REDUCTION OF NEURAL NETWORK COMPLEXITY THROUGH STOCHASTIC NUMBERS

James Ridey

Bachelor of Engineering
Software Engineering

School of Engineering
Macquarie University

June 7th, 2019

Supervisor: Associate Professor Tara Hamilton

ACKNOWLEDGMENTS

I would like to acknowledge and thank, Tara Hamilton and Alan Kan for their support and advice during my thesis.

STATEMENT OF CANDIDATE

I, James Ridey, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the School of Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment at any academic institution.

Student's Name: James Ridey

Student's Signature: James Ridey

Date: 7/6/2019

ABSTRACT

Neural networks considered a core foundation of machine learning, has recently ballooned into a large framework that requires vast amounts of computing resources and time. Here, we examine how to improve these networks, hoping to achieve networks that can calculate the same result with less training time and/or resources using, ideas such as stochastic numbers and probabilistic results.

Contents

List of Figures

Chapter 1

Introduction

1.1 Neural networks: introduction

Neural networks are the core structure of what many believe to be the next step in computing, machine learning. We are starting to see machine learning used to solve problems that originally were only able to be solved by people. However, all this machine learning comes at a cost, machine learning needs vast amounts of data to learn and thus large amounts of computing resources, electricity, time and even the best machine learning models work within a narrow, controlled set of inputs, anything outside this "range" are misclassified, in other words, machine learnings models are sensitive to noise and not general enough.

1.2 Project goal

My thesis project goal is primarily trying to reduce neural network complexity. Complexity that forms due to the vast amount of data that needs to be worked through and the finicky nature of neural networks. Overall I've identified an avenue that I plan to explore in more detail: creating an alternative neural network model using stochastic numbers. With the goal of this model to be more efficient in computational time and/or being able to train faster while still maintaining the same levels of accuracy.

Chapter 2

Background and Related Work

2.1 Neural networks: explanation

Neural networks are a very rough simulation of how the neurons in a human brain actually work [6], they have been researched since the 1960s [7] but at the time we did not have the computing power necessary to run these large neural networks and thus progress was halted until around 1970-1980's when backpropagation was discovered making neural network training much more efficient and parallel computing system was also starting to kick off.

The most common form of a neural network in the feedforward model where the inputs are passed to the neural network and it outputs an answer. The feedforward model consists of an input layer, hidden layers and an output layer. Each of these layers are composed of varying amounts of neurons depending on the complexity of the data that is trying to be learned. More models of neural networks exist which aim to memorise or improve the feedforward model, however, the feedforward model is incredibly ubiquitous and is the main driving force behind machine learning in the industry.

To understand what each layer in a neural network is doing, it is first necessary to understand what a neuron is doing. A neuron is comprised of inputs, a set of weights, one output and an activation function [8], as shown below.

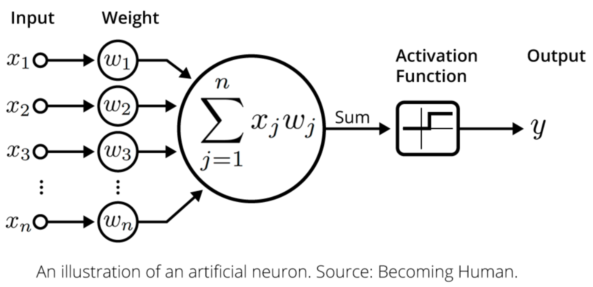

Figure 2.1: Neuron model [1].

Neurons are fed a set of data through its inputs which are multiplied by a specific weight, all of these numbers are then added up and fed to an activation function. An activation function is a function that maps the input to a value between 0 and 1, a commonly used activation function is the sigmoid function, numbers in the positive infinity direction become increasingly closer to 1, while numbers in the negative infinity direction become increasing closer to 0.

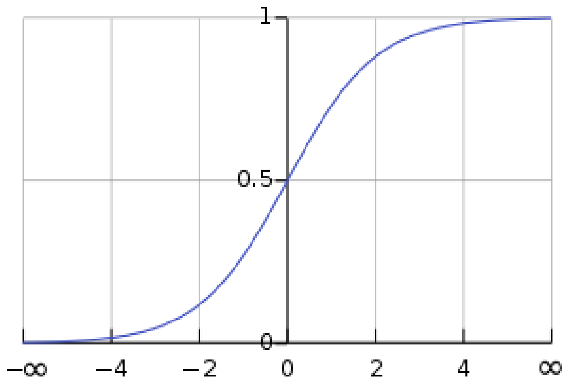

Figure 2.2: Sigmoid graph [2].

The next concept is neural network layers which vary based on the task you want to try an accomplish, some examples include convolutional layers which attempt to use neural networks on squares of pixels, dropout layers which randomly pick weights to drop to try and reduce overfitting and the most common layer used to most machine learning models, is the dense layer. In this layer, every neuron connects to every other neuron in the previous layer.

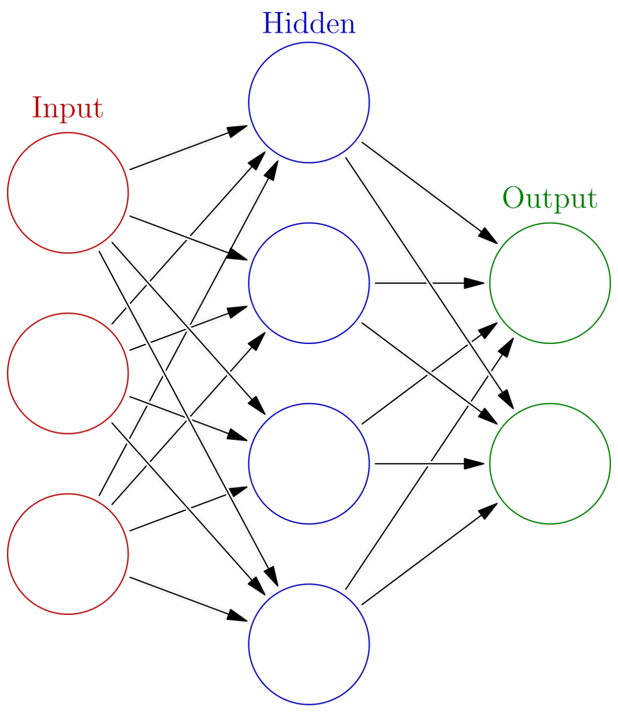

Figure 2.3: Dense layer neural network model [3].

Each circle in this figure represents a neuron and each line in this figure represents a connection between 2 neurons. By manipulating the weights in the neural network, we change what each neuron calculates and outputs as part of its activation function. In doing so, we now have a framework of which for any given input, the neural network can generate any output (given the right weights), regardless of the complexity of the output [9].

