Basic Neural Networks: MNIST

Neural Networks and Deep Learning

Abbiegael Chu
AJ23SYD009
October 13, 2023

TABLE OF CONTENTS

Introduction 2
Literature Review 2
A Brief History of Neural Networks 2
A Brief Overview of MNIST Dataset 3
Methodology 3
Results 4
(1) Basic MNIST Model 5
(2) MNIST Baseline Model 5
(3) MNIST Categorical Cross Entropy 6
(4) MNIST Sparse Categorical Cross Entropy 7
Challenges 7
References 8

INTRODUCTION

In the data science field, more specifically the machine learning and artificial intelligence area, neural networks have become a staple in coding, becoming the foundation of some groundbreaking applications. As the name suggests, neural networks were modeled after the human brain's neurons, even looking like a real biological neuron at that. To keep it short, these artificial neurons take in input, and after some mathematical calculations, they will be able to generate an output. The output's accuracy greatly depends on how the data scientist structured the network, but at its core, neurons should be able to give mathematical outputs or classification outputs, which the student will be covering in this report.

In this report, the student will be discussing a brief history of neural networks and an overview of the MNIST dataset. They will also go through their methodology and results on using the MNIST dataset to code a model that can accurately identify which number the handwritten image shows. Furthermore, the report will discuss the results of small changes made to the model's parameters and how they have affected the overall accuracy and loss of the model during validation.

LITERATURE REVIEW

A BRIEF HISTORY OF NEURAL NETWORKS

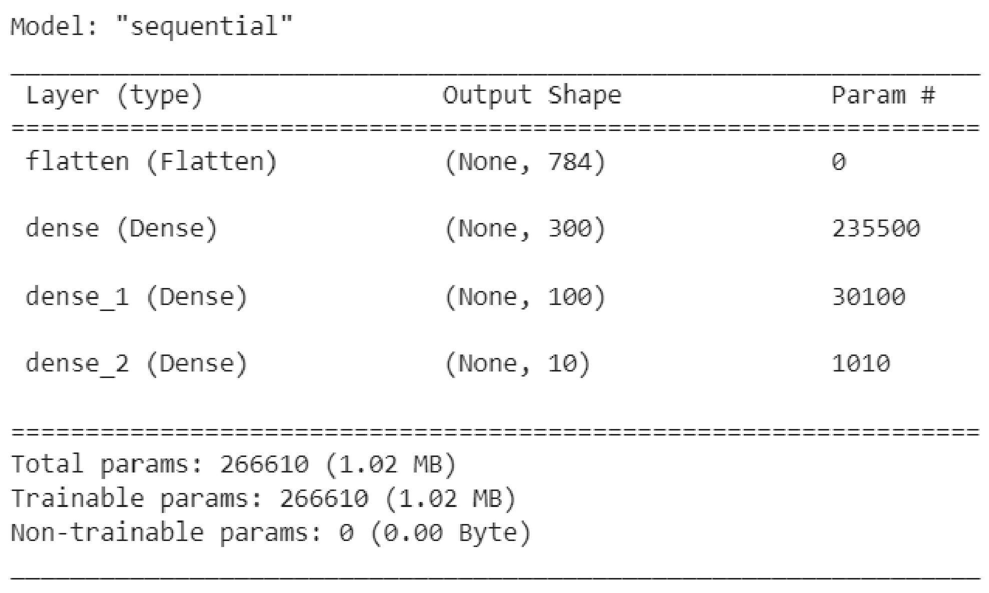
Neural networks are the foundation of deep learning models, which were based on biological neurons found in the human brain (Goodfellow, Bengio, & Courville, 2016). Models from the computational systems formed by the interconnectedness of artificial neurons were proposed by (McCulloch & Pitts, 1990), where they outlined how these artificial neural networks could compute any logical function, theoretically. As the advancement of technology went on, so did the development of algorithms. A notable development was the backpropagation training algorithm, which had allowed neural networks to have a wider application. Backpropagation had the ability to adjust the weights of the connected neurons based on the error of predictions (Rumelhard, Hinton, & Williams, 1986). During the early phases of neural networks, it had faced skepticism due to its exploding gradient problems, but the development of regularization techniques, like dropout, and more advancements in activation functions, renewed the interest in these networks (Krizhevsky, Sutskever, & Hinton, 2017).

