COMP 551 – Applied Machine Learning **Fall 2021**

Mini Project 3 – Multi-label Classification of Image Data Group 81

Junjian Chen 260909101

Zhekai Jiang 260906711

Xichong Ling 260888765

Abstract

In this project, we develop a convolutional neural network (CNN) to identify handwritten digits and letters on images from the combo MNIST dataset. We train the model based on the training dataset as well as the unlabelled dataset. Finally, by bagging several of our well-performed models, the highest accuracy we reached on the public test set on Kaggle is 96.450%.

1 Introduction

Our main goal is to build a CNN and train it with the given training data. We firstly preprocess the format of the data and augment the data via noise reduction and rotation. Then, we build a basic CNN, optimized the architecture, adopted adaptive learning rate and RMSProp optimization, and adjusted hyperparameters such as the number of epochs, batch sizes, filter sizes, etc. to achieve a high accuracy.

2 Dataset

The dataset consists of 30000 labelled training images, their labels, 15000 testing images, and 30000 unlabelled images. Each image has 56×56 pixels with grayscale-encoded values ranging from 0 to 255. Each image shows one English letter and one decimal digit. Each label has 36 bits, the former 10 for digit and the latter 26 for letter, each represented in one-hot encoding.

3 Methodology

3.1 Data Preprocessing

Separation and dimension adjustments. Among the 30000 training images, we use 29000 for training and 1000 for validation. We add a 4th dimension of size 1 as a single channel to fit the Keras-based CNN. We also separate the labels for digits and letters due to the design of our model.

Augmentation. We first tried to reduce the noise of testing images by detecting and removing scattered noisy points, the effect of which is illustrated by Figure 3 in Appendix B. Secondly, each image is rotated by a random degree within a range to generate additional training data.

3.2 Network Architecture

For the construction of the CNN model, we first referred to the baseline standard model proposed by LeCun et al. [1] for handwritten single-digit recognition. Similar to [2], we increased the number of convolution layers and channels to increase the complexity the model can handle. As we need to recognize a digit and a letter, we decided to build a model that produces two separate outputs, following the idea of the one demonstrated in [3]. However, since the predictions of digits and letters are not completely independent (especially for certain digits and letters that largely resemble), we first handle both in a single branch and then separate into two branches near the ending layers, instead of using fully independent branches. Also, as regularization, we add several layers of batch normalization, max pooling, and dropout in between. The overall architecture of our model is shown in Figure 2 in Appendix A.

3.3 Training

The model is trained using mini-batch stochastic gradient descent in multiple epochs. In each epoch, the model will be trained on the training data and validated by the 1000 validation data. We adopt the following strategies to optimize the learning process:

Reducing learning rate. As the stochastic gradients tend to make it hard for the model to fully converge to the optimum, we use Keras's built-in callback `ReduceLROnPlateau` to schedule the learning rate so that it would be reduced

by a factor of 0.5 if the validation accuracy does not increase for two consecutive epochs.

Optimizers. Optimizers also make the step sizes adaptive and helps deal with the problem of oscillations. Our experiments show that RMSProp and Adam make convergence much faster and are the most effective, as shown in Figure 4 in Appendix C. Therefore, we mostly adopt them to train our models.

We also tune hyperparameters such as the number of epochs, batch sizes, filter sizes, etc. The model with the highest validation accuracy will be selected to predict the testing data.

3.4 Utilization of Unlabelled Data

For the unlabelled data, we select the model with the best performance to predict them. Then, the unlabelled images with the predicted labels are reused for training along with the existing labelled images.

3.5 Aggregation

To predict the test images, we select several of our best models, in terms of validation accuracy, to predict simultaneously and decide on the final output based on majority voting.

4 Results

Our vanilla model makes predictions on digits and letters on separate networks and is able to reach an 86.8% accuracy on the public test set. We then attempt to build a single model generating results for both at the same time, leading to a significant increase by 7%. With data augmentation via rotation, the accuracy is enhanced by a narrow edge of 0.4%. Adding a few layers of convolution and regularization slightly increases the accuracy further. Moreover, we investigate the impact of several parameters and find that a batch size of 128 is the best, making the model converge in approximately 20 epochs with a high accuracy while retaining a short runtime for each epoch. Our best single model has validation accuracy growing as shown in Figure 1 (which includes both the case without unlabelled data and later with unlabelled data fed back), eventually converging to 97.5%. It reaches an accuracy of 96.350% on the public test set; and finally, bagging multiple models makes it reach 96.450%.

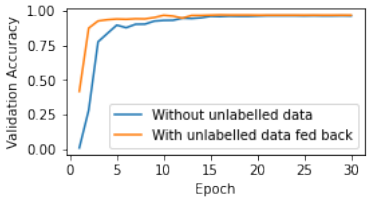


Figure 1: Validation Accuracy with Increasing Epochs

5 Discussions and Conclusions

The following processes play leading roles and might be worth further investigation:

Data augmentation. The performance of the model is expected to enhance with a richer training set. We employ two techniques to extend the training set. One is to make random rotations for each image, which makes the model robust to patterns in different orientations. More advanced techniques, such as future space augmentation and generative modeling [4] could also be used. The other is to make use of the unlabelled data. These unlabelled data generally observe the same distribution of the training set. Therefore, introducing them have an effect of augmenting data. A even more robust approach is clustering and labeling according to those labelled.