2.1.1 Training

The next question about neural networks is then how are the weights calculated for a given input. This is where training a neural network comes into play. The basic summary for how a neural network is trained is they are fed the input data, the error amount is calculated from the difference between the neural network output and the expected output. Then the weights are slowly adjusted to minimize the error rate. How this is all done is where the complexity of neural networks lie and why they take so long to train. One of the simplest algorithms that implements this is the stochastic gradient descent algorithm, which is a core component of many other machine learning algorithms like RMSprop, ADAM and AdaGrad (RMSprop being a commonly used algorithm and the one that was used in the majority of the MNIST tests).

The gradient descent or GD works by measuring the gradient and trying to find a point at which the error is the lowest. The gradient being a measure of "how much the output of a function changes if you change the inputs a little bit" [10]. The best way of explaining GD is with a hill-climbing example, a blind person is trying to find the highest point of a hill and so he starts by taking large steps to estimate where the steepest part of the hill is and then smaller and smaller steps until he finds the top of the hill. Applying this to machine learning, the error of the output of the machine learning model is compared with the actual values that are to be expected and this is the error amount, then the weights are changed are relatively "large" amount and the error is measured again and compared to the previous error, this is the gradient. This gradient is then used to estimate how big of a "step"/how much to adjust the weights by on the next iteration.

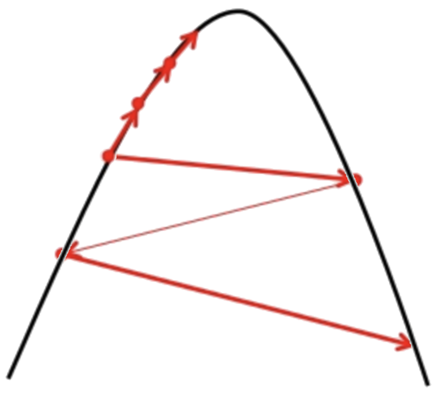

Figure 2.4: Stochastic gradient example. Y-Axis represents accuracy rate

However in realities, there are much more dimensions to a machine learning model and secondly, there is not just one point that is the highest but multiple hills of varying heights, this is one of the many reasons why machine learning is hard. You can create simple machine learning algorithms that find a maximum point but not the global maximum point or the global maxima, you can also try and adjust how much you step by each amount but the best way, which requires all the computation power is a small stepping amount which tries all points on the "hills".

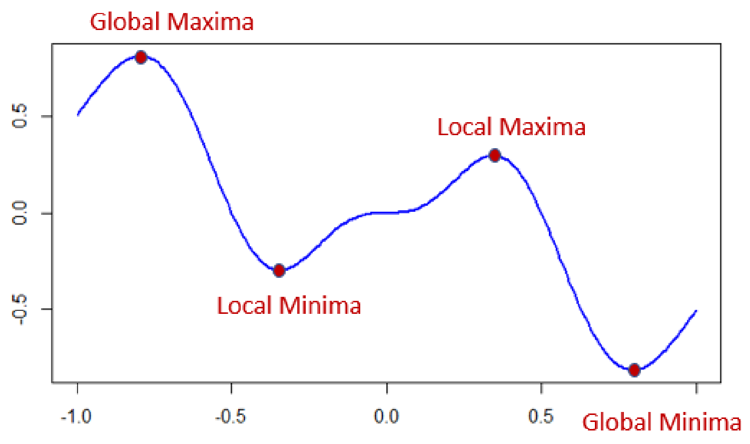

Figure 2.5: Global maxima example. Y-Axis represents accuracy rate

2.2 Complexities of neural networks

The main question about this thesis is what makes training neural networks so computationally complex, why neural networks are trained through large GPU farms and the challenges facing researching trying to shrink the size of these neural networks.

2.2.1 Overfitting and underfitting

Neural networks are a statistics tool with the main purpose of being able to generalize from one set of training data and then predict a value based on a new set of input data.

To train a neural network, we divide the training into 2 phases, the learning/training stage and the prediction state. The learning/training stage is where the weights of the neural network are adjusted using backpropagation or a similar algorithm and the neural network is trying to fit the data as well as it can by correctly predicting the correct value for each given input. Then the prediction stage is where the neural network is evaluated against a subset of the data that it has not seen before, this stage is purely to combat overfitting.

Overfitting is when the neural network begins to fit so well with the training data, that is unable to generalise/achieve good accuracy in the prediction stage. In other words, it fails as a classification or prediction model because it hasn't really learnt the underlying structure of the data but rather is just "cheating" in the training stage.

Conversely, underfitting occurs when the neural network fails to achieve any accuracy in the training or prediction stage, this is quite rare to see in practice but can occur with bad datasets or a too simplistic neural network model [11].

2.2.2 Complexities

Given overfitting and underfitting, it becomes a challenge to train neural networks because having a model that is too simplistic will cause underfitting and having a model that is too large not only increases computation time with all the various amounts of weights that each layer incurs but also leads to overfitting [12].

Not to mention, the training stage usually requires multiple iterations and the back-propagation algorithm has a high time complexity [13]. All in all, this means we end up with a large neural network, with thousands to millions of weights and due to how neural networks work, most of the operations needed are matrix multiplications which is what GPU's tend to be good at, considering computer graphics are just matrix manipulations.

2.3 Research into neural network complexity

There has been a lot of research in neural networks, while much of it is focused upon learning more and more complex datasets [14] [15] [16], there has also been a lot of research into trying to prune neural networks and a large number of parameters (weights) that are needed for these large scale neural networks. A recent example is "Importance Estimation for Neural Network Pruning" [17], where they cite a 40% reduction in complexity by trimming off 30% of the parameters.

In the field, which I'm planning to research there have been a few papers on stochastic numbers and the benefits regarding using them on neural networks, for example, one such paper states a 47% area and 60% power reduction while maintaining a high accuracy rate [18].

Another paper [19] goes on to show how its stochastic neural network allows for building reliable systems due to the high level of noise-immunity that the network model shows.

While both of these papers are impressive research and they show that their networks are comparable with a regular dense layer neural network or MLP (Multi-Layer Perceptron) for short, they fail to address advantages and disadvantages that a stochastic neural network might bring to the table, an example of such an advantage is how a stochastic neural network compares with a regular MLP network when exposed to a noisy dataset. The previous paper regarding noise immunity is mainly focused around random noise injected alongside the input data, not noise inherent within the input data. One of my research avenues is to explore how stochastic neural networks, react to noise during the training and validation stages.

Chapter 3

Research plan

3.1 Stochastic computing

Stochastic computing is a branch of computing that relies on probabilistic bitstreams to calculate numbers, often relying on simple bit-wise operations. Every number in a stochastic computer is arranged as a probability of the ones and zeros in a continuous binary number stream [20].

3.1.1 Advantages of stochastic numbers

There are 2 major advantages with this technique, one is that the system is extremely resistant to noise, flipping multiple bits in the stream will have very little effect or will only change the number minorly. The other major advantage is that certain operations have a simpler solution, for example in terms of multiplication on a conventional computer this would be an n^2 operation, however, on a stochastic computer, this is accomplished with an AND gate for a cost of n [21].