A BRIEF OVERVIEW OF MNIST DATASET

The MNIST dataset is comprised of handwritten images of numbers from 0 to 9. The dataset is highly accessible to beginners looking to delve into neural networks and deep learnings, providing a dataset that is easy to explore and test out simple hypotheses and questions the beginners may have. The dataset contains a total of 70,000 handwritten images, splitting by 60,000 training arrays and 10,000 testing images. Along with the pixel arrays, it also contains the label of these arrays, which indicates what digit it is.

METHODOLOGY

As a starting point, the student used the same neural network structure found in the clothes classification exercise done in class, which can be found in the 1_MNIST Trial notebook. The sequential model was built with an input layer that flattens the 2D arrays of 28x28 pixels into one long 1D array. A sequential model stack the layers on top of each other, wherein each layer has exactly one input and one output tensor (Chollet, 2020). The inputs are then passed through two dense hidden layers with 300 and 100 neurons, respectively. The activation function for both is Rectified Linear Unit, also known as ReLu. According to (DeepAI, n.d.), the ReLu activation function is traditionally used for non-linear functions, similar to the sigmoid and logistic functions. ReLu is noted to replace sigmoid and hyperbolic tangent due to its ability to speed up training speed, making it less computationally expensive. The final layer of the output is a softmax layer used for multiclass classification. In this project, the softmax layer will give an output of 0 to 9. The sequential model is summarized in Figure 1.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 300)               235500

 dense_1 (Dense)             (None, 100)               30100

 dense_2 (Dense)             (None, 10)                1010

=================================================================
Total params: 266610 (1.02 MB)
Trainable params: 266610 (1.02 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 1 Basic MNIST Model Summary

Compiling the model is the next step, and the parameters for loss, optimizer, and metrics need to be set to move forward. The parameters chosen were Sparse Categorical Cross Entropy for loss, which computes loss between labels and predictions (TensorFlow, n.d.), sgd for optimizer, which means it uses the stochastic gradient descent method as a means for optimization, and Accuracy for metrics, which measures how accurate the model's predictions are. It is important to note that this report will be looking at both loss and accuracy values during the testing phase.

Training and testing the model are the next steps. The model was trained using 20 epochs and had a batch size of 512. Indicating a batch size of 512 means that a mini batch was selected to make the training less computationally expensive by only processing 512 random samples from the training dataset. Models were then evaluated by getting the loss and accuracy using the test dataset. The student then used the model to make predictions of the first three samples of the test set.

This basic neural network for the MNIST dataset was then used as a benchmark for the student to explore different parameters, number of neurons, number of layers, and other normalization techniques and see their effects on the loss and accuracy of the model. The student experimented on loss, using sparse categorical cross entropy and categorical cross entropy, and the addition of normalizing data using dropout layers. The student also tested the increase and decrease of both the number of neurons and layers, which will be discussed in the Results section.

RESULTS

The following results will consider the loss and accuracy of testing after training the model. The student created four notebooks for the MNIST dataset, (1) Basic MNIST Model, which follows the parameters of the clothes classification model, (2) MNIST Baseline, an improvement of the Basic MNIST Model by changing data pre-processing steps and additional normalization techniques, (3) MNIST Adam Categorical Cross Entropy, an extension of MNIST Baseline through a series of small parameter alterations, and (4) MNIST Sparse Categorical Cross Entropy, another extension of MNIST Baseline that focuses on Sparse Categorical Cross Entropy.

