Computer Science
COSC 4P80 - Artificial Neural Networks

LeNet-5 Architecture on MNIST Dataset

Prepared by: Amani Anderson and Abhijeet Prajapati
Student Numbers: 6617344 and 5987722
Instructor: Dave Bockus
Date: May 17th 2023

Abstract

The purpose of this project is to extend and experiment with a deep learning paradigm that is trained and tested on an established dataset. There are a multitude of deep learning models that exist for countless task variations. One subset of these deep learning models rely on the concept of convolutional neural networks. Naturally, CNNs can vary vastly in design and architecture, depending on their precise purpose. In this project, we aim to extend the LeNet-5 Architecture to correctly interpret handwritten characters, which is supplied by the very well known MNIST dataset. The dataset is used to train and test our deep network. Along with this dataset, we opted to implement a GUI that allows for a user to draw a number, then feed that number as input for the model to predict. With the LeNet-5 architecture and the MNIST dataset, we expect a model with high performance as the LeNet-5 architecture is already known to be very accurate, particularly on the MNIST dataset. As is discussed later in this report, the findings are very positive.

Contents

1 Introduction

The Modified National Institute of Standards and Technology (MNIST) dataset is one of the most prominent and important datasets for image classification in computer vision. This dataset is a set of 70,000 images of handwritten digits from 0 to 9 that have all been normalized to a resolution of 28x28 pixels. There are several existing deep learning models that make use of this dataset, however one of the most successful ones, and the one used in this project, is the LeNet-5 Architecture. This architecture was developed by Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner in 1998. This deep learning paradigm uses 7 total layers to achieve the task of image classification, with its precise architecture explained further in this report. Using this architecture, we are able to classify the handwritten digits in the MNIST dataset with a high degree of accuracy. To further experiment with this model, we opted to add some testing data of our own. Though the MNIST dataset was divided accordingly for testing, we thought it would be interesting to test it on our own handwriting. The details of how this was implemented is explained in the experimental setup.

2 Experimental Setup

2.1 Model Setup

This project is built on Python, and as it is a deep learning project, we opted to use the Keras API for building our LeNet-5 model. The LeNet-5 architecture consists of seven total layers, with two convolutional layers, two subsampling layers and three fully connected layers. The network works by accepting a 28x28x1 pixel size grayscale image and passing it through its layers to predict the digit that the image represents. The precise architecture of each layer is explained as follows. The first layer is a convolutional layer, with six feature maps, each having a size of 5x5 pixels with a stride of one. The dimensions then become 28x28x6. The second layer is an average pooling layer with a feature map size of 2x2 and a stride of two. This reduces the image dimensions to 14x14x6. The third layer is the second convolutional layer, this time with 16 filters of size 5x5 and a stride of 1. This layer has an interesting property, as only 10 of these 16 filters are connected to the previous layer's 6 filters. This property curves the possibility of overfitting in the network by reducing the training parameters drastically. The output is then a dimension of 10x10x16. The fourth layer is the second average pooling layer, with 16 filters of size 2x2 and a stride of two, like the first one. It produces a reduced output dimensionality of 5x5x16. The fifth layer is a fully connected convolutional layer, with 120 filters of size 1x1. The sixth layer is simply a fully connected layer with 84 neurons. The last layer is then a fully connected layer, serving as a Softmax output layer to output 10 values that correspond to the digits ranging from 0 to 9. The activation function of choice throughout the LeNet-5 architecture is the TanH activation function. This choice of activation function over the Sigmoid function is due to its steeper gradient, more suited for deep learning tasks. The last layer uses a Softmax activation function as it is the output layer. This precise construction defines the LeNet-5 architecture, which makes for excellent results on the MNIST dataset. The MNIST dataset included with Keras is already divided into training and testing data, which we load into our program completely. This includes 60,000 training images and 10,000 testing images. Apart from the actual LeNet-5 model, we also opted to implement a GUI that allows the user to draw a number and input it into the network to test it for themselves. This allows for a more hands on and visual interpretation of the model's functionality. The GUI is simple, allowing a user to draw onto a canvas and press a predict button to feed the number into a saved model.

2.2 Code Running

Upon opening the source code, one would notice that there are three Python files. The GUI.py file contains the GUI class, with the functionality of creating a GUI for the user to write on and then feed into the model. The code for the model's implementation resides in leNet.py. the Project.py file can be seen as the main file that dictates what actions are performed, as it connects the GUI and LeNet classes.

