Neural Network in Analysing Modified MNIST Dataset

Yiheng Lu, Zijin Nie, Yichao Yang

***Abstract*—This work applied Convolutional Neural Networks (CNNs) in analyzing the modified MNIST dataset. The goal was to recognize the largest numerical value of numbers in a picture which contains 3 MNIST handwritten numbers and a random background. In this project, we attempted different ways of image preprocessing, including thresholding, normalization and centering digits, then we compared the performance of each method. We implemented and compared three variations of CNNs model: LeNet, AlexNet and VGG. The overall performance of VGG is the best. The validation accuracy of VGG for 33500 training samples and 16500 validation samples is around 0.946 with 0.22 validation loss. The final accuracy with 50000 training samples and 10000 testing samples is 96.333%, which is our highest result in Kaggle competition as Group 60.**

I. Introduction

A. Dataset background

MNIST database is a collection of handwritten digits first introduced by LeCun in 1998. MNIST is commonly used for training in image processing and machine learning, so far as today it is considered a benchmark for prototyping many newly created pipelines, particularly in image recognition and classification.

B. Task

In this project, we are given a set of MNIST with each image containing 3 handwritten digits, and our task is to apply preprocessing methods and neural network models to to recognize the digit with the largest numerical value in each image.

In order to achieve our goal, several popular image classification CNN models including AlexNet, LeNet and VGG were tested through our validation pipeline. We compared their validation accuracy and discovered the best-performed one.

We also investigated the effect of image preprocessing techniques. Normalization and binary thresholding were used to eliminate background interference. The digits in images were cropped out and put into a new dataset. In the end, for each model, we trained on two different datasets separately: the raw image set and the dataset after preprocessing.

C. Important findings

We tried to preprocess our images to make them more ”readable”, however, sometimes this is also a double-edged sword. For example, we created a new dataset ”modified image set” based on the original dataset. This did not help a lot for two reasons: one is that some useful information lost during processing, the other reason is that too much simplified images may cause serious overfitting when training CNN models.

As for our models, the overall performance of LeNet and VGG model are good, but in some sense they are also ”sensitive”. During our testing for both LeNet and VGG model, even a single filter size value changed may cause the total training diverge, the structure of layers in CNN model is reasonable and may varies a little depending on the input.

II. Raleted Work

”Recent Advances in Convolutional Neural Networks”[1] introduced the improvement of CNN, especially LeNet model from different aspects such as layer design and activation function. We referred to some strategies of layer design to improve our CNN model. By adding some certain layers on the original model, more complicated images could be handled more accurately.

Classification tasks using normal VGG models usually face accuracy bottleneck. ”DeepID3: Face Recognition with Very Deep Neural Networks”[2] introduced a more complicated task, in which

they build their own VGG model to perform face recognition. This gave us inspiration to tried out with different layer structures to improve the performance. By changing parameters of hidden layers, VGG model becomes more suitable for handling image with complex features and does better for our specific task[3].

Convolutional Neural Networks (CNNs) were applied to train the model and to predict MNIST datasets. The vector of weights connected to the neurons of the previous layer, so that the value of a neuron was computed as a weighted sum of the neurons from the previous layer. Neurons can apply an activation function to introduce non-linearity[4].

III. Dataset and Setup

A. Size information

The training dataset used in this task include 50000 images with MNIST handwritten numbers. The testing dataset include 10000 images with MNIST handwritten numbers. The labels for training are the largest numbers that appear in the training images.

B. Preprocessing

We attempted several approaches to remove random noise appeared in picture background and to optimize the readability of the images.

The first approach is using binary thresholding (imported from cv2 package), which suppresses all pixels which grey scale less than 250 to 0, and preserving pixels with values above that. Visually, the background noise is turn to all black and our target digit remains white(pixel value=255).

Furthermore, We managed to split each raw image into 3 separating images which contains only one hand-written digit. The size of each digit picture is 40px * 40px. As shown below, the 3 numbers in raw image may situated at random corner of the 128x128 image, while after this step, every single number is extracted. In our further model-training step, we combine these 3 separate images together into a new image with size 120x120. It is easy to observe from the images below that those randomly located numbers are now permuted in order. In the following sections in this report, we call these "cropped and merged" image set as "modified image set". This indeed improved the performance for our "LeNet" model, both for the speed of convergence and the accuracy obtained.

We also normalized the value of input image pixels by dividing grey scale value by 255.0.

Fig. 1: raw - threshold - modified image

IV. Proposed Approach

A. Model Validation

In order to systematically trained the models and attain best parameter of models, we followed a validation pipeline. We split training data of 50000 entries with 75% of training set and 25% of validation set. Each test performs 50 epochs. For the test of a specific parameter, a set of values are fed into the predictor and best value is chosen through validation. We also compared raw data and "modified image set" by training and validating their validation accuracy on both LeNet and VGG.

B. models

1) AlexNet:

AlexNet is a CNN model designed by Alex Krizhevsky. AlexNet contained eight layers. The first five layers were convolutional layers followed by max-polling layers. The last three layers of AlexNet are fully connected. However, the AlexNet we implemented was overfitting. The difference between training accuracy and validation accuracy is approximately 0.7.

2) LeNet:

LeNet is one of the most popular model used for recognition of handwriting images, so we first tried this model. By following the introduction and basic principle of LeNet-5 (by Yann LeCun in 1990's), we implemented our own model, which consists of 5 layers: **[con2D(64,5x5) – maxpooling(2x2) – con2D(128,5x5) – maxpooling(2x2) – full**

connected layers]. The overall performance is not satisfying(at most 50%).

Then as mentioned in Related Works section, we referred to the improved LeNet model introduced in "Recent Advances in Convolutional Neural Networks"[1]. We accordingly changed our model by adding Convolution and MaxPooling layers and adding Dropout layers to reduce overfitting cases. New Lenet model contains 11 layers, with the number of 5x5 convolutional filters increasing from 64 to 128. The breif structure is shown:
[con2D(64, 5x5, 'relu') – maxpooling(2x2) – con2D(64, 5x5, 'relu') – maxpooling(2x2) – con2D(128, 5x5, 'relu') – maxpooling(2x2) – con2D(128, 5x5, 'relu') – maxpooling(2x2) – con2D(256, 5x5, 'relu') – maxpooling(2x2) – full connected layers]

3) VGG:
VGG is a CNN model proposed by K. Simonyan and A. Zisserman from the University of Oxford. It makes the improvement over AlexNet by replacing large kernel-sized filters with multiple 3x3 kernel-sized filters one after another [5].

We followed the basic feature of VGG model and implemented it. The first a few attempts for our VGG model is not good, the accuracy stayed stably at approximately 26%. Then we tried to add BatchNormalization() as new layers between CONV2D and maxpooling. The new model has a lower speed of convergence than LeNet and AlexNet but has higher accuracy.(it converges at about 94%-95% after 30 epochs). This is also our final model for predicting test image set.

Similar to previous models training, we compared the performance of this model by using raw images and threshold images as input training set separately. It turns out that VGG model performed better on raw image set. Our conclusion is that VGG can handle very complicated images (e.g. many boundaries, objects and noisy point contained in an image) because of its structure: using multiple 3x3 filters instead of one or two 5x5/7x7 filters, it reduces a lot of computation complexity. Moreover, it is studied [6] that two 3x3 filters have the same receptive field as one 5x5 filter, and three 3x3 filters act as one 7x7 filter. (figure 2)

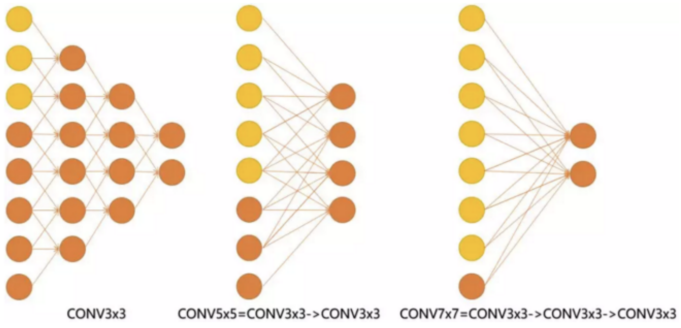

Fig. 2: multiple 3x3 filters behaviour

By doing convolution layer after layer, some noisy points would be well ignored and also largely reduce the loss of information at the same time. However, during preprocessing procedure data may lose some useful point from the perspective of the model.

C. Identifying hyper-parameters

For optimizors, we chose Adam() since it is widely used and treated as the benchmark of CNN optimizer. SGD() was tried, however it caused an unstable accuracy fluctuation. Also we used the grid search(we tried on 32, 64, 128, 256, 512) to seek the best batch size of VGG model.

V. Results

A. AlexNet

The AlexNet model we implemented was overfitting. After 20 epochs, the accuracy for training was over 90% but the accuracy for the validation set was only over 20% for both preprocessing approaches. So we did not make further evaluation on this model.

B. LeNet

For LeNet model, we trained on two dataset separately: raw image after thresholding and normalizing, and "modified image set". It turns out that the "modified image set" has a faster speed of convergence (after 6 epochs) and higher overall accuracy for original LeNet model (final accuracy on test set is 83.333% from kaggle leaderboard). The improved model training on raw image set converged at about 94% after 15 epochs, which

performed better than that on ”modified image set”.

It is interesting that when we tested two LeNet models(one original and one improved), the original LeNet model performed better than improved LeNet model on raw image set, while improved LeNet model has a better result than original one on ”modified image set”. The following are two graphs show the comparison of accuracy on two datasets by using two LeNet models separately.

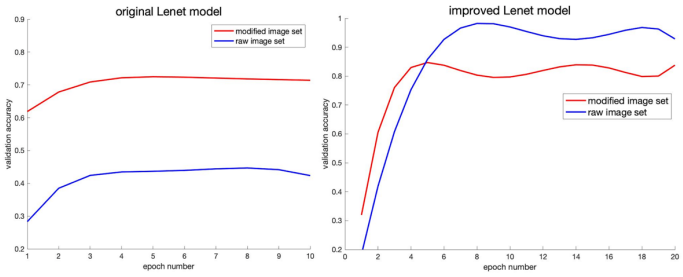

Fig. 3: two LeNet models accuracy comparison

We can see from the graph that the best performance occurred when we trained on raw image set using improved LeNet model (final accuracy is 92.33% from Kaggle leaderboard). While training on raw image set is much more unstable than training on ”modified image set”.

As is said above, when training on the raw image set, we found that the LeNet model is very sensitive that even a single parameter of one layer changed may result in divergence (for all epochs the accuracy are the same, which is around 26.67%), while the ”modified image set” can always guarantee that it converges.(normally between 70%-80%). This could also be an investigation topic for future study.

C. VGG

We found that with preprocessing approaches, the validation accuracy was not stable. The huge gap between training accuracy and validation accuracy indicated preprocessing would case overfitting in the VGG model.

1) Without Preprocessing Approach: During our evaluation of VGG model, we found that the best performance of the model occurred when we trained on the raw image set, with normalization (simply divided all pixel values by 255). Here are the performance of accuracy of training set and validation set, and the validation loss during the model training.

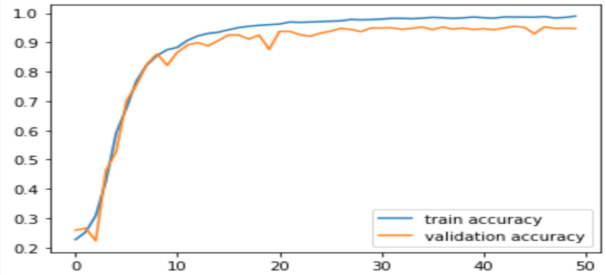

Fig. 4: Accuracy of training set and validation set for VGG

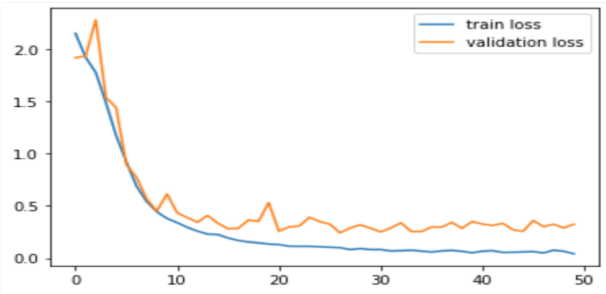

Fig. 5: loss of training set and validation set for VGG

From the graph we can see that the accuracy and validation loss converges at the accuracy 94% after about 30 epochs. The highest accuracy for VGG model reached 96.333% from kaggle leaderboard. Slight Overfitting occurred for the last 20 epochs, where the training accuracy is slightly higher than the validation accuracy. This is also an important issue that limits our final accuracy, which can be an investigation direction in the future neural network study.

2) Preprocessing with Threshold: We applied the threshold approach to the original images. The pixels with values under 250 was changed to 0. The running time of each epoch was significantly lower than the running time of each epoch before preprocess. However, as figure 6 indicated, thresholded value produced significant fluctuation on validation accuracy and lost. The accuracy jumped between 20% and 80%. We suspect that 0 value causes numerical instability during computation, resulting in sharp change feature vectors and huge errors.

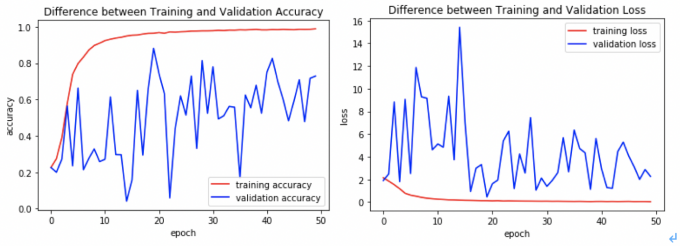

Fig. 6: The comparison of accuracy and loss for VGG with threshold

3) Batch Size: Batch size is one of the important hyper parameters that influences the behaviour of our VGG model, so we did the comparison of final accuracy and average runtime (where epoch=30) for different batch sizes. Here are the visualization of our evaluation:

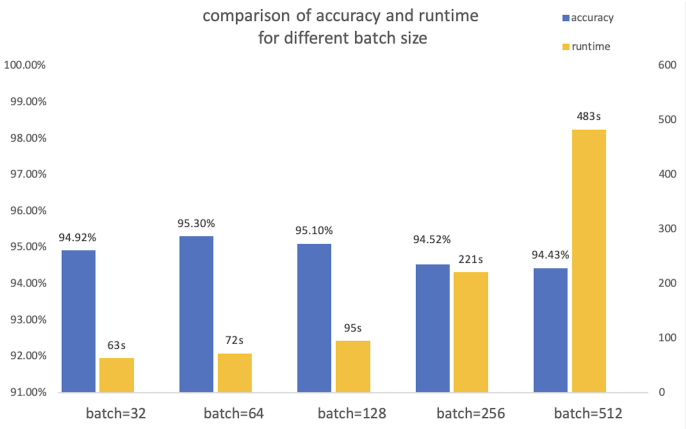

Fig. 7: The comparison of accuracy and runtime for different batch size of VGG model

We can see from the figure 7 that the runtime increases as we add up the batch size, while the best accuracy occurred when batch size is 64 and 128. We submitted the result for both cases and 128 size shows a better accuracy from kaggle leaderboard. So we choose batch size parameter with value 128 for final prediction.

D. Comparison between models

Since AlexNet model performed much worse than the other two, we did not make further evaluation about it. Here in figure 8 we compare the LeNet model and VGG model with respect to their runtime and validation accuracy (We choose the best performance for each model) [Fig.8]

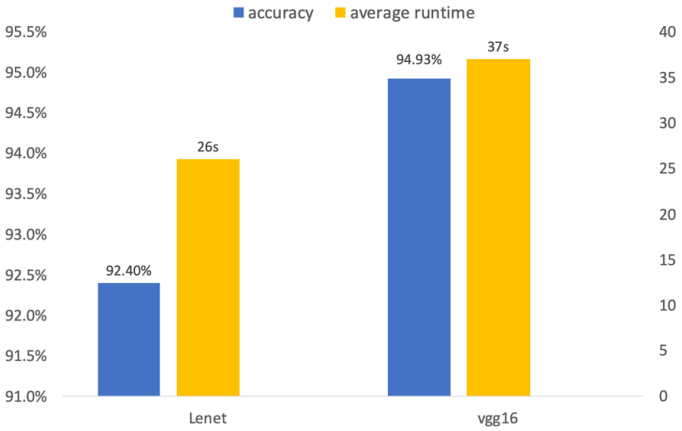

Fig. 8: The comparison of accuracy and runtime for LeNet and VGG model

Although the VGG model shows a better behaviour than LeNet model, it has much higher runtime, and the converge rate is lower as well: LeNet model may converge within 15 epochs while VGG model needs to take 20-30 epochs to converge. This is reasonable since VGG model use more layers with small kernel filter size, and the number of layers affect the runtime and type of filters in convolution affects the rate of convergence.

VI. Discussion and Conclusion

Preprocessing can improve accuracy, but too much simplified preprocessing would result in loss of information and serious overfitting. We should balance the work on preprocessing and model construction.

The multiple variances of one learning model can have different performances. After trying several models, we found that VGG contributed the highest performance among all the models in CNNs. The suitable hyper-parameter adjustment would increase the performance of the model. In modern machine learning, the change in different kinds of recognition problems requires continuous attempts.

Overfitting is one of the common issues that happened in our experiment. We tried several ways to reduce the cases (e.g. reduce preprocessing, add dropout layer) but the effect is limited. How to effectively avoid overfitting should be one of the directions for future investigations.

VII. Statement of Contribution

1) Yiheng Lu (260789862), yiheng.lu@mail.mcgill.ca, code, test, report
2) Zijin Nie (260787728), zijin.nie@mail.mcgill.ca
3) Yichao Yang (ID: 260764629) yichao.yang@mail.mcgill.ca preprocessing, model construction and testing, report writing.

VIII. Appendix

References

[1] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[2] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *CoRR*, vol. abs/1502.00873, 2015. arXiv: 1502.00873. [Online]. Available: http://arxiv.org/abs/1502.00873.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[4] S. Tabik, D. Peralta, A. Herrera-Poyatos, and F. Herrera, "A snapshot of image preprocessing for convolutional neural networks: Case study of mnist," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 555–568, 2017.

[5] M. ul Hassan. (Nov. 20, 2018). Vgg16 – convolutional network for classification and detection, [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.