COMP 551 – Applied Machine Learning **Fall 2021**
Mini Project 3 – Multi-label Classification of Image Data Group 35
Yutian Fu Shichang Zhang Angelina Duan
260789775 260890019 260835863

Abstract

In this project, we developed and trained convolutional neural network-based models to classify the MNIST dataset with handwritten digits and letters. Finally, we reached 95.233 percentage of testing accuracy with a VGG16 model predicting digits and a self-developed model predicting letters. We also attempted semi-supervised learning, noise robustness, and bagging average to obtain more powerful and low-variance models.

1 Introduction

In this project, we were expected to build an image classifier to identify handwritten characters from noised pictures. We started with the idea that building two models to separately classify 10 digits and 26 letters and concatenating results to gain the final prediction. By using Keras[1], a CNN model based on VGG16[2] was developed for the digit prediction and a self-developed CNN model with 21 layers was built for letter prediction. During the experiment, it was observed that our models were frequently overfitted. Therefore, data argumentation, noise robustness, increasing model depth, and increasing the dropout ratio were attempted to reduce the variance.

2 Datasets

The MNIST dataset has 60,000 training samples, which are consisted of 10 digits and 26 letters, and 30,000 of them are labeled. We regarded the last 1000 samples as the validation set. We first attempted to reduce the image noises by checking the siblings of a pixel to identify whether it is scattered or not. Then, we converted the pixels which values are lower than the threshold value to 0 so that a useful part of the image would be further highlighted. Through experiments, the threshold value 70 performs best. Our other attempt was to randomly add noises to training data and remove testing data noises so that we were able to gain robust models. Figure 1 to 8 shows the results of our noise processing.

For the unlabeled data, we firstly used our five already trained high-accuracy models to make an average prediction for it. Next, we added these samples with prediction labels into the training set so that we obtained more training samples and were expected to avoid the overfitting problem.

3 Experimentation and Result

We designed two models to separately make predictions for digits and letters to solve the difficulty of multi-label prediction.

For digit processing, VGG16, VGG19, ResNet50, and ResNet152 were tried and we chose VGG16 as our model since its relative higher training and validation accuracy, as is shown in Figure 15. The intuition from The essays [3][4] pushes us to develop the letter model based on the VGG11 network. Since we only had 1 channel for inputs, we removed some convolution and pooling layers to reduce the complexity of the VGG network, which may lead to the degradation phenomenon.[5] By measuring the performance on the validation set, we finally developed the model with 8 convolutions, 4 max-pooling, and 2 dense layers shown in Figure 9.

In the beginning, as shown in Figure 14, we observed that the training accuracy of models quickly

converged to approximately 0.98 but validation accuracy grew slightly. The models were regarded as overfitted.

Since both the width and depth of our model are large, a large dropout ratio was used to randomly abandon some units. As a consequence, on each layer, our models depended on fewer parameters. Figure 12 shows our improvement.

We also considered applying a semi-supervised learning pattern. As discussed in Dataset, we increased the training set size by adding the unlabelled set with prediction labels. As is shown in Figure 17, it generates significant impacts for boosting validation accuracy.

After models were trained, we picked several models with high validation accuracy and made predictions for the test set. We implemented the aggregation bagging by calculating the average value or voting for a mode of the prediction result. As a result, the testing accuracy grew from 93.95 to 94.82.

4 Discussion and Conclusion

VGG16 is commonly used in image recognition because of its small convolution layer structure and large network depth. Although usually increasing in depth results in more powerful model, for small size images such as 56*56*1, the important features of the image may be vanished after multiple convolution and pooling layers. Although VGG16 showed higher validation accuracy compared with other models, a simplified version of it may produce a more pleasant result. We will imitate our letter model developing to build a model with suitable parameters and depth for digit recognition.

Regularization is essential for CNN. There are many methods, such as weight decay, data augmentation, batch normalization, dropout and etc. In our experiment, we implemented data argumentation, averaging and dropout rates increasing to improve our model performances.

Data argumentation for unlabelled data was already discussed above. For the dropout rate, it provides immediately the magnitude of the regularization, which is adaptive scaled by the inputs.