3 Experiments Results

The results of the training and testing were as expected. Not only were the results high in accuracy, but the amount of epochs required to reach a desirable error percentage was lower than expected. For the following three experiments, a constant batch size of 128 was used. Using just 10 epochs, we achieve a training accuracy of 0.96 on the last epoch, and a testing accuracy of 0.96 as well. With 20 epochs, this training accuracy moves to 0.98, with the testing accuracy at 0.97. At 50 epochs, the training accuracy increases further to 0.99, with a testing accuracy of 0.98. The following three graphs are performed with a batch size of 128.

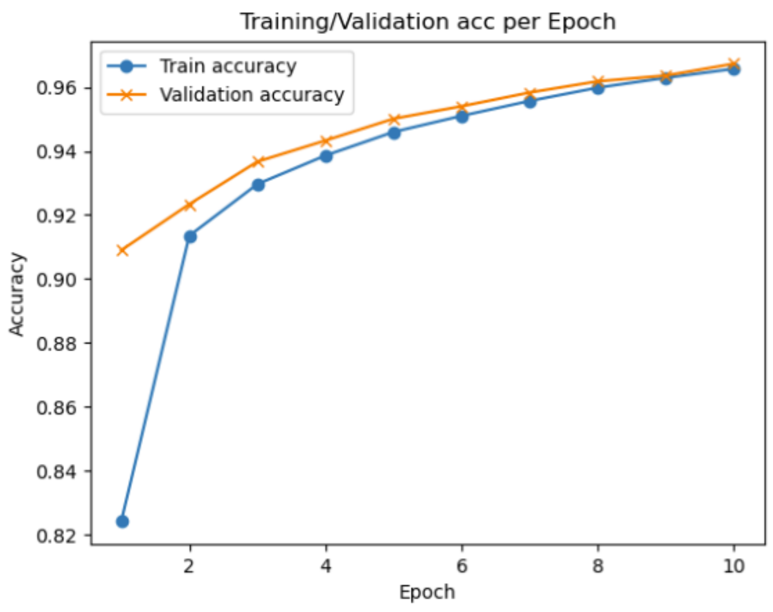


Figure 3.1: 10 Epochs, batch size 128

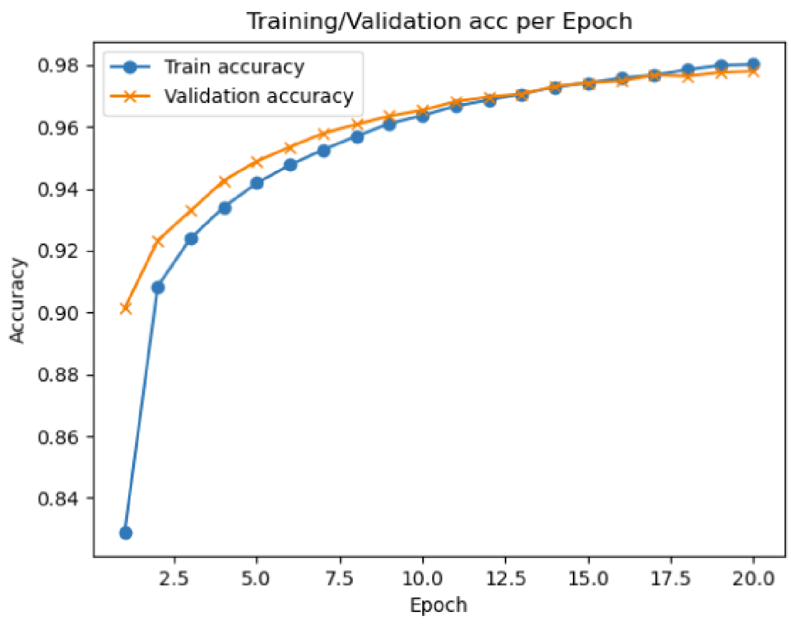


Figure 3.2: 20 Epochs, batch size 128

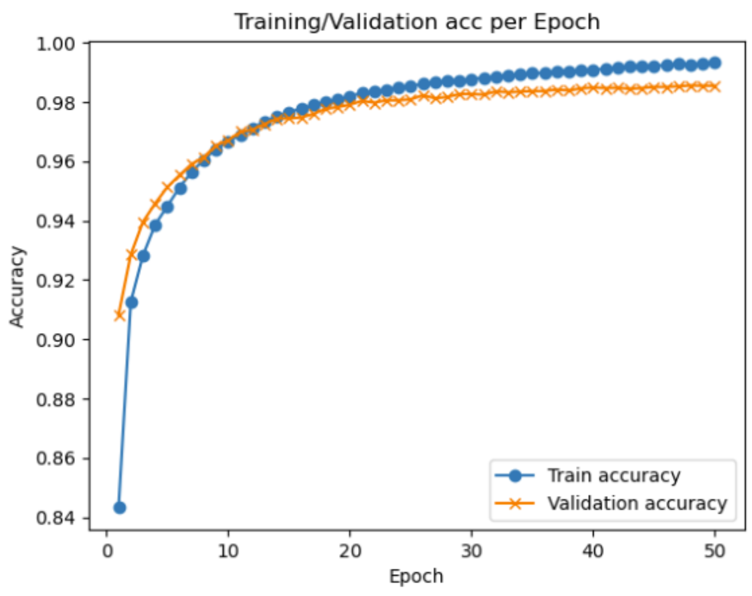


Figure 3.3: 50 Epochs, batch size 128

After this point, we experiment with different batch sizes at a constant epoch number of 20.

