COMP 551 - Project 3 Report

Multi-label Classification of Image Data

Group 3
Zhengwen Fu (260856256), Mohan Zhao (260766884), Chenxuan Zhou (260831229)
December 15, 2020

Abstract

In this project, the classification of image data of multiple digits is investigated using image splitting and convolutional neural network. The dataset used in this project consists of image data of 1 to 5 digits. Therefore, image splitting is introduced to the data preprocessing step to split each digit for classifying each digit individually later. Neural networks of different depths are built and their performance are compared. A neural network architecture consisting of 9 convolutional layers grouped into 3 convolutional layer sets and 1 linear layer is the best performing model for the classification of the preprocessed image data of single digit. The effect on the model with respect to different batch sizes are also compared to pick the best batch size, which is 64 as a result. The model is then trained with the Adam optimizer with 20 epochs to gain the best cross entropy loss. The testing set are then predicted by the trained model. The best test accuracy is 98.9%. The effect on the performance of the model of the different epochs is also studied. It is concluded that a certain number of epochs, 8 epochs in our case, are required for the model to be completely trained.

1 Introduction

Image classification is one of the most well-known applications of machine learning. This application has been further developed since the invention of deep learning [1] algorithms. In this project, the classification of image data of multiple digits is investigated using image splitting [2] and convolutional neural network [1].

The dataset used in this project consists of image data of 1 to 5 digits. Direct classification of the data using a multi-class classifier will result in $11^5 = 161051$ labels. This approach is not only computationally inefficient but also lack scalability, since the number of labels grows exponentially with the number of digits on the image data. To avoid such case, image splitting is introduced in the data preprocessing step. We prepocessed the "modified" MNIST dataset to split each image into a 5 dimensional array of sub-images which could be classified individually.

Using the PyTorch [3] library, neural networks of different depths, inspired by a previous architecture [4], are built and their performance are compared. Then, the best performing model [5] for the classification of the preprocessed image data of single digit can be determined. Moreover, the performance of this best performing model with respect to different batch sizes are also compared in order to pick the best batch size. The model based on this architecture is then trained with the Adam optimizer [6] with 20 epochs to gain the best cross entropy loss [7]. During training, the training and validation losses are recorded to study the effect of different epoch on the performance of the model. Finally, the testing set can be predicted by the trained model.

2 Datasets

2.1 "Modified" MNIST Dataset

The "modified" MNIST dataset from which each image contains 1-5 digits consists of images of handwritten numbers and their labels. It contains 56000 training images, 14000 testing images, and 56000 labels for those training images. Each image has 64 x 64 pixels while that from the original MNIST dataset has 28 x 28 pixels. If the number of digits in an image is less than 5 then the remaining spots belong to the "no-digit" class, whose label is "10". For every image containing 1-5 digits in the training set, a five dimensional label array is given where each label (0-10) is associated with corresponding digit in the image.

2.2 Prepocessing

In order to apply a deep neural network model on single digit dataset, we first convert the images into black-white images represented by binary numbers to easily detect the rectangle contours around each image using the OpenCV [8] library. With the information of rectangular regions, image can be split into 5 sub-images. Before splitting the image based on the coordinates, we also need to handle some boundary issues such as eliminating contours which surround the shadows of some numbers, and reconnecting the split contours of one digit due to some unexpected contour detection problems.

3 Results

3.1 Performance Analysis

3.1.1 Performance with respect to layer depths

We compared different numbers of convolutional layers in each of the three convolutional layer sets. As illustrated in the first graph in Appendix B, more convolutional layers per convolutional layer set generally achieved lower loss. Hence we selected 3 convolutional layers with identical out channel size for each set.

3.1.2 Performance with respect to batch sizes

The second graph in Appendix B , using the most common batch size choices of 4, 16 and 64, shows that the batch size does not have a big influence on the performance of our task. Therefore, we chose the largest batch size as it decreases training time by decreases the number of batches; note that using GPU, the time to compute each batch do not vary much with these three batch sizes.