We observed that CNN with a fully connected layer is powerful but can be easily overfitted on the dataset because of the large number of parameters in the fully connected layer. Therefore, we used a dropout rate of 0.6 for our digit model and 0.25 for the letter model to overcome this problem. The higher dropout rate of the letter model was due to the larger number of units in each layer.

Additionally, we implemented bagging by averaging or voting for a mode of prediction. Every model is biased. The average or mode of those predictions yielded a correction for the bias of a single prediction. Hence, we will obtain a more general model. That's why our testing accuracy increased dramatically when calculating their average or mode.

Theoretically, by adding noise points to the training images, we would train noise-robust models. Possibly due to the small size of the image, the adding noise points with 1 pixel size are relative large and made the training data significantly different from the testing images so we received unpleasant score for the attempt. In the future, we consider to use algorithms such as Gaussian blur, which increases the training difficult but does not alter the image dramatically. Since we often obtain noise-sensitive models, for further improvements, NERNet is regarded as an alternative way to be used to effectively reduce noise on images. The architecture of it, the noise estimation module, and the noise removal module can produce more clear and sharp images. [7]

Statement of Contributions

Yutian Fu: Data organization, report writing

Angelina Duan: Model training and testing, report writing

Shichang Zhang: Model training and testing, framework build-up, model implementation

Reference

[1]Chollet, F. & others, 2015. Keras. Available at: https://github.com/fchollet/keras. Citation in Bibtex format.

[2]Chollet, F. & others, 2015. Keras. Available at: https://keras.io/api/applications/vgg/.

[3]Duan, C., Yin, P., Zhi, Y., & Li, X. (2019). Image classification of fashion-MNIST data set based on VGG network. In Proceedings of 2019 2nd International Conference on Information Science and Electronic Technology (ISET 2019). International Informatization and Engineering Associations: Computer Science and Electronic Technology International Society (Vol. 19).

[4]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[5]Ghosh, S., Shet, R., Amon, P., Hutter, A., Kaup, A. (2018, April). Robustness of deep convolutional neural networks for image degradations. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2916-2920). IEEE.

[6]Kaur, T., Gandhi, T. K. (2019, December). Automated brain image classification based on VGG-16 and transfer learning. In 2019 International Conference on Information Technology (ICIT) (pp. 94-98). IEEE.

[7]Guo B, Song K, Dong H, Yan Y, Tu Z, Zhu L (2020) NERNet: noise estimation and removal network for image denoising. J Vis Commun Image R 71:102851

