COMP 551 Project 3 Report

Krystal Xuejing Pan 260785873
Nghi Huynh 260632588
Corinne Robert 260795131

1 Introduction

In this paper, we focus on developing models to classify image data in a "modified" MNIST dataset. We performed some preprocessing on the provided synthetic MNIST dataset as well as data augmentation. We propose three deep neural networks with different architectures to compare the overall performance. These models are configured with multiple hidden layers, all with feedforward connections.

2 Data preprocessing and augmentation

For this project, we split the synthetic MNIST dataset and used 75% of the dataset as a training set and the remaining 25% as a validation set. We further processed the training set by extracting single digits images from the digit sequence in the synthetic MNIST images. To do so, we used OpenCV findcountours function to select the individual digits, then we resized every digit to a 24 by 24 image with 2 by 2 padding to get a training set consisting of 28 by 28 images. The preprocessing of the synthetic MNIST set can be found in the MNIST_preprocessing.ipynb notebook. To improve the validation accuracy of our model, we also performed some data augmentation [Tabik et al., 2017]. We performed 4 types of transformation on our single-digit training set, clockwise and counter-clockwise rotation by 10°, right translation, left translation and up translation. In the interest of time, the transformations were not performed on all the training set. Once we concatenated the augmented images with the single-digit set, we obtained a training set with 488 391 28 by 28 images, which we used to train all the proposed model. The code for the data augmentation performed can be found in the data_augmentation.ipynb notebook Notice that we also normalized the images during the training, validation and testing part of our optimal model (see section 3.4).

3 Deep neural network models

3.1 Multiple NN models (abandoned)

The first idea we had was to train a classifier before we design the actual neural network model. However, we later abandoned this idea as it is not practical. The classifier is used to classify how many digits are present in the given image. We used the linear kernel SVM and achieved a 99.7 accuracy score. However, we then realized that we need to train five different models (each for 1 to 5 number of digits in the image). Given that the numbers of training examples are distributed equally between 1-5 digits, we are faced with a severe issue of data. For example, there are ten classes for the single-digit classification, and there are 9000 training instances. Therefore, this neural network is likely to get decent performance. However, we have 100000 possible classes for the five-digit classification, yet we only have 9000 training examples. This would result in this model performing no better than random guessing. Thus, we abandoned this idea.

3.2 Sequential Model

The first model we purposed is a sequential model, without any convolutional layers. This model has five densely-connected layers with various numbers of units. Using all augmented data, this model achieved a 97.36 accuracy using a batch size of 512 and trained across six epochs. However, the test accuracy score is 94.5. Thus, the sequential model overfit. In an attempt to get better performance, we have experimented with increasing the number of dense layers. This modification increased the training accuracy yet leads to more severe overfitting as the accuracy score gap between the dev and test set. Even using regularization techniques such as dropouts, this model continues to overfit. Therefore, We are convinced then this model architecture could be improved to achieve better performance. We moved onto convolutional NN models.

3.3 Convolutional Network 1

We then proposed a model with the following architecture: two 2D convolutional hidden layers and two fully connected hidden layers displayed in figure 2. All connections in this model were feedforward with no skip connection. Each convolutional layer had a 2D max pooling with the size of 2x2. All convolution kernels were of size 5x5. We trained this proposed model