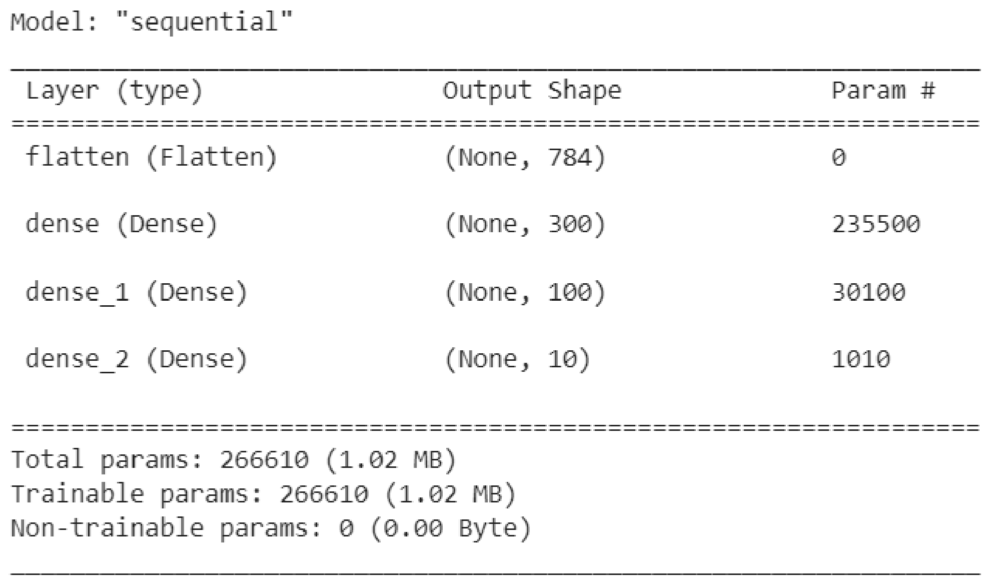
10 tests each were performed by notebooks 3 and 4. The tests focused on small parameter changes in the model through increasing and decreasing neurons in both layers, adding another layer of neurons, and tuning the dropout percentage. The base number of neurons for all layers is 128, and altering the number of neurons was done in increments of +/- 28. Test descriptions were outlined as follows:

1. Decrease Neurons in 2nd layer
2. Increase Neurons in 2nd layer
3. Increase Neurons in 1st layer
4. Decrease Neurons in 1st layer

5. Increase Neurons in 1st and 2nd layer
6. Increase Neurons in 1st layer and Decrease Neurons in 2nd layer
7. Increase Dropout to 50%
8. Increase Dropout to 50% and Increase Neurons on both layers
9. Increasing number of layers to 3
10. Increasing number of layers to 3 and Increasing Dropout to 50%

(1) BASIC MNIST MODEL

As shown in Figure 1, the Basic MNIST Model has an input layer that flattens inputs into a 1D array, two dense hidden layers, and an output layer. The models' configuration used sparse categorical cross entropy as the loss, sgd as the optimizer, and accuracy as the metric. Epochs were set to 20 when training the model with a batch size of 512. The model's evaluation showed that loss was extremely high at 35.08, while accuracy was somewhat high at 0.91. The high loss was a result in not applying one-hot encoding to the validation data during pre-processing.

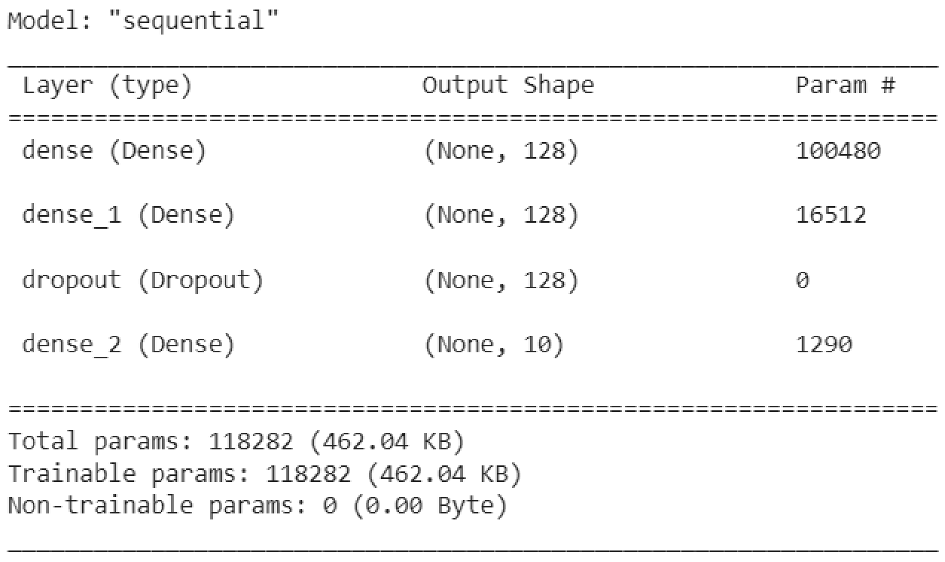


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 300)               235500

 dense_1 (Dense)             (None, 100)               30100

 dense_2 (Dense)             (None, 10)                1010

=================================================================
Total params: 266610 (1.02 MB)
Trainable params: 266610 (1.02 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 2 Basic MNIST Model Summary

