Comp551 Project 3 Report

Group 51: Linda Cai(260720706), Zixin He(260259017), Iyatan Atchoro(260659320)

December 2020

Abstract

In this project, a deep-learning CNN neural network is constructed to recognize a sequence of handwritten digits in images, analyze them based on 11 "digit classes", and ultimately output in specific data formats. Tensorflow is used to develop the workflow of neural network learning and a set of machine learning algorithms are applied to enhance the model performance through extensive training, validation and tuning stages. The library Keras is also applied to reduce the cognitive loads during the development process. Adam is used as an optimization algorithm to update network weights iterative based in training data. Flexible approaches are used to optimize the hyperparameters of the neural network training and the optimizer, as well as graphs and tables are collected to reflect their effects on the accuracy and efficiency of the model.

1 Introduction

In the world of deep learning, the concept of Convolutional Neural Network (CNN) is widely applied to develop robust algorithms in order to model and solve challenging problems. In this project, such a model is specifically designed to learn the representation and correlation between the input images and the output labels [**?**]. Then, the performance statistics are analyzed and optimal hyperparameters are selected. Plots are generated to further demonstrate the effects of hyperparameter-tuning.

2 Model

The recognition is performed in two parts. First, single digit 16x16 images are extracted from the input. Second, the model performs recognition on each of the digits. The 64x64 input image is first processed with a binary threshold and dilated using a 2x2 kernel. Outside contours are then extracted and used to produce cropped out digits in the left to right order. Contours of insufficient width or height are discarded. The extracted digits are then up-scaled to a 16x16 image while preserving original aspect ratio. This processing is applied to all input images for training, validation and testing sets. In the making of the set of training digits, input images are discarded if more digits were extracted compared to the training target. The trained model recognizes single-digit 16x16 images extracted from the input images using OpenCV in preprocessing. The model uses TensorFlow with GPU acceleration. The output is 10 sigmoid neurons that correspond to the 10 recognisable digits. In total, there are three convolution layers with no max pooling due to the small dimension of the input. This model is smaller than what is typically used for the 28x28 input images from the MNIST dataset. The model is trained on an overclocked golden chip nVidia GeForce GTX 1080. Limiting training time will require finding the right balance between batch size and number of training epochs.

3 Optimizer

Adam is applied to the logistic regression algorithm on the MNIST digit recognition and the sparse categorical crossentropy loss is selected to match the predictive modeling of multi-class classification problem. Adam takes 3 hyperparameters: the learning rate, the decay rate of 1st-order moment, and the decay rate of 2nd-order moment [1]. The Adams optimizer demonstrates numerous benefits to build a robust model, such as computational efficiency, memory efficiency [2]. After tuning the value of Adam's learning rate over the range of 0.0001 (default

value) to 1 depicted in Figure 11, the performance remains stable for a period before the divergence. Such subtle difference also implies that the hyperparameters of adams optimizer may require little tuning due to its nature of intuitive interpretation [3].

4 Results

Figure 1 reports the number of discarded input image samples as result of binary threshold with and without dilation. Figures 2 to 7 report the training and validation accuracies in function of training epoch for the model described in the Appendix. The training and validation accuracies are for the prediction of the full 64x64 input image. If any of the 5 digits in the prediction is wrong, that sample is counted as erroneous. Figure 8 and 9 summarize the data to observe the effects of changing the batch size and Figure 10 examines the training duration for 20 epochs at different batch sizes. Figure 11 and 12 reports the effect of learning rate on accuracy and run time.

5 Discussions

For feature extraction, many of the source images have digits composed of disjointed parts. The detection algorithm would count them as separate contours. The use of a dilation layer with a 2x2 kernel closes the small gaps and improves the training and validation accuracy. Such use of a dilation layer increases both training and validation accuracy and sets the binary threshold at 47, which minimizes the number of discarded inputs. For the training of the model itself, the key hyperparameters are the batch size and the number of training epochs. The goal is to reduce overall computational time while increasing training accuracy. The small batch sizes greatly increase the computational time, but also reach higher training and validation accuracies sooner. The use of larger batch sizes greatly reduces the computational time, but the model would converge later. In the training process, the batch sizes below 32 and above 512 lead to errors during training. In the case of a small batch size of 16, the training consistently crashes with less than 10 epochs. On the other hand, with a larger batch size of 1024, the model performance declines to less than 10% accuracy after 10 epochs. The upper bound and the lower bound determine that the optimal batch size must lie within such range.

6 Conclusion

In summary, a deep learning neural network is built in numerous steps. First, the 16*16 images are extracted from the input and then we proceeded to the recognition. Then, we select and adjust the hyper-parameters to improve the accuracy of the final model. One of the significant findings is that a small batch size tends to yield a better computational efficiency and accuracy. In terms of future improvements, the final accuracy could have been further enhanced by additional preprocessing techniques. Besides these aspects, the model's tendency to underfit or overfit can be further examined and devise corresponding solutions to adjust the bias or variance, such that the strength of regularization corresponds more specifically to the model's need.

7 Statements of Contribution

IA wrote the abstract the introduction and the conclusion. LC wrote the optimizer section. LC and ZH worked on data preprocessing and the CNN model and produced data for Kaggle. LC proofreads the report. ZH produced hyperparameter tuning data and wrote the model description, results and discussion sections the report. IA brought text and charts to overleaf. IA researched convolution layer sizes and debugged the code to explored possible ways to improve accuracy.

8 Appendix

Layer	Parameters
Convolution 2D	32 feature maps, 3x3 kernel, ReLU activation
Convolution 2D	32 feature maps, 3x3 kernel, ReLU activation
Convolution 2D	64 feature maps, 3x3 kernel, ReLU activation
Dropout	25%
Flatten	
Dense	128, ReLU activation.
Dropout	50%
Dense	10, ReLU activation

Hyperparameter Tuning data plottings

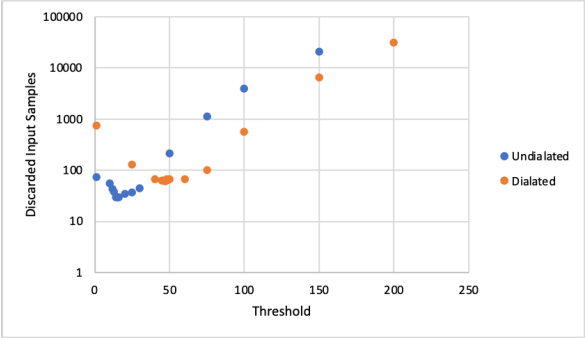

Figure 1: Effect of digit extraction hyperparameters

Plots of training and validation accuracies versus training epoch for select batch sizes

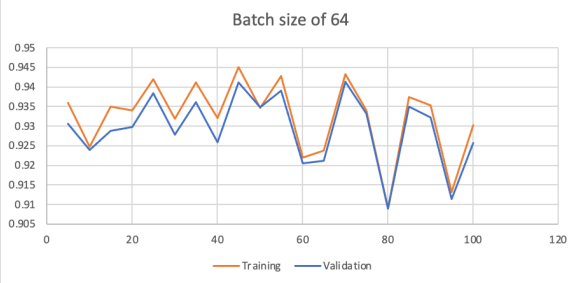

Figure 2

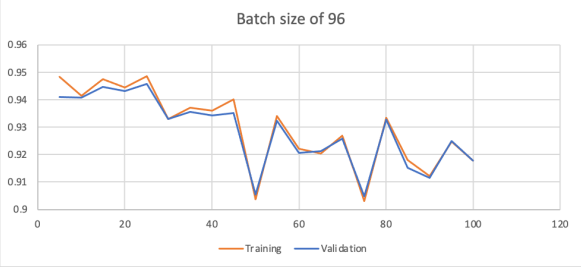

Figure 3

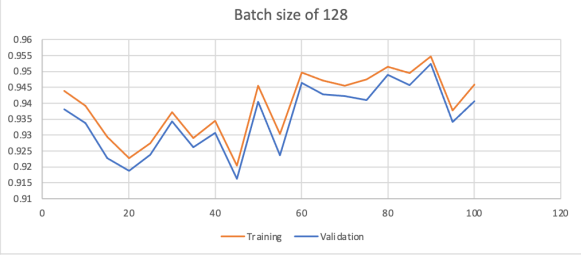

Figure 4

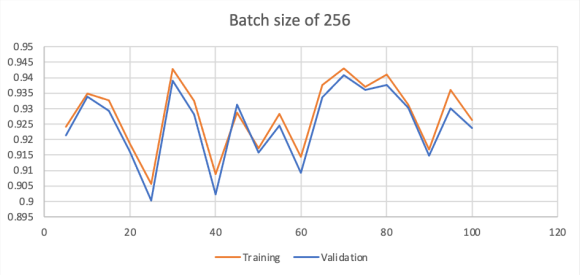

Figure 5

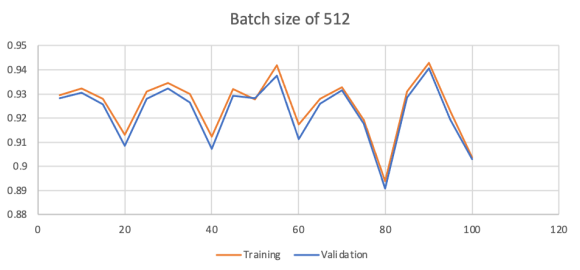

Figure 6

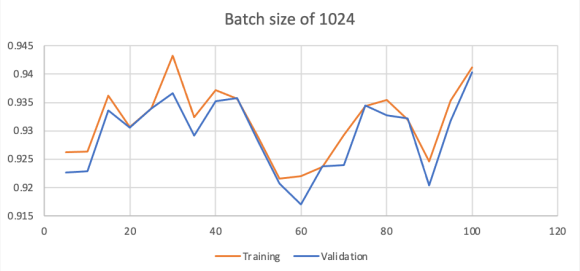

Figure 7

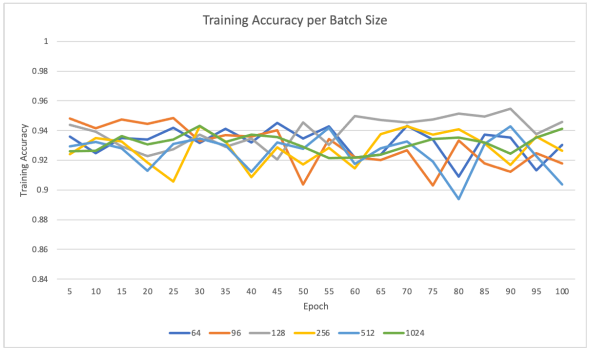

Figure 8

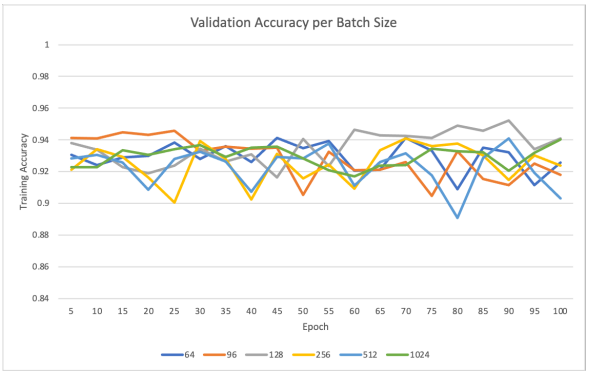

Figure 9

Plot of training duration for 20 epochs versus batch size

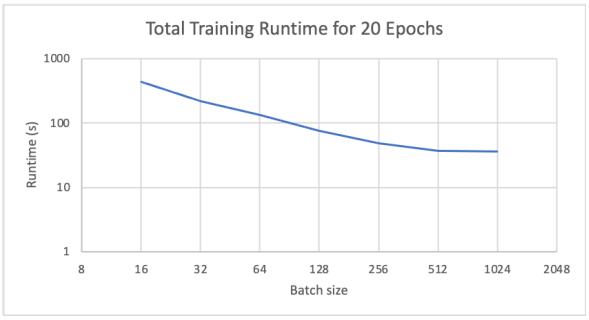

Figure 10: Plot of training duration for 20 epochs versus batch size

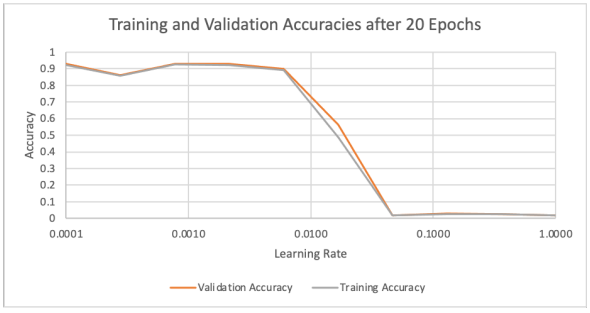

Figure 11

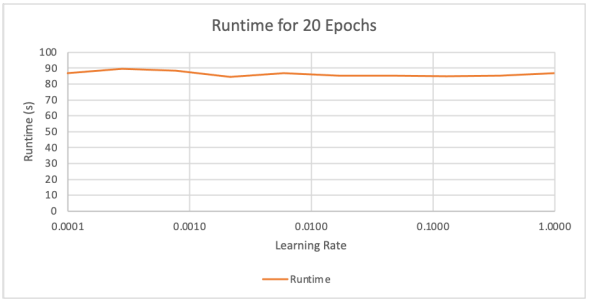

Figure 12

References

[1] D. Vasani, "Momentum, adam's optimizer and more." 2019. [Online]. Available: becominghuman.ai/momentum-adams-optimizer-and-more-7ecf272f4a72

[2] Z. Zhang, "Improved adam optimizer for deep neural networks," pp. 1–2, 06 2018.

[3] J. Brownlee, "Gentle introduction to the adam optimization algorithm for deep learning," 2020. [Online]. Available: machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning

[4] K. Choudhury, "Deep neural multilayer perceptron (mlp) with scikit-learn," 2020. [Online]. Available: towardsdatascience.com/deep-neural-multilayer-perceptron-mlp-with-scikit-learn-2698e77155e

[5] J. Brownlee, "Handwritten digit recognition using convolutional neural networks in python with keras." 2020. [Online]. Available: machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras

[6] K. M. Takase Tomoumi, Oyama Satoshi, "Why does large batch training result in poor generalization? a comprehensive explanation and a better strategy from the viewpoint of stochastic optimization," pp. 1–20, 2018. [Online]. Available: https://eprints.lib.hokudai.ac.jp/dspace/bitstream/2115/71558/3/Tomoumi%20Takase.pdf

[7] M. S. Z. Rizvi, "Cnn image classification: Image classification using cnn." 2020. [Online]. Available: www.analyticsvidhya.com/blog/2020/02/learn-image-classification-cnn-convolutional-neural-networks-3-datasets/