3.1.2 Disadvantages of stochastic numbers

However, the reason for stochastic computing not being proliferated is the inherent weakness which is the randomness of stochastic numbers. Stochastic numbers can never truly represent the number they are trying to calculate and thus need a large amount of sampling to "zone in" on the number being calculated.

Secondly is that while stochastic numbers have a simple analog for multiplication, the core of stochastic numbers like the PRNG (Pseudo-random number generator) for generating the random bitstream and decoding the bitstream by sampling do not have simple gate-level analogues and this is where stochastic numbers lose their advantage.

3.1.3 Reasoning behind stochastic numbers

Stochastic numbers are a relatively old concept dating back to the 1960s but with the introduction of digital binary logic, it quickly faded into obscurity. However recently there has been a resurgence of ideas with stochastic computing that take advantage of its properties and use it to create faster better and more noise resistant systems, with a more recent example being a stochastic CNN (Convolution neural network: Used for image classification, made of convolutional layers) which consumes less energy than that of a normal floating-point CNN [22].

The idea to use stochastic numbers to improve neural networks comes primarily from the idea of brains and neurons as analogue representations, it is well known that neurons communicate not in continuous streams of data but spikes of activity. The idea is further extended to coincide with stochastic bitstreams with the 1's being a spike of activity, also given the noise-resistant properties of stochastic numbers and how neural networks don't require any extreme levels of precision, our hope is to pair stochastic numbers with neural networks to gain benefits through the multiplicative and noise resistant properties.

3.2 Approach

My approach to this research is to create custom Tensorflow layers that will fit within existing neural network models, to keep as much of the variability between models as minimal as possible. Each model will be trained with the same MNIST dataset and the main metrics that I will be measuring in each test is the training accuracy as a percentage and the total amount of time before a model reaches 99% accuracy in the training stage.

3.2.1 Tensorflow

Tensorflow [23] is a machine learning framework primarily written in Python. It was released by Google under the Apache License 2.0 on November 9th, 2015. The Tensorflow network employs the idea of creating a neural network model which is then compiled before being handed off to the appropriate processing unit be it CPU, GPU or TPU, this is in contrast to other machine learning frameworks like PyTorch that generate their models dynamically on the fly which results in a small performance hit.

Tensorflow continues to grow in support with new frameworks like Kera that aim to simplify common operations and overall streamline the process and show no signs of stopping with a stable release only 2 months ago.

3.2.2 MNIST dataset

The MNIST dataset [24] is a commonly used dataset for machine learning that consists of 70,000 images of handwritten digits (60,000 training images and 10,000 test images) and the associated labels for each of these images. The dataset is commonly chosen due to its simplicity and many frameworks support it out of the box.

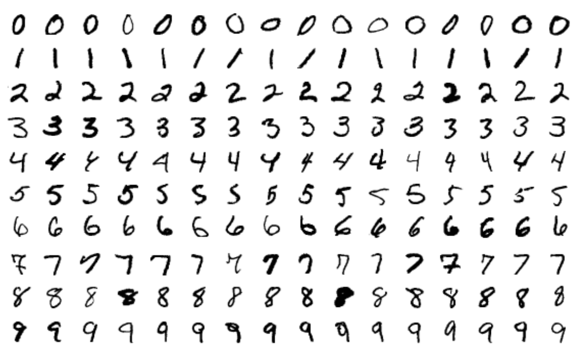
Figure 3.1: Samples of the MNIST dataset [4].

Another common reason that MNIST is chosen is to demonstrate the accuracy of neural networks over more traditional statistical analysis such as linear classifiers and K-nearest neighbours, both of which score around an error rate of 12% and 5% respectively [25] as compared to a simple neural network of 3 layers which can score around 3.05% [25].

However this dataset is not without its faults, several images are dubiously labelled.

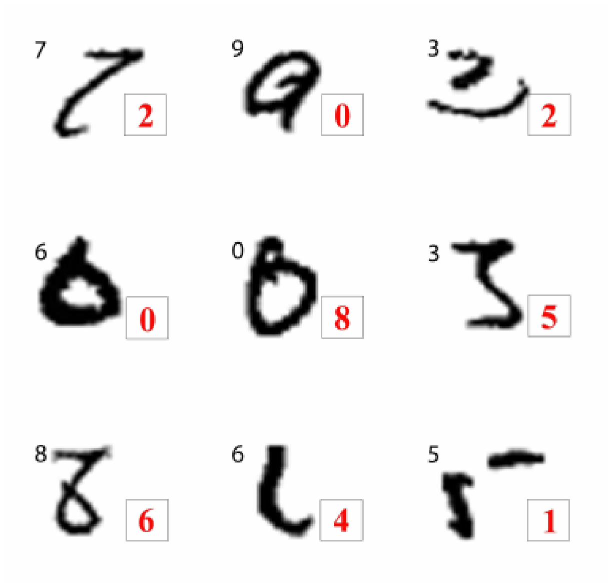

Figure 3.2: Samples of ambiguous digits [5].

The dataset is also often constructed as too small and too simple, consisting of only 70,000 images of greyscale images. Compared to datasets such as CIFAR10 which has the same amount of images but 32x32 colour images or another dataset like ImageNet consisting of 1,331,167 images, MNIST is certainly behind. However, MNIST has been used and is continued to be used as a way to quickly validate a machine learning algorithm before moving on to larger more complicated datasets that take much more time to compute.

3.2.3 Neural network model

The neural network model, I plan to build is one that will take the current floating-point input values, convert those to stochastic numbers, process it in some shape or form and then convert those numbers back to floating-point values, this format should allow me to drop this custom layer anywhere where I wish to test the stochastic process. A good starting point for this is firstly comparing the stochastic model with a regular MLP model and then moving to more complex models like CNN models.

Chapter 4

Project plan

4.1 Setup

For my thesis, most of the tests will be run on my home computer with any long-running tests, run on Amazon cloud servers. Each computer will need to have Tensorflow (with Keras extension), Python 3 and an AMD or NVidia graphics card.

4.2 Baselines

For this part of the project, I need to gather baselines that I will use to compare the stochastic model, this baseline will need to be run multiple times with different models and variations on the large amounts of parameters. For the baselines, I have identified 3 models which I will compare my stochastic model with

- MLP (Multi Layer Perceptron) model
- CNN (Convolutional Neural Network) model
- RNN (Recurrent Neural Network) model

The datasets, I've chosen to train the models with are,

- MNIST
- Fashion MNIST
- CIFAR-10

These datasets are small enough to be able to get results quickly but complex enough to compare results between models.