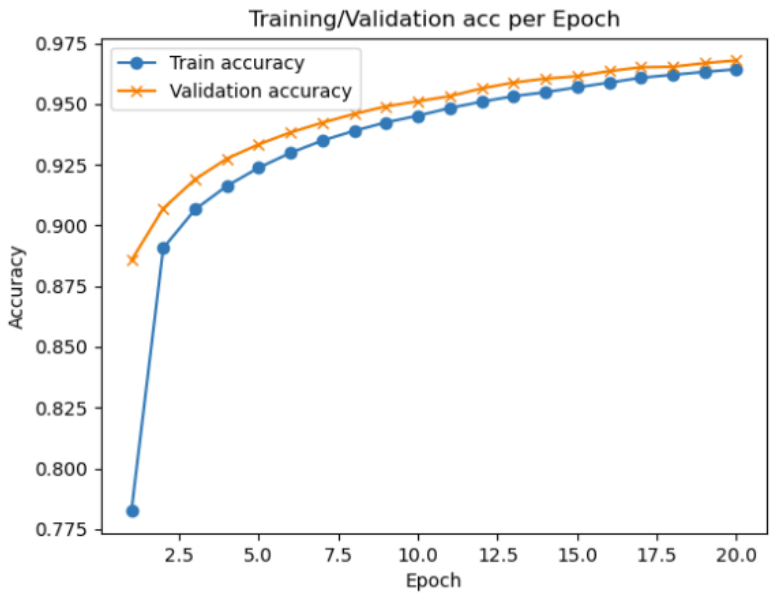


Figure 3.4: 20 Epochs, batch size 300

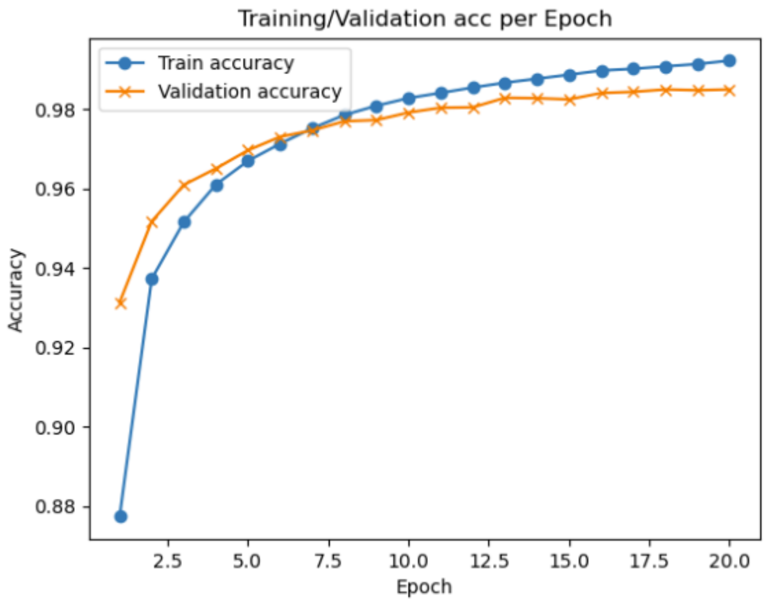


Figure 3.5: 20 Epochs, batch size 50

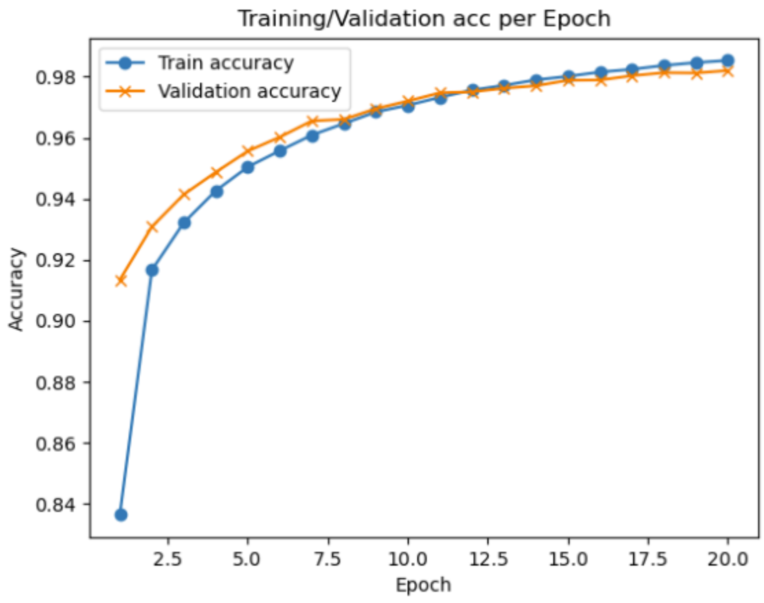
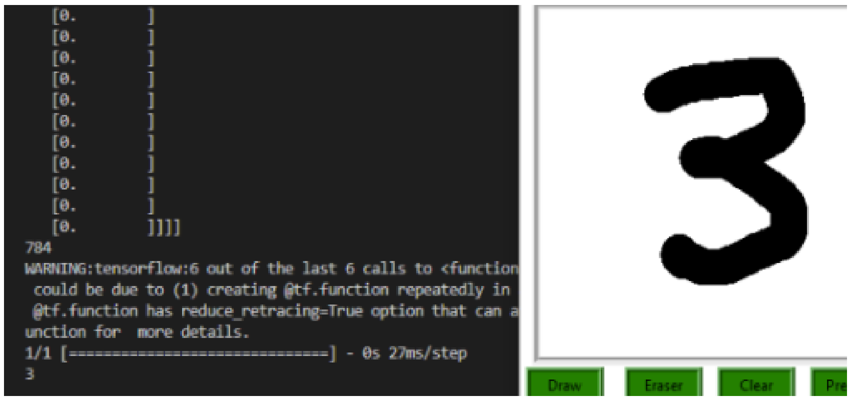


Figure 3.6: 20 Epochs, batch size 100

The following is an example of a manual drawing and correct classification



```
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]
   [0.        ]]]]
784
WARNING:tensorflow:6 out of the last 6 calls to <function
 could be due to (1) creating @tf.function repeatedly in
 @tf.function has reduce_retracing=True option that can a
unction for  more details.
1/1 [==============================] - 0s 27ms/step
3
```

Figure 3.7: Testing with a hand drawn number 3

4 Discussion and Conclusion

The LeNet-5 architecture proves to be an effective deep learning model that classifies data accurately on the MNIST dataset. This is expected, given its prominence and popularity in the artificial intelligence landscape. Based on the resultant graphs, it is the case that 10, 20 and 50 epochs result in sufficiently accurate classification. However, we could conclude that 20 epochs may be the sweet spot, as the validation and training arcs converge at just under 20 epochs as the validation arc continues to plateau. Upwards of this value, the cost of training is not rewarded with a significant boost in accuracy. Regarding batch size, the sweet spot is not a defined number, but seems to be around the 90-140 mark to achieve not only the best accuracy over the epochs, but also the best training time. Batch sizes of too large a size train quickly but are not accurate, and if the batch size is too small, it takes too long to train. Other parameters such as the filter size could be changed as well, however this does not come at any benefit. That's likely due to the fact that this model was studied by a group and its current architecture was deemed the best, which is why the configuration detailed in the experimental setup is used.
As with many machine learning problems, experimentation is key. With respect to this project, the LeNet-5 architecture proves easy to implement, and very powerful given a quality dataset like MNIST. Though it works very well on MNIST, it can be tuned and optimized to work well with a multitude of other datasets. Overall, this project demonstrates the effectiveness of the LeNet-5 architecture and convolutional neural networks as a whole, in image recognition problems such as recognizing handwritten digits. Its ability to mostly accurately classify handwritten digits that are not part of the dataset is also impressive.

5 References

https://www.tensorflow.org/datasets/catalog/mnist
https://pyimagesearch.com/2021/05/22/lenet-recognizing-handwritten-digits/
https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/
https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342