Examiner's commentary

This is a particularly well-written and well-researched essay. The clear writing style together with an appropriate structure allowed a coherent argument to be formed and easily followed. The introductory sections provided an overview of the topic and reasons for the choice of focus and for the methodology to be used. A notable feature of this essay was the depth of analysis shown both in the explanation of how the parameters were set up for the experiment and particularly in the analysis of the results. The fact that these results did not follow the student's expectations forced them into a deeper exploration of the structural reasons for the results which in turn showed clear evidence of the student's understanding of the relevant computer science. This analysis was also critical with a clear admission of the limitations of the research including the choice of data set as being perhaps too simplistic.

Measuring the Effect of Artificial Neural Network Depth on Test Accuracy in the MNIST Digit Classification Problem

How does the depth of a neural network impact its test accuracy in the MNIST digit classification problem?

Computer Science

Word Count: 3990

Table of Contents

1 Introduction

As computers have become more powerful, they can analyse ever-larger datasets. Entities such as Google, governments, and the Internet of Things are creating larger and more complex datasets with higher volume, velocity, and variety—dubbed "big data". [1] This big data is analysed in a variety of ways. One common way is supervised machine learning, [2] which builds predictive models for a target variable using the remaining data features/variables by iteratively fitting a function to the data. Within supervised machine learning, there exist many algorithms such as support vector machines, decision trees, and deep learning (deep neural networks). The last decade has seen deep neural networks—artificial neural networks consisting of multiple hidden layers and a multitude of neurons—as the forerunner in supervised machine learning as they show consistent improvement in performance and superior generalisability on large datasets whereas other models plateau in performance. [3] However, deep learning is considered a black box; it is unclear why it is successful. [4] Since depth is the key differentiator between neural networks and other models, it becomes essential to answer why and to what extent increasing depth increases neural network performance. Understanding the relationship between depth and accuracy is key to building more accurate networks that can be trained faster. Thus, the research question *"How does the depth of a neural network impact its test accuracy in the MNIST digit classification problem? What is the optimal depth for this particular problem?"* was investigated using the MNIST dataset [5] to gain insight into the broader pattern of the effect of network depth on test accuracy.

1.1 The MNIST dataset

The MNIST database is an example of big data. It consists of 28x28 greyscale images of handwritten digits with a training set of 60,000 images and a test set of 10,000 images. It is often used to test learning techniques on real-world data as it has been cleaned and preprocessed.

In both sets, each image is represented as a row with 785 columns: 784 greyscale (0 to 255) values, each corresponding to a pixel, and a column with the ground-truth value ie. the correct classification of the digit. Thus, each image can approximately be represented with the table below:

Pixel 0	**Pixel 1**	**Pixel 2**	**...**	**Pixel 783**	**Digit**
0	128	255	...	0	2

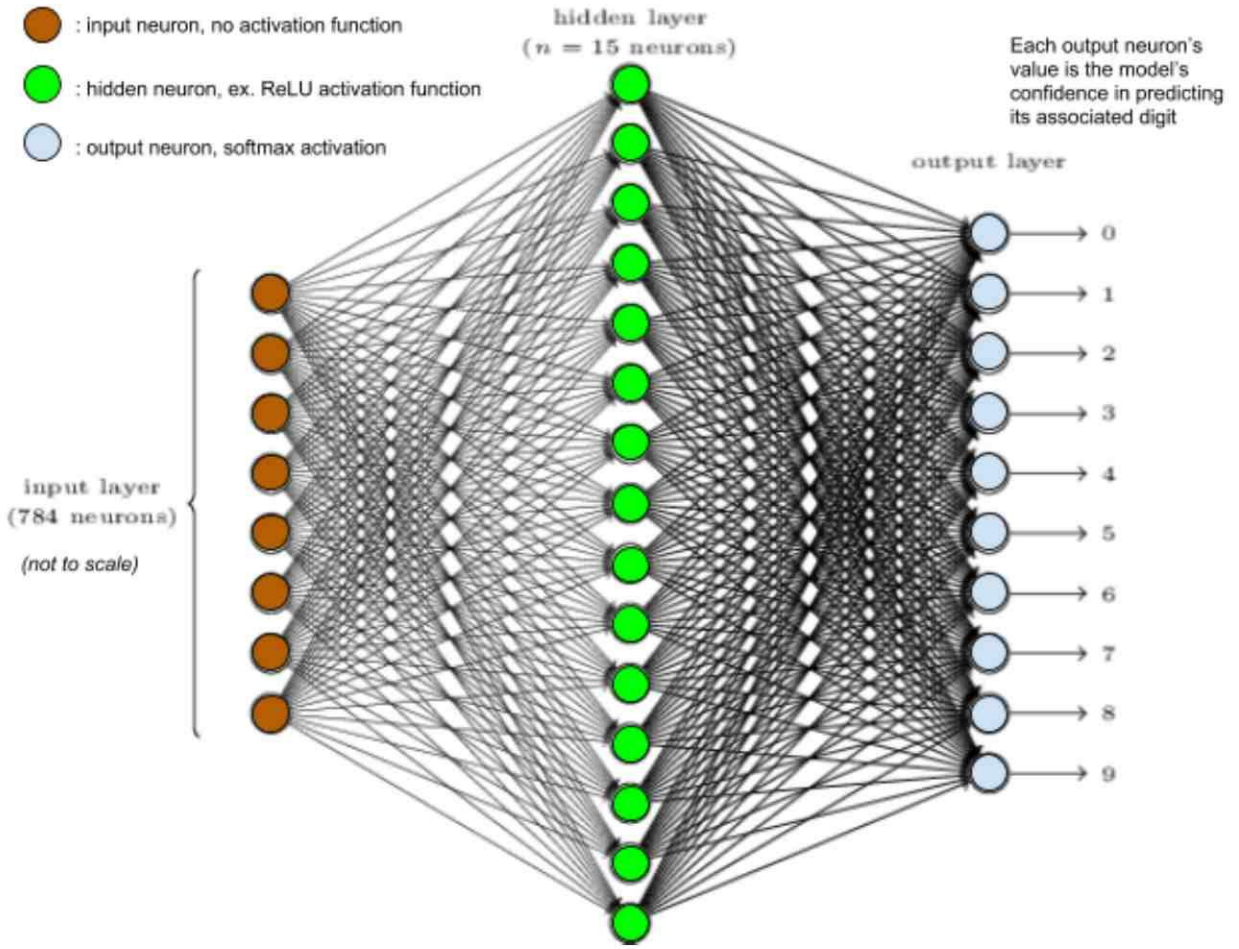
In the MNIST digit classification problem using this dataset, supervised machine learning algorithms such as deep neural networks fit models using the 784 greyscale pixel values to predict the digit they represent. [5]

1.2 Artificial Neural Networks

Artificial neural networks are a machine learning model inspired by biological nervous circuits. They consist of layers of neurons feeding information forward in sequence in an algorithm called

forward propagation. Each *neuron* is a numerical value, determined as the output of an *activation function* (any differentiable function) passed an unscaled number.

Figure 1: Example Neural Network for MNIST Dataset Classification [6]



The example network above for the MNIST dataset contains three layers. The input layer—the first layer—consists of 784 neurons (represented in orange), each without an activation function, as there are 784 features (columns) in the dataset. Each input neuron holds the greyscale value of a pixel. The output layer contains 10 neurons (represented in blue) with a softmax activation

function as the MNIST classification problem has 10 classes (digits). The softmax function returns the index of the output neuron with the highest value, which is equal to the digit it corresponds to, as networks are trained to predict higher values on digits they are confident the greyscale values match.

Between the two are "hidden layers" of neurons (represented in green). The number of hidden layers is the *depth* of the neural network. The above example has a depth of 1. In every hidden layer, each neuron is connected to each neuron in the layer preceding and succeeding it with a *weight* parameter, each weight represented as a line between two neurons. Each neuron also has a *bias* parameter associated with it. [7] *Forward propagation* is defined such that the value of any neuron is the weighted sum of the preceding neurons' values and the bias passed into an activation function. [7] In the above network, this is visualised as the lines between neurons correspond to weighted values of the previous layer's neurons all being inputted into a neuron in the next layer.

After forward propagation, the network's predictions are tested against the ground-truth classification and the error is quantified in a *cost function*—a continuous and differentiable function of the error of classification/regression for a given set of weight and bias parameters. Supervised machine learning uses an algorithm called *gradient descent*—iteratively adjusting parameters by computing and adding the downward gradient of the cost function for the current set of parameters, moving from a high point on the cost function (high error) to a lower point thereon (lower error)—for training/learning.

Neural networks learn through a specific variant of gradient descent known as *backpropagation*, which involves computing layer gradients from the gradients of the forward layer, going backward from the output layer. The *learning rate* is a scalar multiplier controlling the rate of gradient descent. *Gradients* are calculated as the product of the learning rate and the partial derivative of the cost function with respect to a weight/bias at the point defined by the current weights and biases. Through backpropagation, these gradients are then echoed backward through the layers of a neural network, adjusting each parameter. Neural networks commonly experience *vanishing gradients* (tending to 0) or *exploding gradients* (tending to infinity); an annealed learning rate addresses this issue.

Neural networks learn iteratively, performing backpropagation as they iterate over the dataset, each full iteration over the dataset termed an *epoch*. If done optimally, this iterative process balances *bias* (underfitting to a dataset) and *variance* (overfitting to a dataset) to produce networks that generalise well. Regularisation techniques, which punish high weight values by either dropping them or adding them to the cost function, also address overfitting. [10]

Neural networks are configured with *hyperparameters*—values set by the model builder for properties such as as learning rate, activation functions, and depth. Whether a neural network of some depth is considered deep or shallow is dependent on the problem it is solving. [7]

1.3 The Significance of Depth

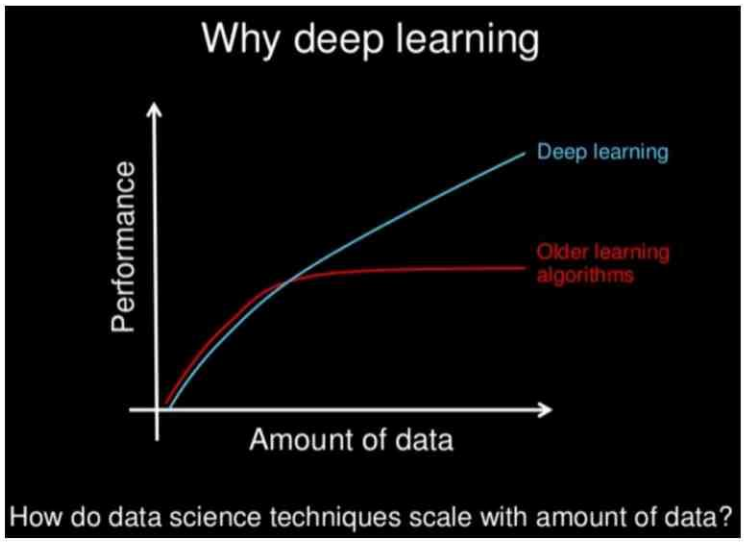
Although neural networks were first explored in the 1950's and 1960's with the invention of the perceptron, they have become popular only in recent years [6].

Starting in 2011, deep neural networks have become commonplace due to their ability to generalise well to unseen datasets. [4] However, it is unclear why deep neural networks outperform other models and shallow networks and generalise better.

In 2015, Andrew Ng, the founder of Google Brain and a leading researcher in deep learning, highlighted that since technology now permits more intensive computation, neural networks have been able to grow both wider and deeper. Unlike other methods of machine learning, their performance continues to increases (rather than plateau) with more data as networks can be made larger to better model big data without overfitting or underfitting. [3]

Figure 2 [3]

Graph showing deep learning's continuous performance increase and other models' plateauing performance with additional data



Research by Kurt Hornik of Technische Universität Wien Austria and Maxwell Stinchcombe and Halbert White of the University of California, San Diego has established that multilayer feedforward networks such as artificial neural networks are universal approximators ie. neural networks can approximate any function to an arbitrary degree of accuracy provided there are enough neurons. Thus, with enough neurons and given enough data, a neural network could perfectly model any function. [8] This could explain why the performance of a deep neural network does not plateau with additional data.

In practice, as universal approximators [8], despite increased numerical instability (vanishing and exploding gradients) and non-robustness [4], deep networks tend to outperform shallow networks as they generalise better. Shallow networks require exponentially more neurons and tend to

memorise. [9] Although it has been argued that deep networks generalise well as they find flat minima on the cost function, they have also been shown to generalise well with sharp minima on a similar reparameterized cost function. [4] Therefore, it appears that the number of hidden layers in a neural network, rather than the parameter space it generates, is responsible for its improved generalisation. Andrew Ng suggests the explanation that deeper networks extract more abstract features (patterns) in data as they learn layer-by-layer. [3]

However, the extent to which his hypothesis is true and remains unclear. This raises the question: what is the effect of network depth on its ability to generalise as reflected in its accuracy on an unseen dataset?

---

2 Research Question

How does the depth of a neural network impact its test accuracy in the MNIST digit classification problem? What is the optimal depth for this particular problem?

---

3 Investigation

In order to investigate the relationship posed by the research question, an experiment was conducted. It was hypothesised that accuracy would increase with additional depth until a certain

depth, after which performance would "degrade" and both train and test accuracy would decrease due early accuracy saturation in initial layers and 'unlearning' from subsequent forward propagation. [11]

3.1 Controls and Method Choices

In order to isolate the effects of neural network depth, other hyperparameters of each network were held (approximately) constant across trials. Controlled hyperparameters included:

1. *Number of neurons (and number of weights):* ~1800 neurons

Variations in the number of neurons in a feedforward network affect its bias and variance and hence accuracy. Fewer neurons increase bias and more neurons increase variance. They also affect a network's train and test error. More neurons allow approximation that is closer to the data as there are more parameters to adjust through gradient descent. Thus networks with more neurons may be more prone to overfitting. [9] Since variations in the number of neurons can have unpredictable effects on train and test accuracy by introducing proneness to over or underfit, in order to isolate the effects of network depth, every neural network trained consisted of approximately 1800 neurons. 1800 was chosen as it is a common multiple for most depths that still yielded a reasonable network size and complexity for the datasets used.

