COMP 551 MiniProject 3: Modified MNIST

Shuhong Shen
shuhong.shen@mail.mcgill.ca

Yiran Mao
yiran.mao@mail.mcgill.ca

Jingyuan Wang
jingyuan.wang2@mail.mcgill.ca

March 18, 2019

Abstract

Object recognition is one of the most significant functionalities in image processing and computer vision. This project aims to develop a classification model to identify the MNIST digit with the largest bounding box in grayscale images that contain multiple of them. By directly feeding binarized training images into an ensemble of stacked ResNet-18 using 10-fold cross validation split, a Kaggle test accuracy of 96.9% was found after fine tuning multiple model hyperparameters such as learning rate, momentum, batch size and pooling layer size. It was found preprocessing the images by cropping out the largest digits using *findcontours* in opencv library and training using established convolutional neural networks (CNN) could not yield satisfactory predictions due to a roughly 6% misclassification rate in cropping. ResNet was observed to significantly outperform AlexNet and LeNet when multiple objects are present in the images. ResNet with deeper structures such as ResNet-101 or ResNet-152 were often found overfitting to the training set.

1 Introduction

Computer vision has been the focus of research in recent advancement in deep learning. Models with high object recognition accuracy and short computation time are highly desired, which need to be validated first using established database. MNIST is one of the most famous databases with handwritten digits and is commonly used to validate the performance of developed machine learning models[1]. The task of this project is to identify the largest MNIST digit with the largest bounding box, from images that contain multiple differently sized and oriented digits. The images are grayscale and 64×64 in size, with 40,000 samples as the training sets with labels and another 10,000 samples as the unlabelled test set that is used for Kaggle submission for performance evaluation. Initially, we took the approach of preprocessing the image to identify the largest digit by *ndimage.find_objects* in scipy[2] library and later train established CNNs such as LeNet[3] and AlexNet[4] using cropped single digit images. However, high misclassification rate during preprocessing was found, at roughly 6%, due to inconsistent orientations and overlapping/connected digits. This led to a low prediction accuracy of 93.0% using single AlexNet. Then, it was decided to train established CNNs including recently successful ResNet[5] directly with the raw images. It was found that with an ensemble of stacked ResNet-18 with 10-fold cross validation after fine tuning the hyperparameters, a Kaggle test accuracy of 96.9% was achieved. Deeper CNNs like ResNet-101 and ResNet-152 were also tested but a significant overfitting to training set was witnessed due to an overly large number of parameters.

2 Related Work

MNIST dataset has been very well studied in the past decades, with LeCun et al.[3] implemented both traditional classifiers like linear, K-nearest neighbors, support vector machine (SVM) and neural networks

(NN), and proposed high performance CNN-based LeNet-5 with deeper structure than previous versions achieving an 0.8% error rate. Later, Lauer et al.[6] used trainable feature extractor + SVM to obtain a lower error rate of 0.54%. Recently, Ciresan et al.[7] proposed a committee of CNNs with width normalization preprocessing and reported an error rate of only 0.23% on MNIST. More CNNs were proposed in response to the ImageNet competition, with Krizhevsky et al.[4] proposed 8-layered AlexNet winning the 2010 competition. More importantly, they employed training using GPUs and largely accelerated the speed of computation by over 10 times. Most recently, ResNet developed by He et al.[5]achieved superhuman performance using residual learning. Residual functions are used to deliberately skip some layers to avoid gradient vanishing and reduce the number of effective parameters. They have reported competitive top-5 error rate of 7.76% versus 9.33% using VGG-16 net[8]. Thus, its Keras[9] implementation will be used in this study.

3 Dataset and Setup

The dataset consists of 40,000 64×64 grayscale images with each containing multiple MNIST digits with different shapes and orientations for training and their labels of the largest digit in the image. Another 10,000 similar images make up the test set that will be used for Kaggle submission for performance evaluation. Initially we preprocessed the images by first binarization and cropping out the digit with the largest bound box via *find_objects* in scipy library but failed to achieve satisfactory prediction accuracy. Then in our proposed model, the images are simply thresholded and used for training and testing without any other preprocessing apart from reading from .pkl files and formulating into the correct input format.