Model architecture and learning strategies. Our greatest leap of accuracy from 0.433% to 86.433% was achieved by a more complex model, adaptive learning rate, and RMSProp optimization. This model is adapted from several existing ones for similar problems, and more tuning can be done to further optimize the design specifically for our problem. Reducing learning rate on a plateau allow us to do fine tuning when training stagnates, and a proper optimizer also helps deal with the oscillations. It might be useful to try different settings of them.

Statement of Contributions

Junjian Chen: Data processing, training, report
Xichong Ling: Data processing, training, report
Zhekai Jiang: Model tuning, training, report

References

[1] 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, Nov. 1998, doi: 10.1109/5.726791.

[2] J. Gupta, "Going beyond 99%—MNIST handwritten digits recognition," *Towards Data Science*, May 3, 2020. [Online]. Available: `https://towardsdatascience.com/going-beyond-99-mnist-handwritten-digits-recognition-cfff96337392`. [Accessed: Nov. 27, 2021]

[3] A. Rosebrock, "Keras: Multiple outputs and multiple losses," *PyImageSearch*, June 4, 2018. [Online]. Available: `https://www.pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses/`. [Accessed: Nov. 27, 2021]

[4] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J Big Data*, vol. 6, no. 60, July 6, 2019, doi: 10.1186/s40537-019-0197-0.

Appendices

Appendix A: Architecture of the Convolutional Neural Network Model

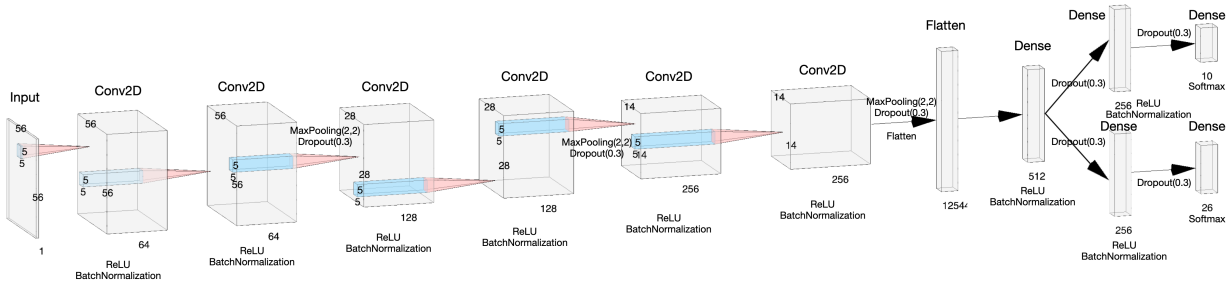
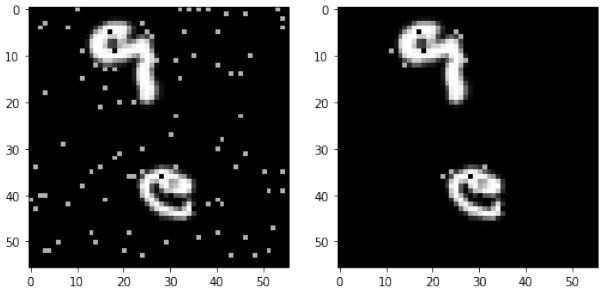


Figure 2: Architecture of the Convolutional Neural Network Model

Appendix B: Effect of Noise Reduction



(a) (b)

Figure 3: (a) Original Image (b) After Noise Reduction

Appendix C: Performance with Different Optimizers

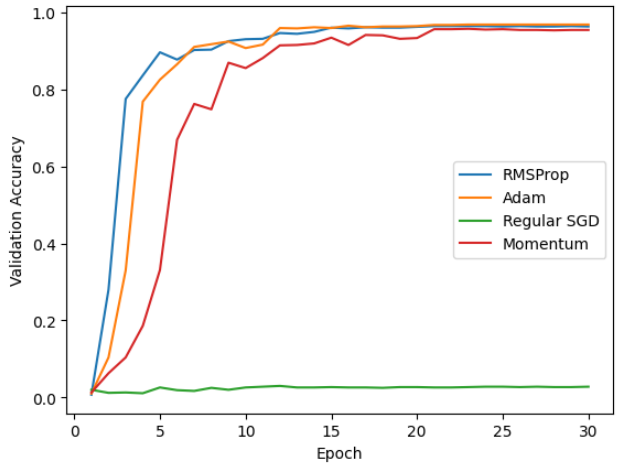


Figure 4: Growth of Validation Accuracy with Different Optimizers, with Batch Size 128