(2) MNIST BASELINE MODEL

As shown in Figure 2, the MNIST Baseline Model has an input layer, two dense hidden layers with 128 neurons each, a dropout layer set at 0.25, and an output layer. The models' configuration used categorical cross entropy as the loss, adam as the optimizer, and accuracy as the metric. Epochs were set to 20 when training the model with a batch size of 512. Based on the learnings from the first model, pre-processing included one-hot encoding of the validation data as well as turning the 2D arrays of X into 1D arrays to minimize computation. The model's evaluation showed that loss has significantly improved to 0.07, while accuracy was high at 0.981.

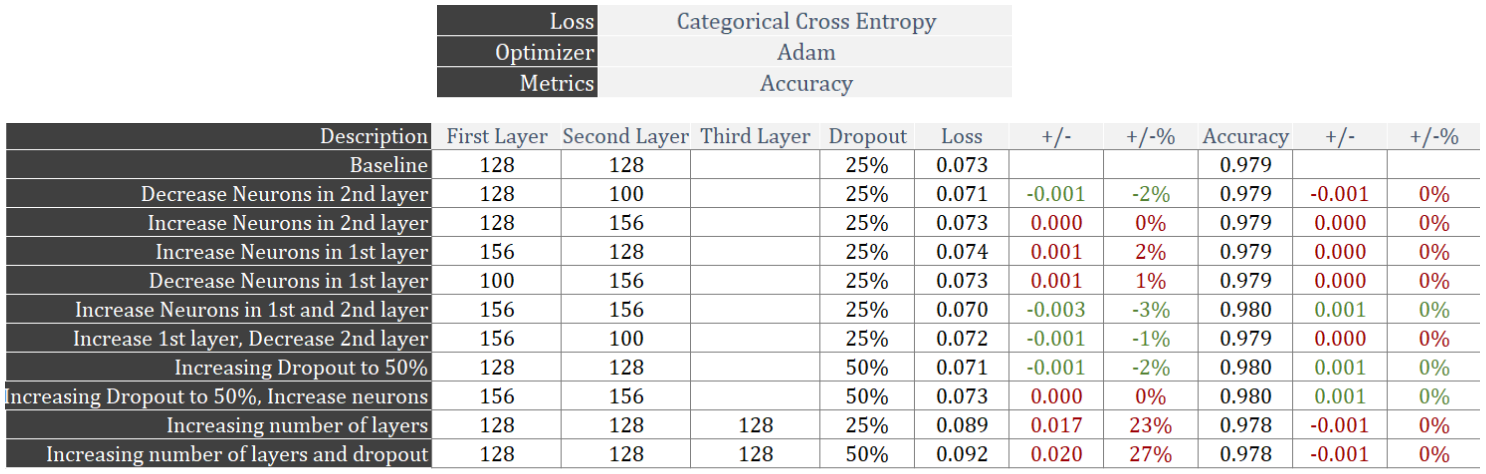


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 128)               16512

 dropout (Dropout)           (None, 128)               0

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 118282 (462.04 KB)
Trainable params: 118282 (462.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3 MNIST Baseline Model

(3) MNIST CATEGORICAL CROSS ENTROPY

MNIST Categorical Cross Entropy Notebook used the model from MNIST Baseline as a benchmark. Figure 4 below shows the parameters of the model as well as the loss and accuracy of the test data.



Loss	Categorical Cross Entropy
Optimizer	Adam
Metrics	Accuracy

Description	First Layer	Second Layer	Third Layer	Dropout	Loss	+/-	+/-%	Accuracy	+/-	+/-%
Baseline	128	128		25%	0.073			0.979		
Decrease Neurons in 2nd layer	128	100		25%	0.071	-0.001	-2%	0.979	-0.001	0%
Increase Neurons in 2nd layer	128	156		25%	0.073	0.000	0%	0.979	0.000	0%
Increase Neurons in 1st layer	156	128		25%	0.074	0.001	2%	0.979	0.000	0%
Decrease Neurons in 1st layer	100	156		25%	0.073	0.001	1%	0.979	0.000	0%
Increase Neurons in 1st and 2nd layer	156	156		25%	0.070	-0.003	-3%	0.980	0.001	0%
Increase 1st layer, Decrease 2nd layer	156	100		25%	0.072	-0.001	-1%	0.979	0.000	0%
Increasing Dropout to 50%	128	128		50%	0.071	-0.001	-2%	0.980	0.001	0%
Increasing Dropout to 50%, Increase neurons	156	156		50%	0.073	0.000	0%	0.980	0.001	0%
Increasing number of layers	128	128	128	25%	0.089	0.017	23%	0.978	-0.001	0%
Increasing number of layers and dropout	128	128	128	50%	0.092	0.020	27%	0.978	-0.001	0%