Also considering the large variation from test to test, all tests will be repeated 20 times and plotted on a box chart. Some other parameters that need to be taken into consideration,

- Epochs: 20

- Batch size: 128
- Optimizer: RMSprop

4.3 Stochastic model

During this stage of the project, I will create my custom stochastic model and run it through the datasets and compare the baselines, if anything interesting pops up I plan to explore that avenue. Otherwise, I will manipulate various settings of both the baseline and the stochastic model to try and find contrasting differences.

Some parameters that I plan to manipulate to find differences are

- Dense layer size
- Amount of hidden layers
- Noise in datasets
- Amount of epochs
- Bitsize of the stochastic model
- Smaller datasets

4.4 Compilation

Nearing the end of my thesis, all of the data will be tabulated on spreadsheets and compiled into various graphs and tables. Most of the data will go onto box charts to show the mean differences and variation between each test.

Chapter 5

Preliminary results

5.1 Stochastic numbers

5.1.1 Stochastic numbers: Abstract

These preliminary results are based upon the idea of if we could use stochastic numbers as a way of improving neural networks, whether that is through the reduction or datasets, noise tolerance, computation complexity or anything that would give it some edge over a more traditional network. The idea mainly comes from spiking neural networks and more theories about how the human brain works on probabilistic methods rather than hardcoded logic.

All of these results were trained with the MNIST dataset, 20 repetitions using a normal model with the following structure.

1. Dense input layer
2. Dropout layer (0.2 probability)
3. Dense layer (RELU activation)
4. Dropout layer (0.2 probability)
5. Dense layer (Softmax activation)

The stochastic model was an extension of the normal model but the first layer was the custom Stochastic layer.

1. Custom Stochastic layer
2. Dense input layer
3. Dense layer (RELU activation)
4. Dense layer (Softmax activation)

5.1.2 Bit size

The following graph show the accuracy results when the bit size of the stochastic layer was modified.

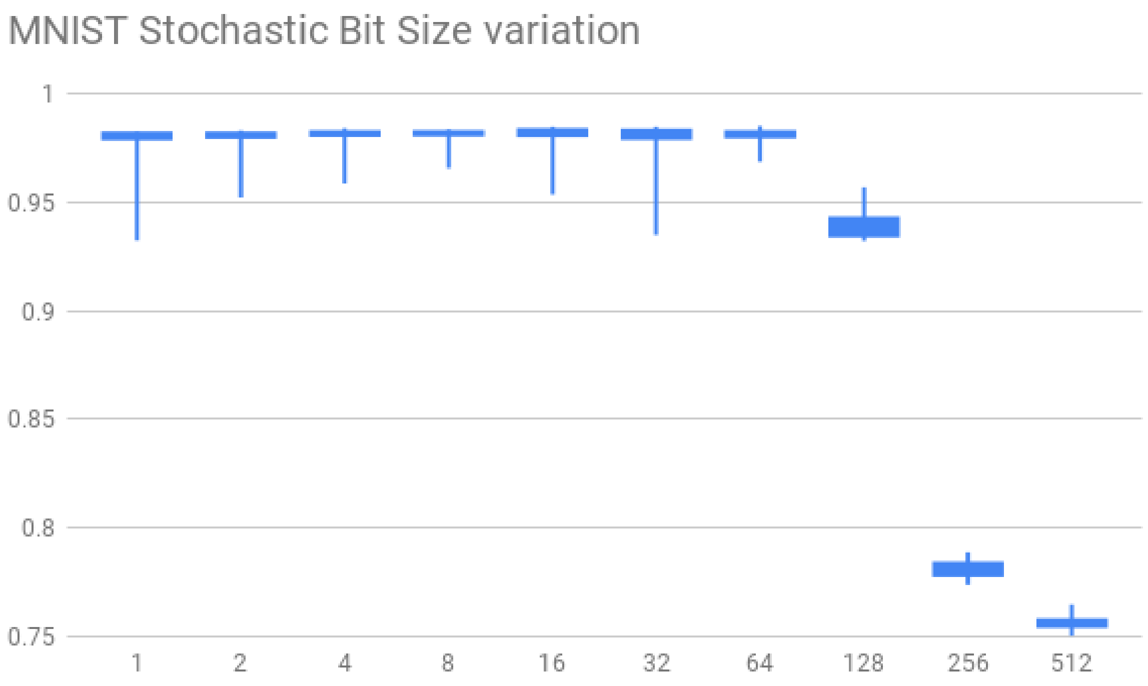

Figure 5.1: Bit size results.

From this test we can see that there is not much difference when I start increasing the number of bits, so for the rest of these tests, they were done with 8 bits.

However interestingly to note that above 128 bits we start seeing a large drop in accuracy, I'm unsure of the reason behind this.

5.1.3 Noise tests

The following graph shows the accuracy results when a normal model and a stochastic model are trained with the MNIST dataset and then fed the testing dataset from the Noisy MNIST dataset.

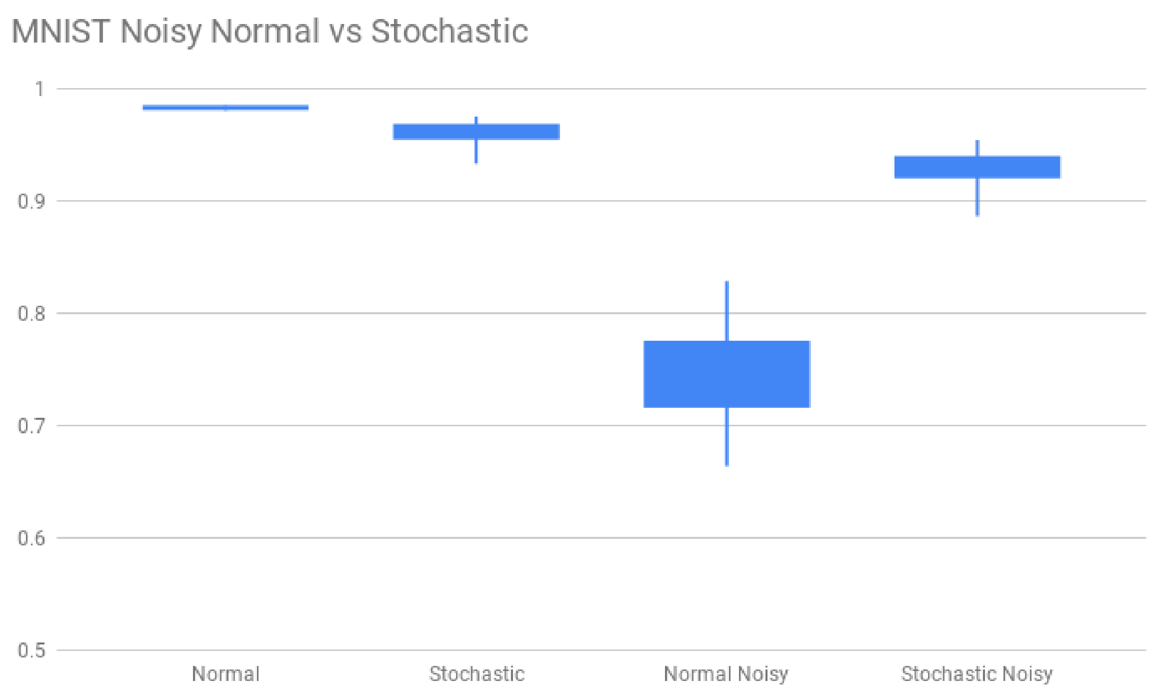

Figure 5.2: Noise test results.