4 Proposed Approach

The foundation of our attempted approaches is creating different feature extraction pipelines and then using powerful CNNs to recognize the digits.

We implemented two image preprocessing approaches. The first one is to threshold the images into binary channel, and to use the function *ndimage.find_objects* from scipy package to locate all connected components. Then we identify the digit with the largest square bounding box. Once the image of largest digit is extracted, they are then scaled to the same size of 28×28 before feeding into CNNs for training. For the second one, the raw images are only binarized where all pixels with the highest intensity were recognized without finding the largest digit. This approach is able to eliminate the information loss that occurs during the preprocessing in the first pipeline.The processed images were stored in .txt files for the convenience of future operation.

We tested three CNN models in total, LeNet, AlexNet and ResNet. The layer design of our models closely followed [3], [4], [5] with some network hyperparameters fine-tuned with grid search method, including batch size, pooling layer size, kernel size, learning rate. In addition, different gradient optimization functions were also investigated for our CNN models. Loss function was set to cross-entropy for classification task.

Initially we used a hold-out validation to evaluate performance of each model by spliting up input data into training and test set with *train_test_split* from sklearn package[10]. To further improve our model, ensemble technique was used to collectively lower prediction variance by stacking differently initialized ResNets trained with K-fold cross validation implemented. Ensemble prediction on test set images is accomplished by voting to the category with the highest probability. The training process was run on GPU using CUDA toolkit[11] for accelerated computation using high-level neural networks framework *Keras* with *tensorflow* [12] backend.

5 Result and Discussion

5.1 Model Setup and Performance

To give a comprehensive overlook of model performance, we conducted for each model the two feature extraction pipelines with the first one preprocessing input and giving single digit images and the second one directly showing raw binarized images. Some of the important hyper-parameters in each CNN model are tuned carefully. Through comparison, the model with best performance is selected and further optimized.

5.1.1 LeNet

LeNet[3], the most fundamental and inspiring CNN model, consists of two sets of convolution layers and sub-sampling layers, followed by two fully-connected layers and finally output probability distribution of each class through a softmax layer. We investigated impact of different hyper-parameters as well as gradient descent optimization algorithms on CNN models and fine tuned them to achieve a model with the best

performance. The accuracy of each model on the validation set is shown in Table 1 with pipeline 1 and 2 representing feature extraction with and without preprocessing, respectively. The combination of the first pipeline and RMSprop optimizer gives the highest accuracy.

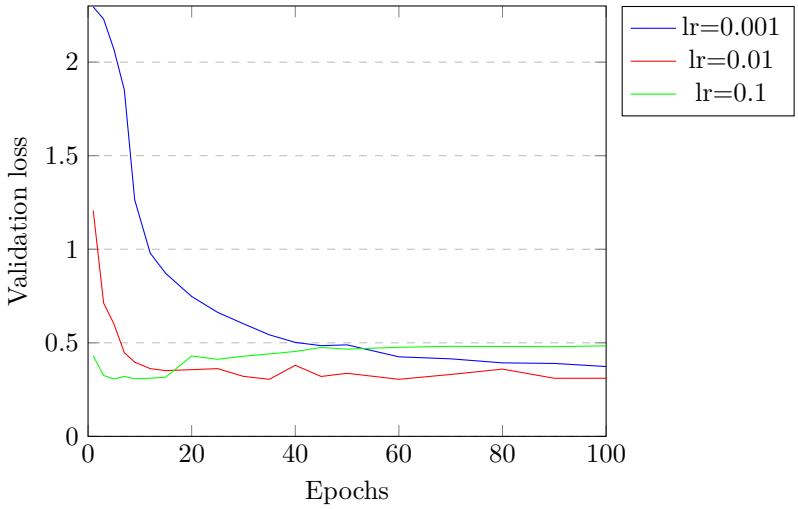
Feature	SGD(lr = 0.001)	SGD(lr = 0.01)	SGD(lr =0.1)	Adam	RMSprop
pipeline 1	91.05%	91.08%	92.87%	93.32%	**93.61%**
pipeline 2	43.27%	37.84%	50.01%	41.51%	46.06%

Table 1: LeNet model accuracy for pipeline 1, 2

Parameters of Adam and RMSprop are default setting of Keras API with both learning rate=0.001.