3.1.3 Performance with respect to epochs

To train our model, we use cross entropy as our loss criterion. As shown in the third graph in Appendix B, both the training loss and validation loss decrease lineally at the beginning. This means the model was improved by the training set rapidly initially. Then, approximately after the 8 epochs, both the validation loss and the training loss are steady. This means the parameters converges, i.e. the gradient optimizer reaches a minimum. Furthermore, the epoch used here is 20, larger than 8, which can represent the best performance.

3.2 Best Performing Model

By selecting best hyperparameter as described in the previous section, we established that our model of choice consists of 3 convolutional layer sets followed by 1 fully connected layer. The 3 convolutional layer sets have convolutional layers of out channel sizes 32, 64 and 128 respectively; each covolutional layer uses max-pooling to reduce the dimensions and dropout to prevent overfitting. Furthermore, each convolutional layer has a ReLU activation function for regularization followed by batch normalization. The Adam optimizer is chosen, which uses adaptive learning rates leading to better convergence. A detailed architecture can be found in Appendix A.

4 Discussion and Conclusion

In this work of digit recognition, we examine a variety of deep neural networks on a multi-label classification task to predict labels for image contains 1-5 digits, after comparing training and validation losses of different networks, we choose the one with 4 layers in total, 3 sets of 3 convolutional layers each and 1 fully connected layer. Besides, we also research on the contribution of hyperparameters to model performance. As an important finding, the performance of the model with respect to different training epochs shows that a certain number of epochs, 8 epochs in our case, are required for the optimizer to converge to a minimum. If not, the model is not completely trained and could not achieve the best performance the setup could provide. This model has well-proved accuracy and robustness as it achieves 98.9% accuracy on Kaggle. Moreover, with the truth that the accuracy of test dataset can not been further improved by tuning hyperparameters on our current model, we suggest a more mature deep neutral network architecture could be used in the future.

5 Statement of Contributions

Chenxuan Zhou contributed to hyperparameter selection, model implementation and report writing.
Zhengwen Fu contributed to model training and report writing.
Mohan Zhao contributed to dataset prepocessing and report writing.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. 1

[2] B. S. Hanzra, "Digit recognition using opencv, sklearn and python," http://hanzratech.in/2015/02/24/handwritten-digit-recognition-using-opencv-sklearn-and-python.html, 2015. 1

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf 1

[4] 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, 1998. 1

[5] C. Deotte, "How to choose cnn architecture mnist," https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist, 2018. 1

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: http://arxiv.org/abs/1412.6980 1

[7] K. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2012. [Online]. Available: https://books.google.ca/books?id=RC43AgAAQBAJ 1

[8] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015. 2

A Table for the best performing model

	Best Performing Model
Test accuracy	0.989
Convolutional layer set 1	• Conv2d(1,32,3,1), ReLu, Batch normalization • Conv2d(32,32,1), ReLu, Batch normalization • Conv2d(32,32,3,1,2), ReLu, Batch normalization • MaxPool2d(2,2) • Dropout(0.25)
Convolutional layer set 2	• Conv2d(32,64,3,1), ReLu, Batch normalization • Conv2d(64,64,3,1), ReLu, Batch normalization • Conv2d(64,64,3,1,2), ReLu, Batch normalization • MaxPool2d(2,2) • Dropout(0.25)
Convolutional layer set 3	• Conv2d(64,128,3,1), ReLu, Batch normalization • MaxPool2d(2,2) • Dropout(0.25)
Linear layer	128 to 11
Loss criterion	Cross entropy
Optimizer	Adam
Epoch	20
Batch size	64

Table 1: The performance and hyperparameters of the best performing model. Note that each convolutional layer is denoted as Conv2d(in channel, out channel, kernel size, padding, stride). Note that MaxPool2d(2,2) means 2 by 2 max-pooling.

B Hyperparameter Tuning