Appendix

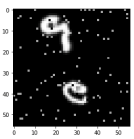

Figure 1: Before processing

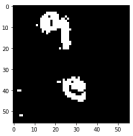

Figure 2: After processing

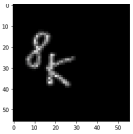

Figure 3: Before processing

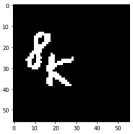

Figure 4: After processing

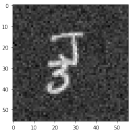

Figure 5: Before Processing

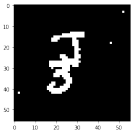

Figure 6: After Processing

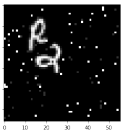

Figure 7: Before adding noise

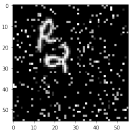

Figure 8: After adding noise

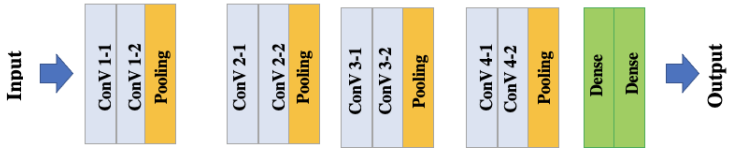

Figure 9: Letter Model

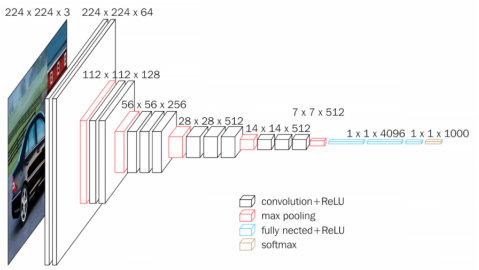

Figure 10: VGG16

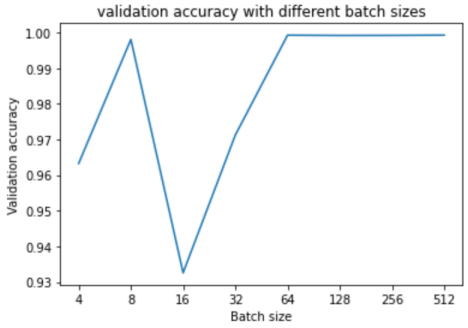

Figure 11: Accuracy of different Batch_size

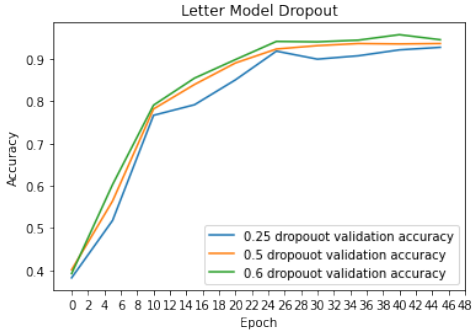

Figure 12: Accuracy of different Drop-out Rate

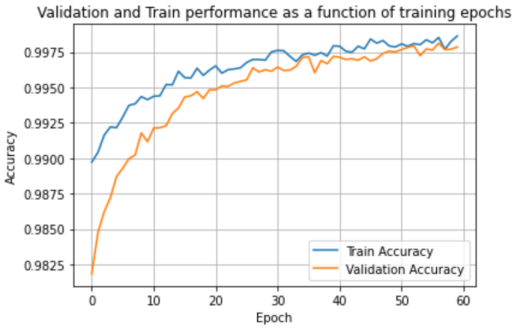

Figure 13: Accuracy of different Epoch

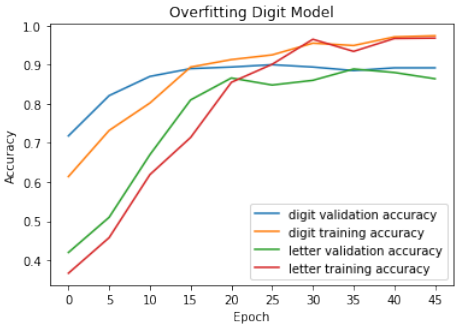

Figure 14: Overfitting Models

# of Epoch	VGG16	VGG19	ResNet50	Inceptionv3
1	0.892	0.858	0.146	0.1
2	0.918	0.892	0.172	0.094
3	0.918	0.91	0.21	0.134
4	0.92	0.922	0.22	0.145
5	0.92	0.934	0.31	0.245
6	0.936	0.92	0.333	0.234
7	0.93	0.91	0.4	0.22
8	0.938	0.93	0.457	0.34
9	0.902	0.94	0.525	0.453
10	0.938	0.922	0.52	0.44
11	0.934	0.916	0.6	0.473
12	0.922	0.926	0.635	0.542
13	0.926	0.92	0.66	0.546
14	0.936	0.922	0.75	0.67
15	0.94	0.93	0.754	0.75
16	0.942	0.925	0.771	0.753
17	0.953	0.923	0.833	0.84
18	0.95	0.92	0.861	0.835
19	0.958	0.926	0.912	0.843
20	0.96	0.926	0.916	0.875

Figure 15: Different models with accuracy

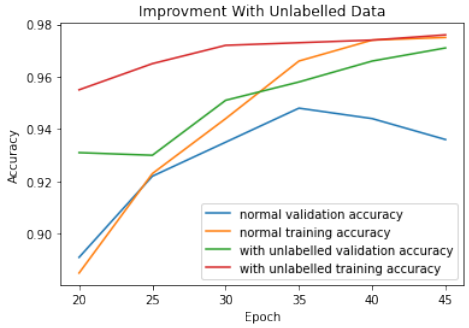

Figure 16: Improvement with Unlabelled Data