It is discovered that for LeNet-5, result of pipeline 1 is distinctly superior to pipeline 2 since the simplified structure of LeNet cannot accomplish such complex task of capturing the largest digit in multi-digit images and can only reach an accuracy of 50.01% at best over all attempts. According to Figure 1, an overly small learning rate makes gradient descent extremely slow and can barely converge in 100 epochs while an overly large one starts with rather low validation loss but may oscillate around the local minima and even diverges(increasing loss on validation set). Consequently, model with learning rate=0.01 outperforms the other two from our investigation.

Figure 1: Validation loss on epochs with pipeline 1



Choices of different optimizers is also notable during training process. We have observed that Adam and RMSprop have slight better performances compared to general SGD function. SGD(Stochastic Gradient Descent) is a fine gradient descent function which performs a parameter update for each training example and computes faster with large datasets. However, SGD has a drawback that when facing areas where surface is much steeper in one dimension than others, it will oscillate in the slopes and make slow progress[13] whereas Adam and RMSprop can solve this problem by adjusting learning rates to recent gradient values.

5.1.2 AlexNet

AlexNet contains eight layers in total. The first five are convolution layers with some max-pooling layers inserted in between and then followed by three fully-connected layers. It uses the non-saturating ReLU activation function, which shows improvement on model performance over tanh and sigmoid. Likewise, we ran the same experiments on Alexnet as on LeNet by tuning hyperparameters and choosing optimizers. The best performance of model with each pipeline is shown in Table 2. It is found that AlexNet performance with pipeline 2 has been improved compared to LeNet but precedingly cropping the largest digit picture is still helpful to our model.

Feature	Validation Accuracy	Optimizer and Hyper-parameters
pipeline 1	**94.06%**	RMSprop(lr=0.001, rho=0.9)
pipeline 2	72.78%	Adam(lr=0.001, beta_1=0.9, beta_2=0.999)

Table 2: AlexNet model accuracy for pipeline 1, 2

5.1.3 Resnet

ResNet is a residual net with a depth of up to 152 layers while a low complexity is maintained [5]. We tried 3 types of ResNet depending on its depth: ResNet-18, ResNet-101, ResNet-152. Hyperparameters in this section become larger in amounts and due to the expensive computation cost, we made limited attempts to tune batch size and pooling size. With batch size equal to 32, pooling size equal to 2, we achieved our best model with pipeline 2 and learning rate of optimizer Adam set to 0.001. However, deeper ResNet has not performed well as we expected because the sheer volume of parameters in these models leads to high variance and severely overfits to the training set.

Feature	ResNet Depth	Validation Accuracy
pipeline 1	ResNet-18	91.80%
pipeline 2	ResNet-18	**94.89%**
pipeline 2	ResNet-101	89.31%
pipeline 2	ResNet-152	85.87%

Table 3: ResNet model accuracy for pipeline 1, 2

5.2 Performance Comparison

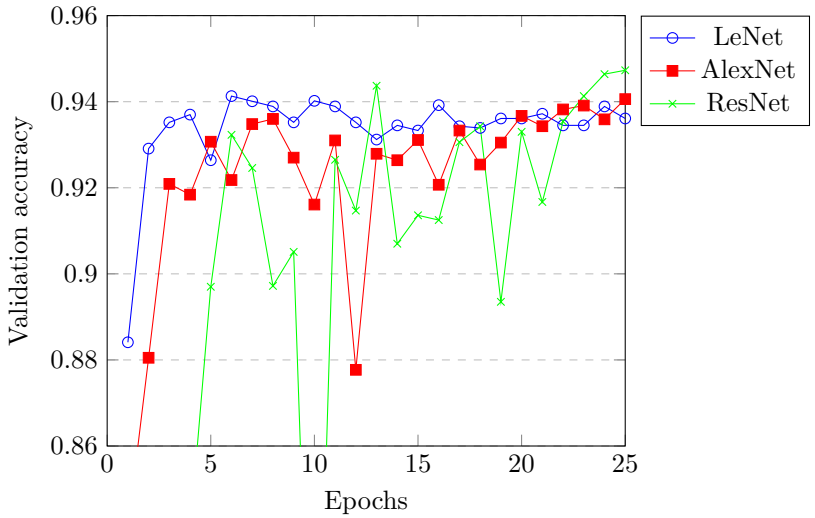
The accuracy of the best model using each CNN on validation set was shown in Table 4, while their fluctuation along with epoch number is plotted in Figure 2. It was discovered that with network complexity increased from LeNet to ResNet, it became harder to converge and needed more epochs to perfectly fit current training data. In the meantime, the performance of complicated models exceeded simple ones such as LeNet.

As mentioned in 5.1.1 and 5.1.2, LeNet and AlexNet achieved their best performance, 93.61% and 94.06% respectively, with feature extraction pipeline 1. However, ResNet reached the overall highest accuracy , 94.89%, with pipeline 2. As can be seen from the results, due to the complexity of the network structure, ResNet was able to learn the relationship between the labels and the features directly from the binarized multi-digit images. This ability allowed it to avoid information loss caused by defining largest digit's bounding box in pipeline 1 and thus achieved the best result.

Model	Validation Accuracy
LeNet+pipeline 1	93.61%
AlexNet+pipeline 1	94.06%
ResNet + pipeline 2	**94.89**%

Table 4: Best performance of each CNN

Figure 2: Validation accuracy of different CNNs



5.3 Ensemble Approach

To further increase the ResNet model accuracy, we tried to apply a stacking ensemble method by performing cross validation. We split the dataset into 10 folds and through cross validation, we obtained 10 different weight sets(i.e. different models). Later on we used each of these trained models to predict the test instances, with the softmax output summed up and identify the class with the largest probability output. By this method, the accuracy on Kaggle test set has grown by 1.9% shown in Table 5.

ResNet	Test Accuracy
Single model	95.00%
Ensembled model	96.9%

Table 5: ResNet model accuracy before and after ensemble

6 Conclusion

In this digit recognition task, we implemented two feature extraction pipelines as well as three different CNN models. We have fine tuned various relevant hyperparameters of each CNN model, including learning rate, gradient descent otimization functions and batch size, etc. Finally, ResNet with pipeline 2 stands out, reaching the highest accuracy on validation set. In order to further improve its performance, we applied the ensemble approach to the ResNet model and got a final accuracy of 96.9% on the test set. During this task, we understood differences between the three CNN models, which inspired our interest in future exploration of the relation between different CNN structures and their properties.

7 Statement of Contribution

Shuhong Shen implemented and tuned ResNet, wrote abstract, introduction, related work, dataset and setup and proposed approach section of this report. Yiran Mao implemented feature extraction pipelines and wrote result and conclusion sections of this report. Jingyuan Wang experimented with LeNet and AlexNet and also prepared for graphs, tables and references in the writeup.

References

[1] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[2] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[3] Y. LECUN. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] Ilya Sutskever Alex Krizhevsky and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *In NIPS*, 2012.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] Fabien Lauer, Ching Y. Suen, and Gérard Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, 40(6):1816–1824, 2007. 19 pages.

[7] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. cite arxiv:1409.1556.

[9] François Chollet et al. Keras. `https://keras.io`, 2015.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008.

[12] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[13] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.