2. *Epochs trained:* One epoch

The number of epochs on which a neural network trains is generally proportional to the network's accuracy to a certain point, after which it overfits. Thus variations in the number of epochs used to train a network affect test accuracy. To improve data precision and accuracy, all networks were trained on one epoch of training data.

3. *Learning rate and Gradient Descent optimiser:* default Adam Optimiser

The learning rate controls the rate of gradient descent and hence how fast a neural network fits data. Since the training is controlled to one epoch, higher learning rates in that time could either lead to faster convergence (higher test accuracy after one epoch) or divergence (very low test accuracy). Conversely, lower learning rates would lead to slower convergence and a higher accuracy than divergence but lower than faster convergence. [12] To prevent these unpredictable effects on data, the default TensorFlow value (0.01) for the Adam optimiser learning rate (alpha) was used in all trials. [13]

Even with a consistent learning rate, every gradient descent optimiser converges at a different rate. [14] To ensure consistency, the default Adam optimiser was used across all trials as it does not get stuck on saddle points and converges relatively quickly by using a windowed average of gradients, which points each step more directly towards a minimum, scaled to optimise step size. [15]

4. *Regularisation:* Dropout with 0.2 probability

Regularisation reduces overfitting and thus influences convergence and test accuracy. Dropout regularisation [16] was used in every layer of every network with a 0.2 chance of a neuron dropping out to provide consistent regularisation. Dropout was chosen over L_1 or L_2 norm regularisation as it prevents the emergence of specific learning pathways and thus reduces interdependent learning, making each neuron a better learner and allowing better generalisation. [17]

5. *Cost function:* Cross-entropy

The cost function of a neural network affects both the rate of convergence as well as the extent it can converge. Additionally, the extent of convergence during training for a cost function is dependent on the number of hidden layers in a neural network. [18] Thus to isolate the effects of depth, cross-entropy cost was used. Cross-entropy cost is defined as:

$$L(Y) := \frac{1}{m}(-Y\,ln(Y') + (1-Y)\,ln(1-Y'))$$ [19]

Where Y' is the ground truth m x 10 matrix, Y is the is the neural network output layer 10 x 1 matrix, and m is the number of datapoints. This equation was chosen as it accounts for the confidence in all 10 digit predictions, rewarding high confidence in the correct classification and punishing even low confidence in incorrect digit classifications. Furthermore, the product of these matrices collapses to a scalar, thus no additional summation is required, reducing computation time due to NumPy's distributed linear algebra calculations.

6. *Train/test datasets:* TensorFlow/Keras MNIST datasets

All models were trained on the same train set and tested on the same test set to prevent precision losses caused by variations in the distribution, features, and complexity of the train/test sets. The TensorFlow/Keras MNIST [5] TensorFlow datasets were downloaded and used with the call: tf.keras.mnist. [20]

3.2 Method

In order to conduct this experiment, a Python program was written using the TensorFlow-GPU python module (see appendix 1). TensorFlow is an open source software library developed by the Google Brain team for fast numerical computation. It is used often in the ML community for its Deep Learning support. [21] In this experiment, TensorFlow's Keras sequential model was used.

For each trial, the MNIST digit dataset was first loaded into TensorFlow and then split into a train and test set. A 1800 neuron neural network with n hidden layers was then defined using a flattened keras sequential model. As commanded by the dataset, the neural network had 784 input neurons, one per pixel in each 28x28 image, and 10 output neurons, each corresponding to a digit 0-9. With the exception of $n = 0$, which had no additional neurons, each hidden layer had $\lfloor \frac{1800}{n} \rfloor$ neurons, so that all networks had exactly/approximately 1800 neurons. All hidden neurons had the same Rectified Linear Unit (ReLU) activation function, a popular function as it

does not experience gradients approaching 0 at high neuron values, and dropout regularisation. [16] The output neurons used a softmax activation so as to return the digit predicted. The model was then compiled with an Adam optimiser, a sparse categorical cross-entropy cost function, and accuracy as a comparison metric. [20]

After compilation, the model was trained on the train set. Train cross-entropy cost and train accuracy (both from the history object returned by TensorFlow) were then recorded in the program memory with corresponding variables. Next, the model was tested on the test set. Then, as with the train figures, test cross-entropy loss and test accuracy were recorded as variables.

Finally, at the end of each trial, data recorded in variables was appended to a CSV file containing the results of all trials. The console log was then appended to a text file containing previous console logs.

Five trials were conducted for various depths: 0 to 15, 20, 25, 50, 75, 100, using the following hardware:

OS: Windows 10

Python version: Python 3.6.5

GPU: GeForce GTX 1070ti

CPU: AMD Ryzen 2700x @ 4.05 GHz

RAM: 32GB DDR4 RAM

Memory: 1TB SSD, 2TB HDD

Motherboard: X470 motherboard

3.3 Raw Data

The loss, accuracy, and time data recorded from the trials can be seen in tables 1 and 2 below.