In this test, we can see compared to the normal, the stochastic model loses 0.002% in accuracy. However when both of these pre-trained models are fed the noisy MNIST dataset, the stochastic model fairs considerably better.

However, upon further consideration, this seems to be an artifact of the dataset, after some investigation, there was a slight mistake in the custom stochastic layer which meant that all the values in the training and testing datasets for the stochastic layer were inverted. This should not have affected the results, comparing the images below shows an image that has been passed through the stochastic layer and both the mistake and fixed version shows minor differences other than the inversion.

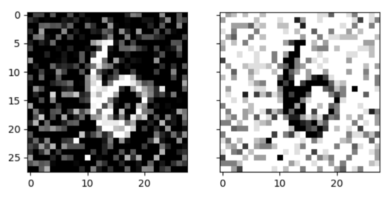

Figure 5.3: Left is original, right is inverted stochastic layer result.

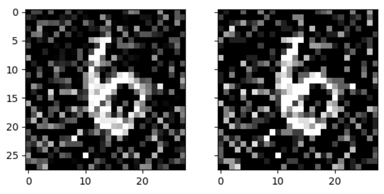

Figure 5.4: Left is original, right is corrected stochastic layer result.

After the correction, the advantage disappears.

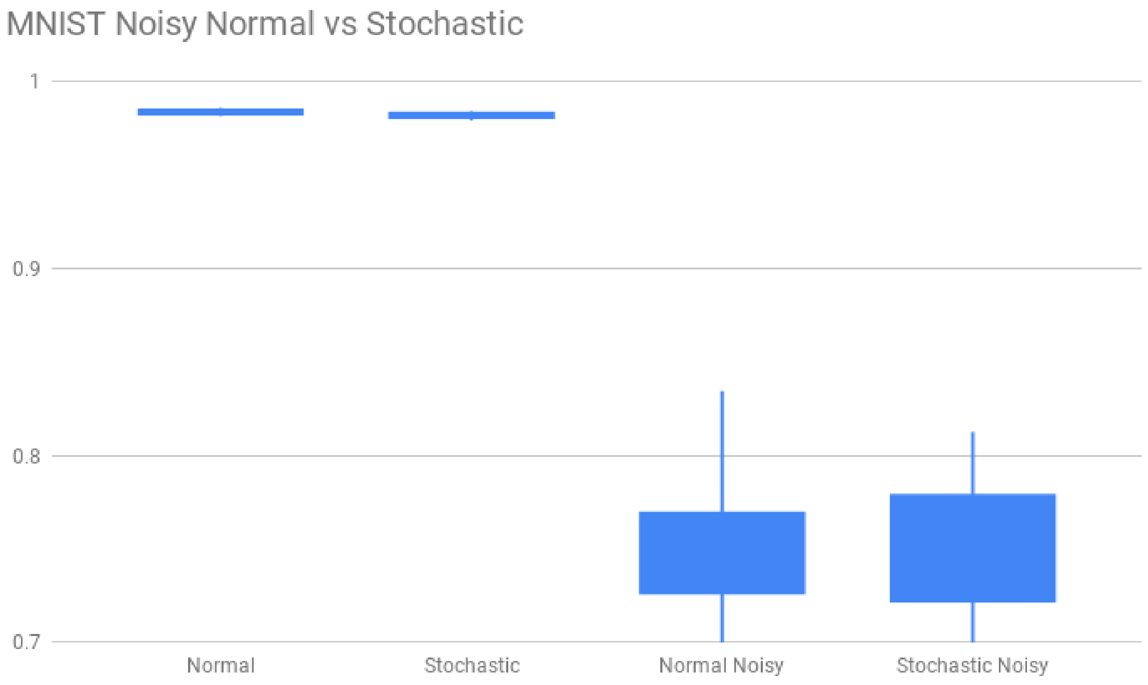

Figure 5.5: Corrected noise test results.

This is further shown by inverting the test and training data for the normal model, this now shows that the results have flipped and that this result is an artefact of the dataset rather than the stochastic layer.

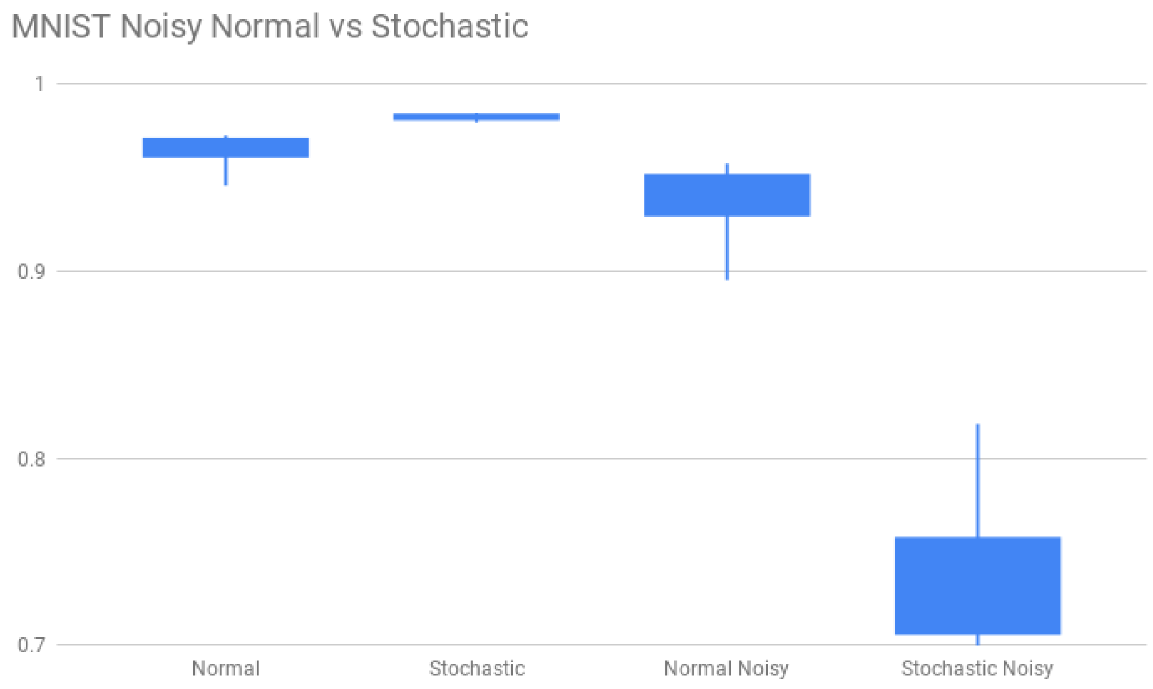

Figure 5.6: Inverted noise test results.

5.1.4 Noise shuffle

The following graph shows the accuracy results when we take both the normal dataset and the noisy dataset and shuffle them together, producing a new dataset that contains normal and noisy MNIST images.

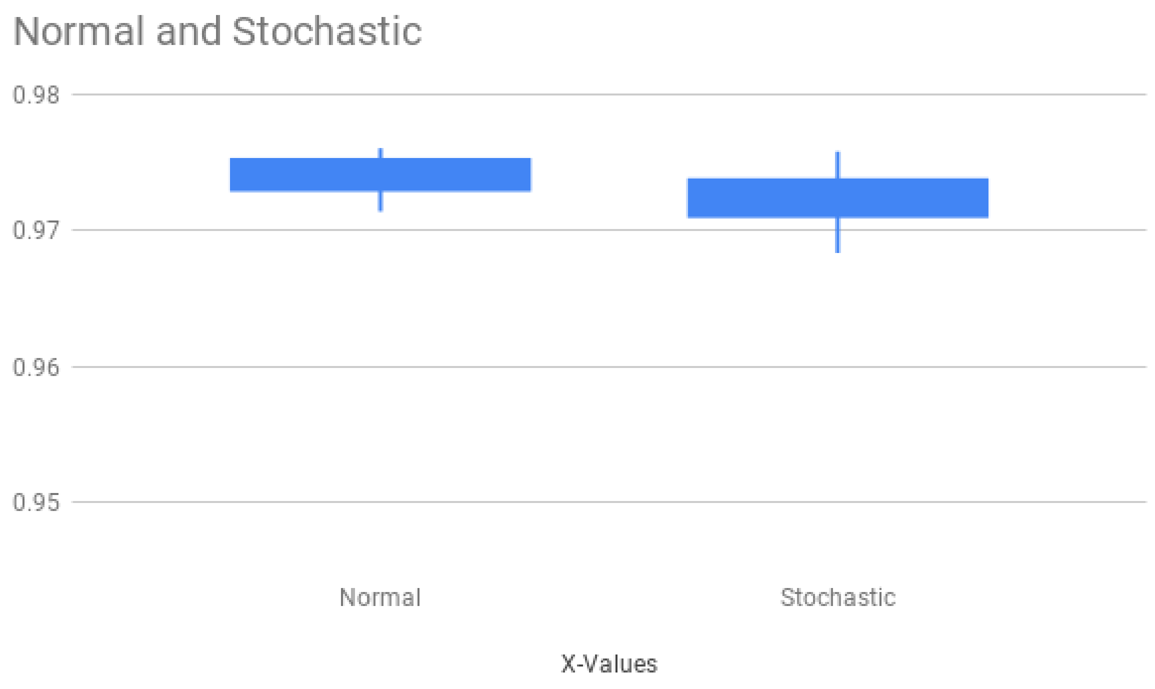

Figure 5.7: Noise shuffle results.

The stochastic model performs similarly to the normal model, nothing much to note here.

5.1.5 Reduced dataset

The following graph shows the accuracy of both the normal model and stochastic model as the size of the training datset is reduced.

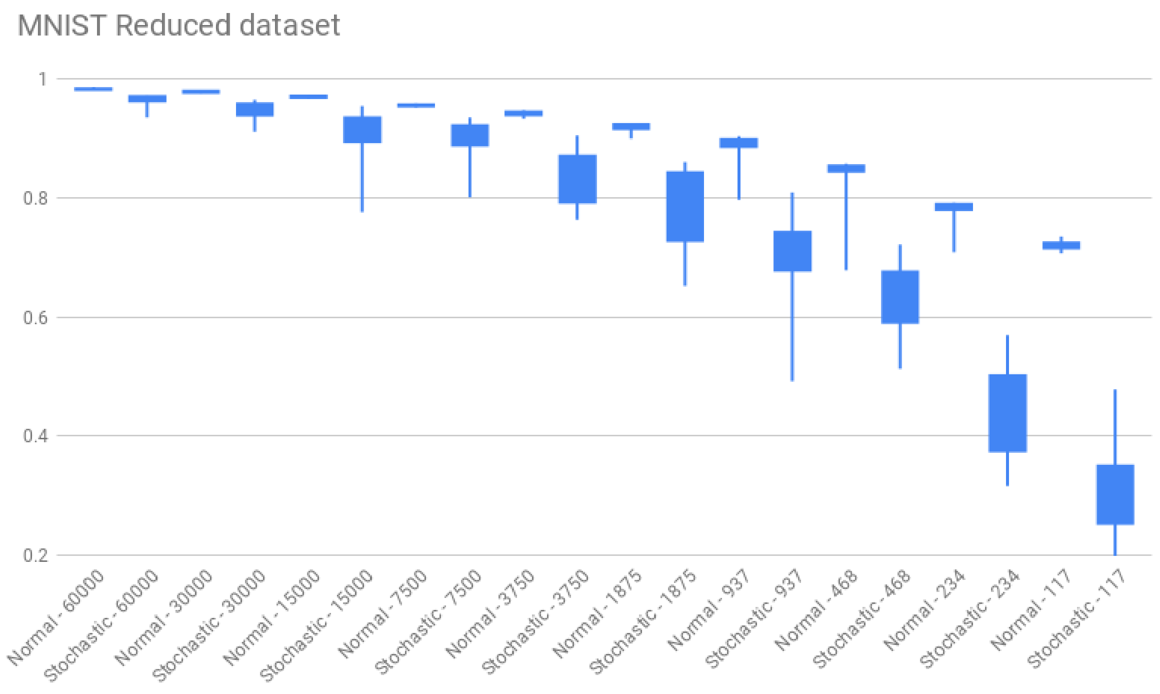

Figure 5.8: Bit size results.

The stochastic model has significantly worse performance as the dataset size decreases.

5.1.6 Reduced Epochs

The following graph shows the accuracy of the test dataset as training epochs are increased.

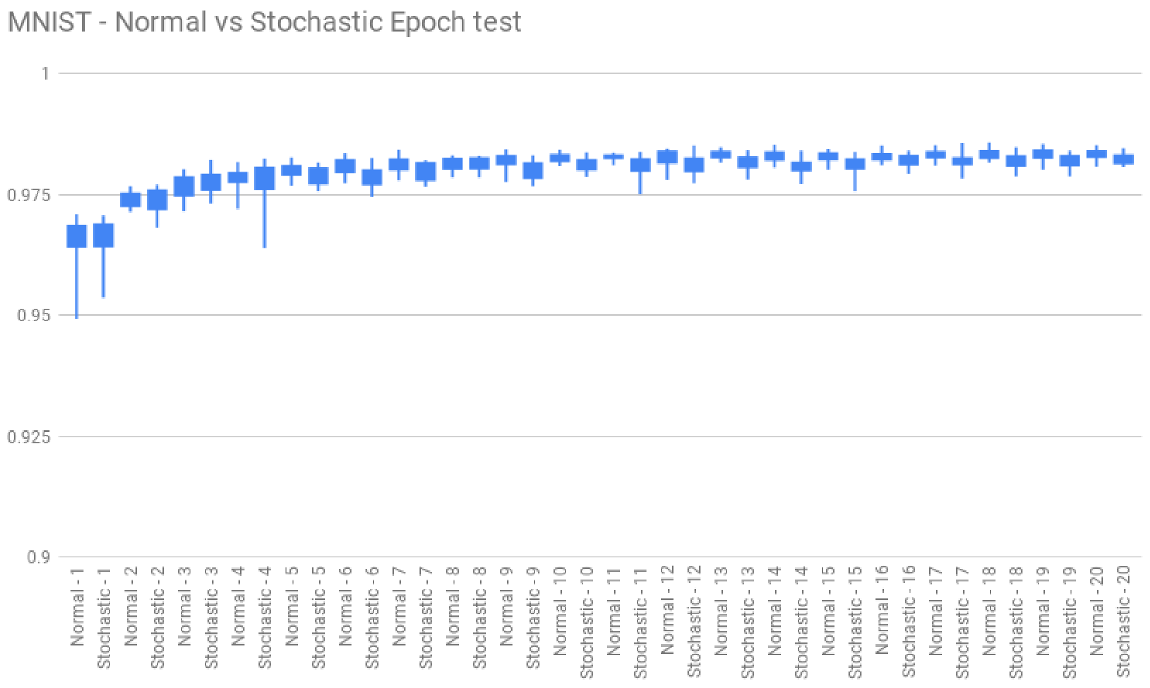

Figure 5.9: Epoch results.

The stochastic model performs similarly to the normal model, nothing much to note here.

5.1.7 Convolution2D

For this test, the model was changed to use a convolution model of the following form

1. Conv2D(32 neurons, (3x3 kernel), RELU activation)
2. MaxPooling2D((2x2 kernel))
3. Conv2D(64 neurons, (3x3 kernel), RELU activation)
4. MaxPooling2D((2x2 kernel))
5. Conv2D(64 neurons, (3x3 kernel), RELU activation)
6. Flatten layer
7. Dense layer (64 neurons, RELU activation)
8. Dense layer (10 neurons)

With the stochastic model being

1. StochasticLayer(728 neurons)
2. Reshape
3. Conv2D(32 neurons, (3x3 kernel), RELU activation)
4. MaxPooling2D((2x2 kernel))
5. Conv2D(64 neurons, (3x3 kernel), RELU activation)
6. MaxPooling2D((2x2 kernel))
7. Conv2D(64 neurons, (3x3 kernel), RELU activation)
8. Flatten layer
9. Dense layer (64 neurons, RELU activation)
10. Dense layer (10 neurons)

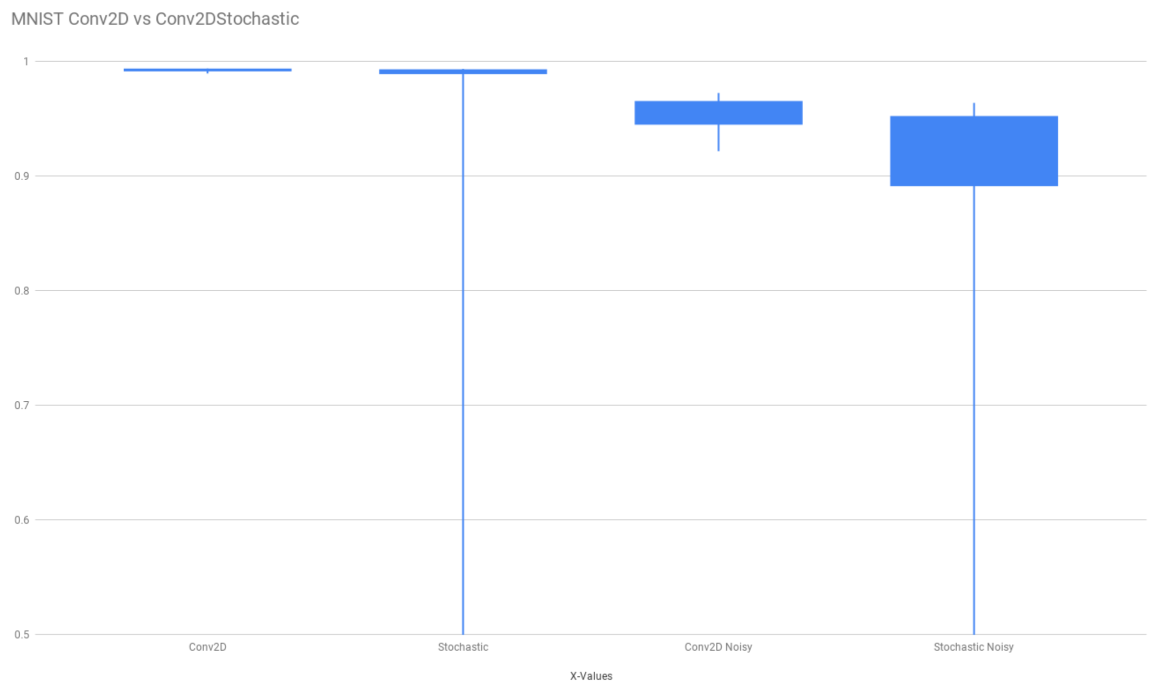

Figure 5.10: Conv2D results.

The stochastic model performs similarly to the normal model, nothing much to note here.

5.1.8 Dense layer size

The following graph shows the accuracy of the test dataset as the size of the hidden dense layers neurons are increased.