Figure 4 MNIST Categorical Cross Entropy Model

The results show that the best accuracy and loss improvement was from Increasing the Dropout to 50% while maintaining 128 neurons for the two layers of the network. However, the improvements seen were minimal.

(4) MNIST SPARSE CATEGORICAL CROSS ENTROPY

MNIST Sparse Categorical Cross Entropy Notebook used the model from MNIST Baseline as a benchmark and changed the loss to Sparse Categorical Cross Entropy. Figure 5 below shows the parameters of the model as well as the loss and accuracy of the test data. Compared to Notebook 3, the Sparse Categorical Cross Entropy saw much more improvement in both loss and accuracy, with all accuracy scores improving. Among these tests, the best model was from decreasing neurons in the first layer and increasing the neurons in the second layer. The loss was the lowest among all testing, while the 0.979 accuracy is still in the same range as the other better performing models.

Loss	Sparse Categorical Cross Entropy
Optimizer	Adam
Metrics	Accuracy

Description	First Layer	Second Layer	Third Layer	Dropout	Loss	+/-	+/-%	Accuracy	+/-	+/-%
Baseline	128	128		25%	0.079			0.977		
Decrease Neurons in 2nd layer	128	100		25%	0.076	-0.002	-3%	0.978	0.001	0%
Increase Neurons in 2nd layer	128	156		25%	0.077	-0.002	-2%	0.979	0.002	0%
Increase Neurons in 1st layer	156	128		25%	0.079	0.001	1%	0.978	0.001	0%
Decrease Neurons in 1st layer	100	156		25%	0.067	-0.011	-14%	0.979	0.003	0%
Increase Neurons in 1st and 2nd layer	156	156		25%	0.071	-0.007	-9%	0.981	0.004	0%
Increase 1st layer, Decrease 2nd layer	156	100		25%	0.078	-0.001	-1%	0.978	0.001	0%
Increasing Dropout to 50%	128	128		50%	0.072	-0.006	-8%	0.979	0.003	0%
Increasing Dropout to 50%, Increase neurons	156	156		50%	0.068	-0.011	-13%	0.981	0.004	0%
Increasing number of layers	128	128	128	25%	0.084	0.005	7%	0.979	0.002	0%
Increasing number of layers and dropout	128	128	128	50%	0.081	0.003	4%	0.980	0.003	0%

Figure 5 MNIST Sparse Categorical Cross Entropy Model

CHALLENGES

Challenges in this coding project were mostly on which starting parameters should be used as a baseline for the results. Hence, the clothes classification exercise from class was used as the start of a work-in-progress model for MNIST. Another challenge faced was understanding the effects of changing one parameter to the entire model. As the student aimed to create an optimal model with high accuracy and low loss, they faced many combinations to try. The results were inconclusive, not proving that adding another layer of neurons, increasing the number of neurons, or decreasing the number of neurons would drastically improve loss and accuracy.

REFERENCES

Chollet, F. (2020, April 12). *The Sequential model*. Retrieved from Keras: https://keras.io/guides/sequential_model/

DeepAI. (n.d.). *What is ReLu?* Retrieved from DeepAI: https://deepai.org/machine-learning-glossary-and-terms/relu

Geeks for Geeks. (2020, October 24). *Intuition of Adam Optimizer*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/intuition-of-adam-optimizer/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press.

Keras. (n.d.). *ReLU layer*. Retrieved from Keras: https://keras.io/api/layers/activation_layers/relu/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 84-90.

McCulloch, W. S., & Pitts, W. (1990). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 115-133.

Rumelhard, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 533-536.

TensorFlow. (n.d.). *tf.keras.losses.SparseCategoricalCrossentropy*. Retrieved from TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy