.167 ± 0.022
7	3	50	62830	173280	0.916 ± 0.002	0.79 ± 0.012	0.224 ± 0.019
8	3	100	59700	96756	0.952 ± 0.004	0.839 ± 0.003	0.27 ± 0.016
9	3	67.67	87200	309900	0.924 ± 0.006	0.799 ± 0.011	0.281 ± 0.011
10	3	50	17310	189100	0.553 ± 0.181	0.599 ± 0.119	0.263 ± 0.022
11	4	30	369400	366150	0.877 ± 0.052	0.757 ± 0.026	0.192 ± 0.029
12	4	75	99400	105060	0.957 ± 0.003	0.842 ± 0.006	0.23 ± 0.025
13	5	21	74700	51630	0.931 ± 0.005	0.811 ± 0.009	0.146 ± 0.029
14	6	55	8840	976400	0.715 ± 0.088	0.569 ± 0.146	0.337 ± 0.008
15	6	87.5	169400	398200	0.949 ± 0.008	0.833 ± 0.007	0.332 ± 0.018
16	10	10	79020	180010	0.951 ± 0.003	0.832 ± 0.01	0.278 ± 0.018
17	10	100	64850	122050	0.939 ± 0.004	0.824 ± 0.008	0.262 ± 0.059
18	10	200	54170	262060	0.933 ± 0.005	0.81 ± 0.014	0.335 ± 0.016
19	10	17.5	49920	1002300	0.794 ± 0.052	0.648 ± 0.106	0.184 ± 0.026
20	11	34.55	518800	31720	0.955 ± 0.006	0.835 ± 0.011	0.14 ± 0.037
21	11	35	21100	269195	0.93 ± 0.005	0.823 ± 0.007	0.363 ± 0.016
22	13	42	36420	328200	0.91 ± 0.008	0.789 ± 0.01	0.364 ± 0.016
23	15	30	41844	174100	0.92 ± 0.004	0.805 ± 0.011	0.349 ± 0.015
24	15	50	13860	235650	0.909 ± 0.005	0.8 ± 0.012	0.328 ± 0.02
25	15	75	16580	206848	0.927 ± 0.003	0.823 ± 0.007	0.359 ± 0.009

Table 2: Summary of the architectures of the first 25 neural networks used in Figure 1, as well as their performance on the test datasets. Note that the number of parameters differs between the (F)MNIST and CIFAR-10 datasets due to the fact that CIFAR-10 images are in colour requiring 3 colour channels, while the MNIST and FMNIST images are in grayscale. This table is continued in Table 3.

#	Depth	Avg. Width	# Parameters		Average Score ± Standard Deviation		
			(F)MNIST	CIFAR	MNIST	FMNIST	CIFAR
26	16	35	42200	159100	0.943 ± 0.004	0.838 ± 0.004	0.343 ± 0.021
27	16	22.5	198800	656400	0.963 ± 0.003	0.845 ± 0.01	0.37 ± 0.016
28	20	25	94900	323700	0.955 ± 0.002	0.843 ± 0.006	0.367 ± 0.006
29	20	50	60416	62340	0.951 ± 0.003	0.837 ± 0.005	0.163 ± 0.058
30	20	37.5	44700	156600	0.948 ± 0.003	0.834 ± 0.008	0.346 ± 0.028
31	23	31.30	194550	598200	0.927 ± 0.005	0.788 ± 0.008	0.17 ± 0.004
32	25	15	64050	48180	0.951 ± 0.002	0.84 ± 0.004	0.186 ± 0.071
33	25	75	55160	125880	0.899 ± 0.014	0.748 ± 0.033	0.274 ± 0.048
34	25	150	53760	64390	0.782 ± 0.077	0.676 ± 0.064	0.206 ± 0.041
35	28	35.71	74715	78300	0.953 ± 0.001	0.844 ± 0.001	0.244 ± 0.075
36	30	15	60860	152380	0.819 ± 0.08	0.719 ± 0.033	0.17 ± 0.02
37	30	30	18630	145280	0.862 ± 0.08	0.772 ± 0.017	0.168 ± 0.02
38	30	100	34360	146680	0.941 ± 0.003	0.826 ± 0.009	0.165 ± 0.022
39	30	26.67	659100	118560	0.932 ± 0.014	0.785 ± 0.011	0.175 ± 0.007
40	30	31.67	18435	52755	0.313 ± 0.131	0.349 ± 0.109	0.158 ± 0.026
41	35	40	86160	276600	0.753 ± 0.074	0.586 ± 0.11	0.148 ± 0.029
42	35	75	250800	450525	0.725 ± 0.163	0.608 ± 0.077	0.165 ± 0.007
43	40	50	137200	251600	0.522 ± 0.141	0.513 ± 0.089	0.167 ± 0.007
44	40	75	278925	422400	0.467 ± 0.123	0.466 ± 0.09	0.161 ± 0.022
45	50	50	162200	177680	0.242 ± 0.064	0.22 ± 0.042	0.161 ± 0.019

Table 3: Continuation of Table 2 for networks 26 through 45.

#	Hidden Layer Widths
1	50, 50
2	85, 85
3	200, 200
4	20, 30
5	100, 150
6	25, 25, 25
7	50, 50, 50
8	100, 100, 100
9	64, 75, 64
10	75, 50, 25
11	40, 40, 20, 20
12	50, 100, 100, 50
13	15, 15, 15, 30, 30
14	80, 70, 60, 50, 40, 30
15	25, 50, 75, 100, 125, 150
16	10, 10, 10, 10, 10, 10, 10, 10, 10, 10
17	100, 100, 100, 100, 100, 100, 100, 100, 100, 100
18	200, 200, 200, 200, 200, 200, 200, 200, 200, 200
19	20, 20, 20, 20, 20, 15, 15, 15, 15, 15
20	55, 30, 30, 30, 30, 30, 30, 30, 30, 30, 55
21	40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30
22	24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60
23	30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30
24	50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50
25	75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75

Table 4: Ordered list of hidden layer widths for the first 25 networks used in Figure 1. This table is continued in Table 5.

#	Hidden Layer Widths
26	50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20
27	15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
28	25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25
29	50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50
30	45, 45, 45, 45, 45, 40, 40, 40, 40, 40, 35, 35, 35, 35, 35, 30, 30, 30, 30, 30
31	40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20
32	15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15
33	75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75
34	150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150
35	25, 25, 25, 25, 50, 50, 50, 50, 25, 25, 25, 25, 50, 50, 50, 50, 25, 25, 25, 25, 50, 50, 50, 50, 25, 25, 25, 25
36	15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15
37	30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30
38	100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100
39	40, 40, 40, 40, 40, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 40, 40, 40, 40, 40
40	40, 40, 40, 40, 40, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30
41	40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40
42	75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75
43	50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50
44	75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75, 75
45	50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50

Table 5: Continuation of Table 4 for networks 26 through 45.