Table 1: Raw data for depths 0-10

Depth	Trial	Train Cross-Entropy Loss	Train Accuracy	Test Cross-Entropy Loss	Test Accuracy	Train Time	Test Time
	1	0.4761621626	0.87383	0.3077758218	0.9151	4.7608606815	0.5097680092
	2	0.4698261185	0.87703	0.3072223457	0.9139	4.5579767227	0.4596931934
	3	0.4671640720	0.87852	0.3045051179	0.9161	4.6409807205	0.4847297668
	4	0.4690858537	0.87815	0.3065221800	0.9137	4.4757378101	0.4536838531
0	5	0.4705009760	0.87765	0.3073582830	0.9157	4.8082456589	0.4937450886
	1	0.1943779687	0.94195	0.0888630210	0.9733	8.0430274010	0.5628485680
	2	0.1923766260	0.94158	0.1047269804	0.9668	7.9481167793	0.5958974361
	3	0.1916163096	0.94170	0.0971289814	0.9707	8.1733152866	0.5938947201
	4	0.1939208512	0.94152	0.0916829982	0.9704	8.0645678043	0.5818772316
1	5	0.1913587008	0.94192	0.0974784707	0.9707	8.2522304058	0.5948970318
	1	0.2009798698	0.93983	0.1071780019	0.9653	8.8658559322	0.5878858566
	2	0.2013221687	0.93845	0.0989170284	0.9708	9.1037683487	0.6149260998
	3	0.2003030914	0.93917	0.0982761272	0.9690	9.2583317757	0.6740159988
	4	0.2001964616	0.93967	0.1042706804	0.9694	9.0129992962	0.6009051800
2	5	0.2058711999	0.93670	0.1116843889	0.9640	9.1477451324	0.6629989147
	1	0.2315318815	0.92987	0.1003622158	0.9694	9.1305851936	0.5898885727
	2	0.2348403364	0.92862	0.1145069750	0.9653	9.1887254715	0.6810257435
	3	0.2332991042	0.92855	0.1345789077	0.9594	9.2659249306	0.6009049416
	4	0.2290608674	0.93048	0.1302089310	0.9616	9.3009722233	0.6139242649
3	5	0.2268666457	0.93115	0.1308576213	0.9607	9.4311227798	0.6419668198
	1	0.2637906086	0.92157	0.1399214161	0.9609	9.8646242619	0.6089186668
	2	0.2697438648	0.91858	0.1308829525	0.9618	9.3781325817	0.6159257889
	3	0.2671167338	0.91997	0.1240778937	0.9622	9.8357996941	0.7311024666
	4	0.2663127440	0.92025	0.1274340543	0.9608	9.9295210838	0.6289470196
4	5	0.2678683809	0.92022	0.1214606874	0.9656	10.0441343784	0.6680061817
	1	0.2992226158	0.91165	0.1408742020	0.9593	10.6816542149	0.7861855030
	2	0.3007748898	0.91048	0.1656398301	0.9534	11.1653239727	0.7321028709
	3	0.2979290443	0.91292	0.1353245005	0.9599	10.9324369431	0.6449723244
	4	0.3035137490	0.90972	0.1248760151	0.9625	10.6798722744	0.6930444241
5	5	0.3026451615	0.91123	0.1711203099	0.9527	11.0836787224	0.8162305355
	1	0.3375270786	0.90202	0.1693929087	0.9530	11.9176404476	0.6800251007
	2	0.3402227739	0.90055	0.1400181713	0.9615	11.7435491085	0.7611465454
	3	0.3447742789	0.90005	0.1340402907	0.9658	12.0719530582	0.7290978432
	4	0.3336121716	0.90432	0.2108535775	0.9383	11.8029258251	0.6780223846
6	5	0.3501127989	0.89868	0.1482577787	0.9592	11.9800446033	0.6569895744
	1	0.4068281010	0.87877	0.1571252103	0.9596	12.5750021935	0.7791743279
	2	0.3819753614	0.88865	0.1752912307	0.9549	12.7997982502	0.8432707787
	3	0.3833223607	0.88878	0.1640285786	0.9538	12.7684643269	0.7060637474
	4	0.3823063167	0.88848	0.1499018869	0.9617	12.4236562252	0.7381124496
7	5	0.3828229920	0.88970	0.1860997357	0.9539	12.6590495110	0.8192346096
	1	0.4433936317	0.87203	0.1819863410	0.9541	14.0223655701	0.8072164059
	2	0.4400050123	0.87012	0.2083913109	0.9521	13.7557134628	0.7461247444
	3	0.4526862318	0.86702	0.1808164338	0.9560	13.5823709965	0.7080667019
	4	0.4332450176	0.87328	0.1998844436	0.9525	13.7165598869	0.8022091389
8	5	0.4478791666	0.86885	0.1643484224	0.9550	13.8418254852	0.7541370392
	1	0.5231933431	0.83853	0.1772508284	0.9517	15.0216343403	0.8362610340
	2	0.5286184588	0.83948	0.2054764325	0.9492	14.8550405502	1.0015099049
	3	0.5192941734	0.84267	0.1869598641	0.9518	15.0436542034	0.8592948914
	4	0.5337465577	0.83987	0.1980161009	0.9481	15.0548329353	0.8332560062
9	5	0.5198361529	0.84522	0.2048674061	0.9476	14.9480202198	0.7641513348
	1	0.6563744600	0.78487	0.2267415938	0.9469	15.8656299114	0.8262457848
	2	0.6340705788	0.79403	0.2648368831	0.9398	16.0108025074	0.8092193604
	3	0.6122532713	0.81195	0.2172055275	0.9463	15.7377030849	0.8002059460
	4	0.5978583149	0.81773	0.2071263748	0.9462	15.8338210583	0.9484293461
10	5	0.6500325544	0.79120	0.2397010079	0.9338	16.0421376228	0.8112220764

Table 2: Raw data for depths 10-100

Depth	Trial	Train Cross-Entropy Loss	Train Accuracy	Test Cross-Entropy Loss	Test Accuracy	Train Time	Test Time
	1	0.7142091863	0.77003	0.2480565375	0.9401	17.2996747494	0.8422694206
	2	0.7367923003	0.75378	0.2640802324	0.9361	17.4654357433	0.8733155727
	3	0.8022513462	0.71413	0.2659610394	0.9350	17.3631610870	0.8302485943
	4	0.7232665794	0.76623	0.2390593829	0.9398	17.2319509983	0.9273977280
11	5	0.8091586837	0.70435	0.3044690680	0.9155	17.3791537285	0.9233915806
	1	0.9351260629	0.63382	0.4447967700	0.8403	18.3616535664	0.8262460232
	2	0.7940604867	0.73698	0.3519050666	0.9210	18.2121853828	0.9434213638
	3	0.8374289096	0.71178	0.2917675954	0.9321	18.1142337322	0.8953492641
	4	0.8279364331	0.72147	0.2627580354	0.9370	18.1253108978	1.0465781689
12	5	0.9761598248	0.61800	0.4038411881	0.8632	18.7331497669	0.9354093075
	1	0.9674865948	0.61343	0.5033045169	0.7703	19.3536708355	1.0706138611
	2	1.0183979076	0.60248	0.4298628436	0.8768	19.4573175907	1.0005073547
	3	1.0321251795	0.60100	0.5325946084	0.8185	18.9569797516	0.9223902225
	4	1.0521190786	0.59095	0.5141264089	0.8411	19.2089436054	0.9013571739
13	5	1.1108988741	0.56363	0.5437128160	0.8067	19.5594577789	0.8793241978
	1	1.0881610775	0.58277	0.5018508643	0.8246	20.6444578171	0.9424192905
	2	1.1588585998	0.57170	0.5005671941	0.8472	20.9453599453	0.9484307766
	3	1.0995121144	0.60347	0.4202383036	0.8897	20.3382055759	0.8773214817
	4	0.9848551562	0.63357	0.4441310290	0.8555	20.1404712200	1.0385665894
14	5	1.2150164828	0.48113	0.7875393070	0.6779	20.5979466438	0.9464261532
	1	1.4786887091	0.39595	0.8483160892	0.6665	21.4142525196	0.9484291077
	2	1.2751463499	0.48487	0.6626521770	0.7529	21.5124135017	0.9694607258
	3	1.1784563767	0.53365	0.5519347743	0.8112	21.7416028976	1.0906434059
	4	1.2336262383	0.51022	0.6103326198	0.7871	21.1658918858	1.0365622044
15	5	1.3656753891	0.41440	0.7867620429	0.7032	22.2593209743	0.9203870296
	1	1.4926853513	0.34880	1.1679810839	0.4840	27.3915195465	1.3119804859
	2	1.5590110199	0.35637	1.3394787617	0.3540	27.3366482258	1.0545887947
	3	1.5891498513	0.32928	1.2762476919	0.4590	27.2869524956	1.0615987778
	4	1.5825883313	0.33440	1.5216624249	0.3763	26.9204907417	1.3019621372
20	5	1.5389533828	0.32475	1.1904166975	0.4581	27.5715913773	1.2498838902
	1	1.6517305573	0.31585	1.6548585136	0.2716	33.1309075356	1.2939496040
	2	2.0474747661	0.19950	1.9707054794	0.2098	32.3549604416	1.2088217735
	3	1.6412101247	0.35073	1.5744331915	0.2901	33.2229862213	1.6114280224
	4	1.6898477771	0.30780	1.6917536232	0.2306	32.2956786156	1.2138276100
25	5	1.6238397265	0.30723	1.4481904781	0.3386	32.8607902527	1.2188365459
	1	2.3016995107	0.11207	2.3013154472	0.1135	60.4918143749	2.2363696098
	2	2.3015930289	0.11088	2.3013612835	0.1135	60.9038238525	2.1532454491
	3	2.3017557542	0.11118	2.3012206108	0.1135	60.4975032806	2.0991632938
	4	2.3016863631	0.11267	2.3016345276	0.1135	61.0069224834	1.9529426098
50	5	2.3017739380	0.11180	2.3011729546	0.1135	60.8205385208	2.0581014156
	1	2.3016960809	0.10990	2.3009783829	0.1135	86.5189082623	2.9003698826
	2	2.3017117537	0.11150	2.3010672119	0.1135	87.3233568668	2.9384269714
	3	2.3016833822	0.11133	2.3014576237	0.1135	87.7148933411	2.9123883247
	4	2.3016293376	0.11165	2.3011518494	0.1135	86.4512465000	2.7922058105
75	5	2.3016141506	0.11178	2.3012833214	0.1135	87.0125033855	3.0325694084
	1	2.3016867704	0.11108	2.3011141453	0.1135	112.7378304005	3.6124429703
	2	2.3015566792	0.11118	2.3012032707	0.1135	114.6324505806	3.6905617714
	3	2.3017085089	0.11223	2.3010528328	0.1135	112.3642838001	3.7606658936
	4	2.3015608776	0.11127	2.3011943260	0.1135	113.2083222866	3.6975712776
100	5	2.3015601994	0.11025	2.3010756828	0.1135	113.7863795757	3.7065844536

3.4 Processed Data

Table 3: Processed data for all depths

Depth	Average Train Cross-Entropy Loss	Train Cross-Entropy Loss Standard Deviation	Average Train Accuracy	Train Accuracy Standard Deviation	Average Test Cross-Entropy Loss	Test Cross-Entropy Loss Standard Deviation	Average Test Accuracy	Test Accuracy Standard Deviation
0	0.4705478366	0.003020960	0.87704	0.0016773458	0.306676750	0.0011584471	0.9149	0.0009549869
1	0.1927300913	0.001214872	0.94173	0.0001738454	0.095976090	0.0054624313	0.9704	0.0020759576
2	0.2017345583	0.002110273	0.93876	0.0011383419	0.104065245	0.0050557063	0.9677	0.0025938389
3	0.2311197670	0.002866709	0.92973	0.0010231759	0.122102930	0.0128713564	0.9633	0.0036350516
4	0.2669664664	0.001952608	0.92012	0.0009490931	0.128755401	0.0064194119	0.9623	0.0017522557
5	0.3008170921	0.002073584	0.91120	0.0010829999	0.147566972	0.0178378179	0.9576	0.0038427074
6	0.3412498204	0.005730995	0.90112	0.0019198843	0.160512545	0.0278713448	0.9556	0.0095692424
7	0.3874510264	0.009699346	0.88688	0.0040768806	0.166489328	0.0128912898	0.9568	0.0032517072
8	0.4434418120	0.006647375	0.87026	0.0022276944	0.187085390	0.0154921131	0.9539	0.0014732277
9	0.5249377372	0.005514742	0.84115	0.0024539628	0.194514126	0.0109060966	0.9497	0.0017679367
10	0.6301178359	0.022187609	0.79996	0.0126437582	0.231122277	0.0199946939	0.9426	0.0051072497
11	0.7571356192	0.040362035	0.74171	0.0272235654	0.264325252	0.0224379010	0.9333	0.0091220612
12	0.8741423434	0.069314568	0.68441	0.0486958413	0.351013731	0.0676549562	0.8987	0.0393714313
13	1.0362055269	0.046665596	0.59430	0.0169092184	0.504720239	0.0399787731	0.8227	0.0354535414
14	1.1092806861	0.077021041	0.57453	0.0512278197	0.530865340	0.1322138606	0.8190	0.0735751969
15	1.3063186126	0.105678115	0.46782	0.0537413218	0.691999541	0.1100578827	0.7442	0.0531476961
20	1.5524775873	0.034791416	0.33872	0.0119656611	1.299157332	0.1270361518	0.4263	0.0512583808
25	1.7308205903	0.159799132	0.29622	0.0509258062	1.667988257	0.1728806024	0.2681	0.0453246776
50	2.3017017190	0.000063545	0.11172	0.0006341574	2.301340965	0.0001612176	0.1135	0
75	2.3016669410	0.000038284	0.11123	0.0006833740	2.301187678	0.0001682965	0.1135	0
100	2.3016146071	0.000068158	0.11120	0.0006306434	2.301128052	0.0000610638	0.1135	0

3.5 Graphs

Figure 3

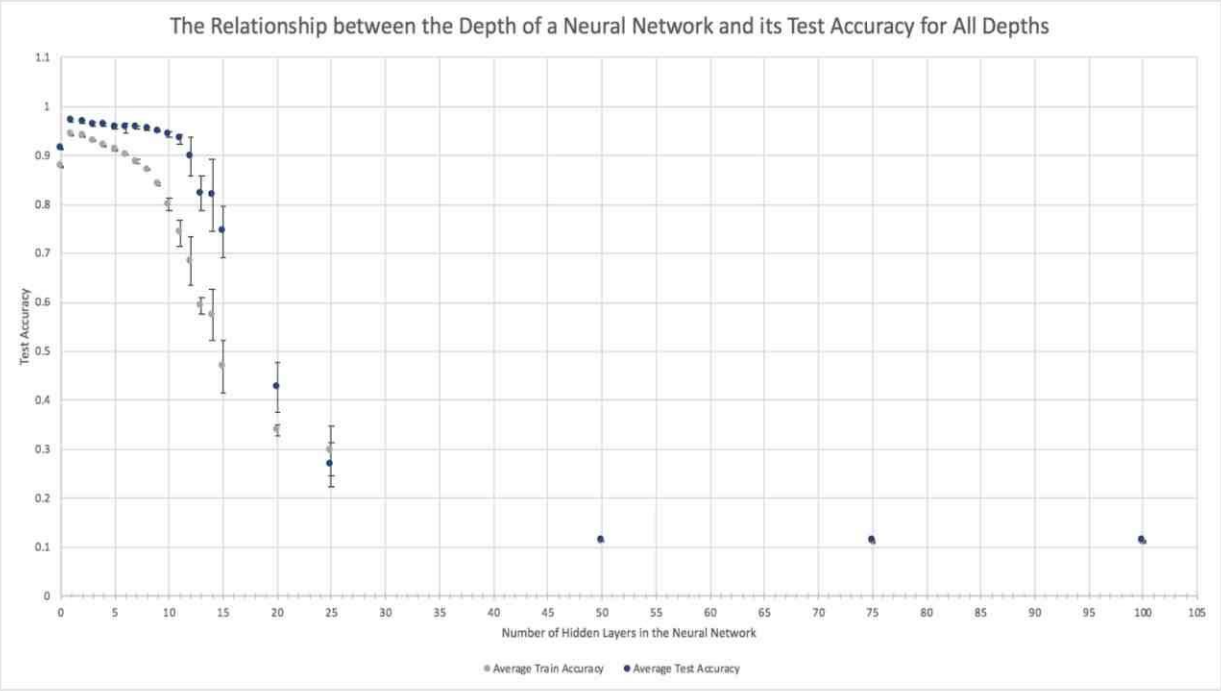
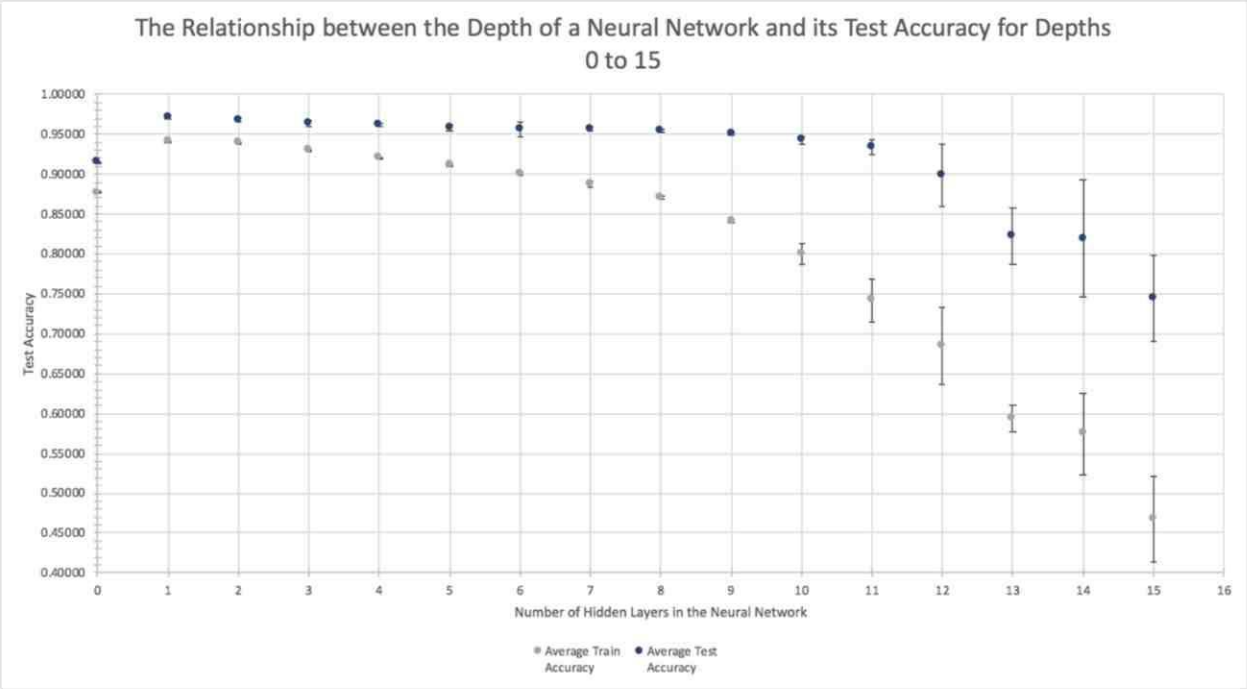


Figure 4



3.6 Trends in Test Accuracy

As predicted, as can be seen in *figures 3* and *4*, test accuracy was maximised at a non-zero value and then started decreasing with the addition of more layers. The trend, however, was not uniform and could not be well modelled with a trendline. Test accuracy was maximised at 1 hidden layer—97.04% accuracy, after which it decreased to approximately 95% until 10 hidden layers, after which it started decreasing dramatically (see *table 3*).

From somewhere between depths 25 and 50 onwards, even after training, the neural network fared no better than random guessing. As in *table 3*, all depths including and greater than 50 hidden layers achieved an average accuracy of 11.35%. Assuming an evenly balanced test set,

guessing accuracy would be an expected 10%—the same value as randomly picking an integer from 0-9 and predicting that for all images. Since all these networks achieved the same accuracies with no deviation, it is likely the were able to memorise the most common target in the train set. Since the train and test set are from the same distribution, this value was likely the most common test target as well, explaining the slightly higher than expected guess accuracy.

As deeper networks are better at finding hidden features—learning "features at various levels of abstraction"—in data, [9] building a larger feature set from a train set which may not match a validation/test set, it may be expected that deeper neural networks become more prone to overfitting. However, overfitting does not explain the performance loss as training accuracy also falls with depth.

On the other hand, decreased accuracy at higher depths can be explained by vanishing or exploding gradients. In both cases, the deep network cannot converge on the global minimum. Vanishing gradients lead to being stuck, and exploding gradients cause divergence. In either scenario, test accuracy drops substantially. These occur more severely in deep networks than shallow ones, as small or large gradients carry multiplicatively through all layers by backpropagation. [22] As a result, since the learning rate was fixed (and thus unable to raise vanishing gradients or lower exploding gradients to useful values), their occurrence could explain decreased performance with depth and the equivalent-to-guessing accuracy of the deepest networks tested.

Alternatively, decreased accuracy with increased depth could be caused by the degradation of the neural networks due to the low complexity of the MNIST handwritten digit dataset. As its creators suggest, it has become too simple for most new deep learning methods. [23] High accuracy (91.49%) with logistic regression (no hidden layers and fewer neurons) supports this as few parameters are needed to approximate a function to classify digits. An optimal depth (number of hidden layers with maximum test accuracy) with one hidden layer further supports this, as deeper networks find greater complexity in datasets—one hidden layer may find edges or loops in numbers and that information on—which is more effective for more complex datasets.

Low complexity could hence explain the success of shallow networks over deeper ones as deep networks "degrade". Degradation occurs when initial layers (the first layer in this case) learn enough from the train set to generalise well, and accuracy becomes saturated. Later layers are identity maps of this first layer and are copied from this layer. However, additional layers do not necessarily fit the layer of saturated accuracy, [11] and hence the network 'forgets' or 'misfits' what it learns in earlier layers, causing the output layer to have significantly lower accuracy.

Accuracy saturating at one hidden layer explains why it is the optimal depth. The degradation of accuracy is dependent on how well later layers fit previous, better saturated layers. [11] The trend effects of this degradation seem to increase with depth, as seen in *figures 3* and *4*, as accuracy falls more dramatically with increasing depth.

Another possibility is that networks were bottlenecked, becoming too narrow as they become deeper since the number of neurons is approximately fixed. Hidden layers with too few neurons have fewer weights and nodes to relay information to the next layer. Hence the 18 neurons (depth 100) in the first layer cannot capture all complexities the in training data and hence pass on incomplete information to the next layer, which faces the same issue. Thus the fit becomes progressively worse through layers until perhaps the only information left is the most common number. Bottlenecking could hence explain why test accuracy decreases with additional depth after the optimal point. Since the number of neurons per layer is proportional to $\frac{1}{n}$ where n is the number of hidden layers, increased bottlenecking could also explain why test accuracy decreases more significantly at higher depths.

3.7 Evaluation of Investigation

Despite producing coherent results, this investigation had a variety of shortfalls that should be discussed.

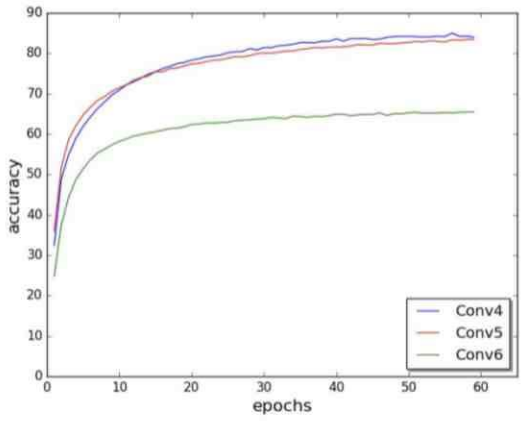
In order to isolate the effects of network depth, a decision was made to keep all other hyperparameters constant (see section 2.1). However, at varying depths, choices in other hyperparameters influence test accuracy. For instance, since wider networks are better at memorisation [9], using 18000 neurons could have resulted in worse test accuracy in shallower networks. The optimal depth of 1 hidden layer may not hold true for 18000 neurons as a network wide may simply memorise training data, increasing overfitting and lowering test accuracy.

Conversely, an 18 neuron network with one hidden layer would have likely performed worse than logistic regression due to bottlenecking.

Similarly with the number of epochs trained, as shown below, different network depths and parameter initialisations impact its rate of convergence.

Figure 5 [24]

Graphs of the relationship of test accuracy and train epochs on three different convolutional neural networks



Thus, it is possible that deeper networks could achieve higher accuracy than shallower networks given more epochs to train as shallower networks are better at memorisation, whereas deeper networks can find more patterns in data, [9] which requires more training epochs.

Moreover, as described in the analysis (section 3.6), a fixed learning rate made deep networks more susceptible to vanishing or exploding gradients. [22] A variable learning rate would therefore have likely had some effect on the data, potentially raising the test accuracy of deeper networks.

Non-depth hyperparameters were decided somewhat arbitrarily, ie. without exploring their effects, and they have varying influence on test accuracy based on the network depth. In other words, the optimum accuracy of any network depth is dependent on the combination of hyperparameters rather than simply depth. As a result, the scope of investigation narrows to exploring a smaller region of hyperparameter space for MNIST digit classification, and thus all results are relevant to this particular combination of hyperparameters, where only depth is variable. Exploring variations of other parameters as well as depth would broaden the scope of investigation by exploring more dimensions of hyperparameter space. Comparing the test accuracy of various depths, each with an optimal combination of other hyperparameters, would allow the results to generalise more effectively.

This investigation was conducted with the hardware described in the method (see section 2.2). Most industrial applications require far more powerful machines to train deep neural networks in a reasonable time, particularly for more complex tasks with larger datasets. Such setups include multiple, more powerful GPU's and CPU's, more memory and RAM, and other specialised hardware. [25] Considering the low complexity [23] and small-size of the MNIST handwritten dataset, it is unclear how significant an impact industrial hardware would have had on test

accuracy. However, the results of this investigation could be more relevant and useful by studying network architectures with appropriate hardware.

4 Conclusion

This paper investigated the effects of a neural network's depth, measured as its number of hidden layers, on its test accuracy on the MNIST handwritten digit dataset. Five neural networks of various depths were built, trained, and tested with a Python script using the TensorFlow-gpu Keras sequential implementation of neural networks.

The results of the investigation, seen in *figures 3* and *4*, demonstrated that test accuracy was maximised with one hidden layer at 97.04%, hovered around 95% until 10 hidden layers, after which it decreased dramatically until 25-50 hidden layers, after which it was equivalent to expected guessing accuracy. These patterns in test accuracy could be explained by low data complexity, which led to degradation of identity maps propagating forward through the network [11], by vanishing or exploding gradients hampering convergence in deeper networks [22], or by bottlenecking caused by decreasing width due to a fixed number of neurons.

These results are, however, limited to a particular region of hyperparameter space other hyperparameters of the neural networks were controlled. Hence, while the results are useful for this region, more research is necessary to see the effects of depth across hyperparameter space ie, with different network hyperparameter configurations. Similarly, further clarity can be gained on

the reason for particular optimum depths and deep networks' performance degradation through the same investigation of depth on other, more complex datasets.

5 References

[1] Oracle Big Data. Slowly Changing Dimensions. Oracle.

https://www.oracle.com/big-data/guide/what-is-big-data.html

[2] What is big data analytics? - Definition from WhatIs.com. SearchBusinessAnalytics.

https://searchbusinessanalytics.techtarget.com/definition/big-data-analytics

[3] What is Deep Learning? (2016, September 22). Machine Learning Mastery.

https://machinelearningmastery.com/what-is-deep-learning/

[4] ShChadei, D. S. (2017, December 20). Modern Theory of Deep Learning: Why Does It Work so Well. Medium.com. Medium.

https://medium.com/mlreview/modern-theory-of-deep-learning-why-does-it-works-so-well-9ee1f7fb2808

[5] THE MNIST DATABASE. MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges. http://yann.lecun.com/exdb/mnist/

[6] Nielsen, & A., M. (1970, January 1). Neural Networks and Deep Learning. Neural networks and deep learning. Determination Press. http://neuralnetworksanddeeplearning.com/chap1.html

[7] Ng, Andrew. (2011, August 15). Week 4: Model representation I. Machine Learning - Stanford University. Coursera. https://www.coursera.org/learn/machine-learning/lecture/ka3jK/model-representation-i.

[8] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), 359–366. doi:10.1016/0893-6080(89)90020-8.

[9] J. O'Brien Antognini (https://stats.stackexchange.com/users/115448/j-obrien-antognini). (2016, July 13). Why are neural networks becoming deeper, but not wider? StackExchange. https://stats.stackexchange.com/q/223637

[10] Ng, Andrew. (2011, August 15). Week 5: Backpropagation algorithm. Machine Learning - Stanford University. Coursera. https://www.coursera.org/learn/machine-learning/lecture/ka3jK/model-representation-i.

[11] He, Zhang, Ren, & Sun. (2015, December 10). Deep Residual Learning for Image Recognition. [1512.03385] Deep Residual Learning for Image Recognition. https://arxiv.org/abs/1512.03385

[12] Zulkifli, H. (2018, January 21). Understanding Learning Rates and How It Improves Performance in Deep Learning. Towards Data Science. Towards Data Science.

https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

[13] tf.train.AdamOptimizer | TensorFlow. TensorFlow. https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer

[14] Sebastian Ruder. (2018, May 31). An overview of gradient descent optimization algorithms. Sebastian Ruder. Sebastian Ruder. http://ruder.io/optimizing-gradient-descent/index.html

[15] Kingma, P., D., Jimmy, & Ba. (2017, January 30). Adam: A Method for Stochastic Optimization. [2412.6980] Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980

[16] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014, January 1). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. http://jmlr.org/papers/v15/srivastava14a.html

[17] Budhiraja, A., & Budhiraja, A. (2016, December 15). Learning Less to Learn Better - Dropout in (Deep) Machine learning. Medium.com. Medium. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5.

[18] Soares, R. G. F., & Pereira, E. J. S. (2016). On the performance of pairings of activation and loss functions in neural networks. 2016 International Joint Conference on Neural Networks (IJCNN). doi:10.1109/ijcnn.2016.7727216. https://ieeexplore.ieee.org/document/7727216

[19] Daniel Godoy. (2018, November 21). Understanding binary cross-entropy / log loss: a visual explanation. Towards Data Science. Towards Data Science. https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.

[20] Get Started with Tensorflow. TensorFlow. https://www.tensorflow.org/tutorials/

[21] TensorFlow. TensorFlow. https://www.tensorflow.org/

[22] Nielsen, & A., M. Neural Networks and Deep Learning. Neural networks and deep learning. Determination Press. http://neuralnetworksanddeeplearning.com/chap5.html

[23] Zalandoresearch. (2018, October 5). zalandoresearch/fashion-mnist. GitHub. https://github.com/zalandoresearch/fashion-mnist

[24] Jawahar, C., Purini, S., & Yasaswi, J. Predicting the Training Time of Deep Neural Networks. IIIT. https://researchweb.iiit.ac.in/~jitendra.katta/papers/predicting_training_time_final.pdf

[25] Sverdlik, Y. (2017, February 6). A Cloud for the Artificial Mind: This Data Center is Designed for Deep Learning. Data Center Knowledge. https://www.datacenterknowledge.com/archives/2017/02/06/this-data-center-is-designed-for-deep-learning

[26] Ren, SJ., J., Xu, & Li. (2015, January 29). On Vectorization of Deep Convolutional Neural Networks for Vision Tasks. [1501.07338] On Vectorization of Deep Convolutional Neural Networks for Vision Tasks. https://arxiv.org/abs/1501.07338

[27] A History of Deep Learning. (2018, September 18). Import.io. http://www.import.io/post/history-of-deep-learning/.

6 Appendix

Appendix 1: Investigation source code

```
import tensorflow as tf
import ssl
import time
import csv
import gc


# prevents ssl certificate error
try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    # Legacy Python that doesn't verify HTTPS certificates by default
    pass

else:
    # Handle target environment that doesn't support HTTPS verification
    ssl._create_default_https_context = _create_unverified_https_context

num_neurons = 1800
depth = [0] # 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,20,25,50,75,100
trial_num = 1
num_trials = 1
depths = []

for i in range(len(depth)):
    for n in range(num_trials):
        depths.append(depth[i])

mnist = tf.keras.datasets.mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0 # normalise the data scale

for i in range(len(depths)):
    # print("Depth: " + str(depths[i]) + ", Trial: " + str((i % num_trials) + 1))
    print("Depth: " + str(depths[i]) + ", Trial: " + str(trial_num))
    model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten()
    ])

    gc.collect()

    for j in range(depths[i]): # add layers
        model.add(tf.keras.layers.Dense(num_neurons // depths[i], activation=tf.nn.relu))
        model.add(tf.keras.layers.Dropout(0.2)) # regularisation

    model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax)) # output layer

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    ct = time.time()
    history = model.fit(x_train, y_train, epochs=1) # train model
    traintime = time.time() - ct

    train_crossentropy = history.history["loss"][0]
    train_accuracy = history.history["acc"][0]

    ct = time.time()
    test_crossentropy = model.evaluate(x_test, y_test)[0]
    test_accuracy = model.evaluate(x_test, y_test)[1]
    testtime = time.time() - ct


# append data to tabular csv file
with open("tensorflow_data_mnist.csv", 'a') as file:
    csvWriter = csv.writer(file, delimiter = ',')
    for i in range(len(depths)):
        csvWriter.writerow([depths[i], train_crossentropy, train_accuracy,
                            test_crossentropy, test_accuracy, traintime, testtime])


# record console logs
with open(r"C:\Users\Abhay\Desktop\EE console full log.txt", 'a') as file:
    with open(r"C:\Users\Abhay\Desktop\EE console logs.txt", 'r') as file1:
        for line in file1:
            file.write(line)
        file.write("\n")
        file.write("-----------------------------")
        file.write("\n")
```

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

Extended essay - Reflections on planning and progress form

Candidate: *This form is to be completed by the candidate during the course and completion of their EE. This document records reflections on your planning and progress, and the nature of your discussions with your supervisor. You must undertake three formal reflection sessions with your supervisor: The first formal reflection session should focus on your initial ideas and how you plan to undertake your research; the interim reflection session is once a significant amount of your research has been completed, and the final session will be in the form of a viva voce once you have completed and handed in your EE. This document acts as a record in supporting the authenticity of your work. The three reflections combined must amount to no more than 500 words.*

The completion of this form is a mandatory requirement of the EE for first assessment May 2018. It must be submitted together with the completed EE for assessment under Criterion E.

Supervisor: *You must have three reflection sessions with each candidate, one early on in the process, an interim meeting and then the final viva voce. Other check-in sessions are permitted but do not need to be recorded on this sheet. After each reflection session candidates must record their reflections and as the supervisor you must sign and date this form.*

First reflection session

Candidate comments:

Though I was initially unsure of what subject to pursue for my EE, after reading some computer science literature, I decided on CompSci. Within CompSci, I knew that I wanted to research within the field of Machine Learning as it has always interested me, and as I have been taking online courses on it for a few months. I am fascinated by Deep Learning, particularly its efficacy on big data and how each layer in a deep neural network finds new 'features'. I thus decided to look into how the depth of a neural network impacts its accuracy on the MNIST handwritten digit dataset.
I have hitherto written the code for the neural network, and it is more-or-less working, but it still has a few bugs. However, the preliminary data collected (1 hidden layer being optimal) suggests that this dataset may not be complex enough to effectively explore. Hence, I may want to train the network using more/different datasets — I have found a few online. My supervisor agreed and also recommended establishing a timeline for my research and writing.

Date: May 29, 2018

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

Interim reflection

Candidate comments:

Currently, I have finished researching external sources for material regarding the theory behind deep learning and its societal impacts and have written a rough outline. I changed my neural network script from my initial manually made network to a keras/tensorflow model after finding it in my research. After collecting preliminary data, I realised that the time taken to train increased after every trial, which should not have been the case. Instead, the times should have been similar for each depth with some random variation. Based on the increased time each trial, the issue must be that memory was not being cleared by tensorflow, slowing the program. Going forward, I need to recollect the data, perhaps on more than one dataset, one trial per run. After that, I need to start writing and fill in any possible holes in my argument. As recommended by my supervisor, I need to emphasise clarity, particularly in my data presentation.

Date: September 25, 2018

Final reflection - Viva voce

Candidate comments:

I enjoyed writing the Extended Essay, and it has deepened my interest in my chosen topic within Computer Science and Machine Learning.
My initial plans were appropriate to investigate and were largely realised. While I was able to find a wealth of research and information, I could not find any previous work that directly explained my data. I learned various skills including finding appropriate research material, synthesising ideas and arguments from related material, and presenting knowledge formally. I also got a deeper understanding of the intricacies of neural networks.
My most significant struggles led to my most rewarding successes—solving methodology issues causing erroneous trends, condensing issues into a problem-scope, and creating a flowing argument including multiple perspectives. If I were to change anything, I would use a slightly broader scope using multiple datasets of varying complexity to find more meaningful trends, and I would be more organised with my time.
I look forward to using skills this essay has developed for academic research in college.

Date: January 31, 2019