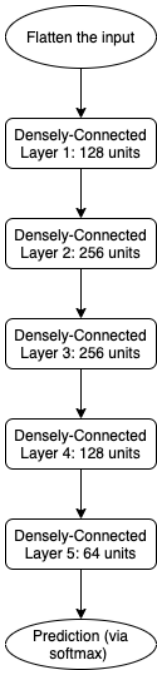

Figure 1: Diagram of the sequential model

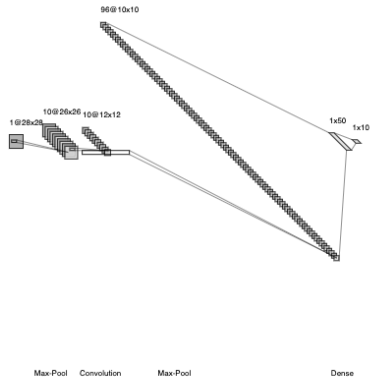

Figure 2: Diagram of the first CNN

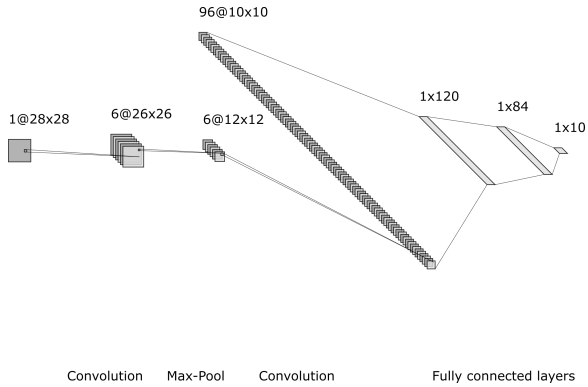

Figure 3: Diagram of the second CNN

with dropout applied to all hidden layers but not the input. This model achieved an 89 percent training accuracy using a batch size of 64 and trained across 10 epochs. However, the test accuracy was only 81 percent.

3.4 Convolutional Network 2

We explored the convolutional neural network architecture further by developing another model with two 2D convolutional hidden layers and three fully connected hidden layers. The connections in this model were also feedforward with no skip connection. We also used 2D max poling with the size 2x2 for each convolutional layer, but we used kernels of size 3x3 for all convolution layers in this model. A diagram of this model is shown in figure 3. We performed a grid-search to tune the hyperparameters of this model. As shown in the grid-search, we decided to use 20 epochs to trained this model as we observed a decrease in validation accuracy due to overfitting when training across more epochs, while less than 20 epochs seemed to lead to underfitting. We also decided to used the adaptative momentum estimation (Adam) gradient as an optimizer since we observed an increase in the cross-entropy loss between epoch when using a stochastic gradient optimizer. We used a batch size of 128 during training. We increased the batch size since it led to an increase in validation accuracy compared to a batch size of 64. To choose the learning rate α, we started with $\alpha = 0.1$ and decreased it by 10^{-1} until we observed a consistent decrease across epoch. Hence, we used $\alpha = 0.001$ and establish by grid-search that $\beta_1 = 0.99$ and $\beta_1 = 0.999$ were the optimal parameters for this model. We tried applying L_2 regularization to deal with overfitting, but we observed that early stopping was efficient enough while adding a weight decay term did not yield a significant improvement in the validation and test accuracy. We obtained a validation accuracy of 98.5% and test accuracy of 98.2% with this model.

4 Conclusion

To summarize, extracted single digits from the sequence of digits in the synthetic MNIST dataset provided and developed three deep neural networks that predicted a single-digit given an image. We also performed some data augmentation to improve training loss and validation accuracy. We could conclude from the results of all proposed models that neural networks tend to overfit. Also, we noticed that the performance of our neural network is strongly dependent on the quality and the quantity of the training data provided to them. By performing data augmentation, we could see a substantial improvement in models' performance (above 3%). We also note that hyperparameter tuning demanded a lot of time and consisted of one of the major limitations we encountered. Even though they are much potential, there are many limitations present. Given the impressive performance of deep neural networks, it is undeniable that they require substantially-sized data and computing resources to train.

5 Appendix

5.1 Hyperparameters and accuracy of the proposed models

Hyperparameters and Accuracy	Sequential model	ConvNets 1 model	ConvNets 2 model (best performing model)
Gradient	Adam	SGD	Adam
α	0.001	0.001	0.001
β or β_1 and β_2	0.9 and 0.999	-	0.99 and 0.999
Weight decay	0	0	0
Epochs	6	10	20
Validation accuracy	94%	89%	98.5%
Test Accuracy	94.5%	81%	98.2%

Table 1: Table of the optimal hyperparameters and accuracy of the network models that were tested

5.2 Statement of Contribution

Krystal contributed to the sequential model, the abandoned multiple NN model and parts of writing the report. Corinne worked on the MNIST preprocessing and data augmentation as well as the second convolutional neural networks. Nghi contributed to the first convolutional neural networks and a part of the introduction.

References

S. Tabik, D. Peralta, A. Herrera-Poyatos, and F. Herrera. A snapshot of image pre-processing for convolutional neural networks: case study of mnist. *International Journal of Computational Intelligence Systems*, 10(1):555–568, 2017.