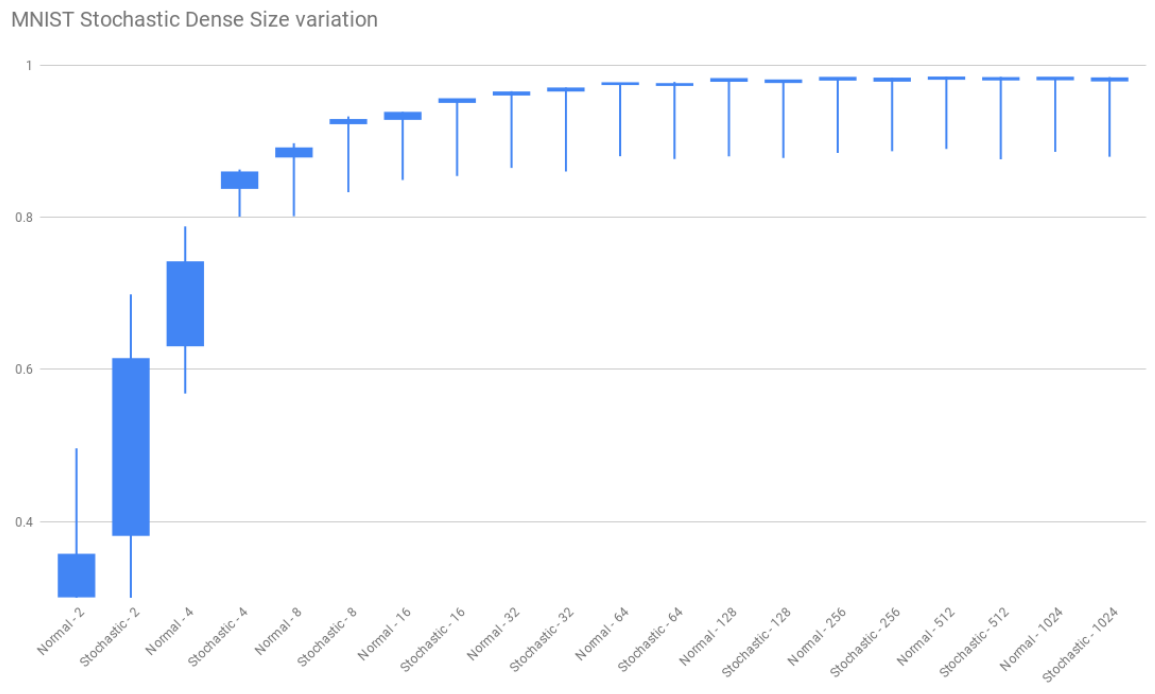

Figure 5.11: Dense layer results.

This is by far the most interesting results so far and thus the results above are 30 samples instead of the usual 20 samples. The results show that for hidden dense layer sizes of 32 and below the stochastic layer fairs much better, however, I'm still unsure if this is because of the stochastic layer or another artefact of the dataset, I will continue to research.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In conclusion, it seems that the stochastic layer has a minimum impact on the performance of neural networks in all the tests I have done and in some case worse performance. My speculation as to why the stochastic layer is not enough to cause any drastic improvements is that the stochastic layer is not enough of a change of the standard neural network and seems to be just adding noise which causes a slight performance dip, however regarding the dense layer test there seems to be some ground to be gained if the addition of this noise in small layer size networks perform better than the standard model.

6.2 Future work

My next steps for this project is to look into alternative ways of trying to use the stochastic network, another representation that might yield better results and into the dense layer tests to see how noise is affecting the neural network. Notably, MNIST may not provide the complexity that I'm looking for, especially after the inverted artefact. My next steps for this is to look into more complicated datasets like CIFAR100 to see if I can find performance in that area.

Chapter 7

Abbreviations

AWGN	Additive White Gaussian Noise
SGD	Stochastic Gradient Descent
SC	Stochastic numbers
NN	Neural network

.1 Codebase and results

https://github.com/AeroX2/stochastic-demo

Bibliography

[1] Neuron model figure. [Online]. Available: https://github.com/trekhleb/machine-learning-octave/blob/master/neural-network/README.md

[2] Sigmoid figure. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Logistic-curve.svg/320px-Logistic-curve.svg.png

[3] Neural network model figure. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/1200px-Colored_neural_network.svg.png

[4] Mnist label examples figure. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png

[5] Incorrect mnist labels figure. [Online]. Available: https://www.researchgate.net/profile/Antonio_Zippo/publication/232906025/figure/fig3/AS:300525175230467@1448662193349/Sample-of-common-incorrect-classifications-on-MNIST-dataset-Numbers-in-the-upper-left-of.png

[6] M. Buckland. Using neural nets to recognize handwritten digits. [Online]. Available: http://www.ai-junkie.com/ann/evolved/nnt1.html

[7] (2014) Neural network history. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html

[8] K. Gurney, *An Introduction to Neural Networks.* Bristol, PA, USA: Taylor & Francis, Inc., 1997.

[9] M. A. Nielsen. (2018) Neural networks and deep learning. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap4.html

[10] N. Donges. (2018) Gradient descent in a nutshell. [Online]. Available: https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0

[11] Overfitting in machine learning: What it is and how to prevent it. [Online]. Available: https://elitedatascience.com/overfitting-in-machine-learning#overfitting-vs-underfitting

[12] J. Heaton. The number of hidden layers. [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html

[13] K. Fredenslund. Computational complexity of neural networks. [Online]. Available: https://kasperfred.com/series/introduction-to-neural-networks/computational-complexity-of-neural-networks

[14] R. M. Sadek, S. A. Mohammed, A. R. K. Abunbehan, A. K. H. A. Ghattas, M. R. Badawi, M. N. Mortaja, B. S. Abu-Nasser, and S. S. Abu-Naser, "Parkinson disease prediction using artificial neural network," *International Journal of Academic Health and Medical Research (IJAHMR)*, vol. 3, no. 1, pp. 1–8, 2019.

[15] R. Ferduła, T. Walczak, and S. Cofta, "The application of artificial neural network in diagnosis of sleep apnea syndrome," in *Advances in Manufacturing II*, J. Trojanowska, O. Ciszak, J. M. Machado, and I. Pavlenko, Eds. Cham: Springer International Publishing, 2019, pp. 432–443.

[16] K. Feng, X. Pi, H. Liu, and K. Sun, "Myocardial infarction classification based on convolutional neural network and recurrent neural network," *Applied Sciences*, vol. 9, no. 9, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/9/1879

[17] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[18] B. Li, Y. Qin, B. Yuan, and D. J. Lilja, "Neural network classifiers using stochastic computing with a hardware-oriented approximate activation function," in *2017 IEEE International Conference on Computer Design (ICCD)*, Nov 2017, pp. 97–104.

[19] V. Canals, A. Morro, A. Oliver, M. L. Alomar, and J. L. Rosselló, "A new stochastic computing methodology for efficient neural network implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 551–564, March 2016.

[20] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1515–1531, Aug 2018.

[21] B. R. Gaines, "Chapter 2 stochastic computing systems."

[22] J. Yu, K. Kim, J. Lee, and K. Choi, "Accurate and efficient stochastic computing hardware for convolutional neural networks," in *2017 IEEE International Conference on Computer Design (ICCD)*, Nov 2017, pp. 105–112.

[23] (2015) Tensorflow. [Online]. Available: https://www.tensorflow.org/

[24] Y. Lecun. (2019) Mnist handwritten digit database, yann lecun, corinna cortes and chris burges. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.