COMP 551 MiniProject 3: Modified MNIST

Yan Miao
260711311
yan.miao@mail.mcgill.ca

Yifei Tang
260762026
yifei.tang@mail.mcgill.ca

Yingzi Xu
260764627
yingzi.xu@mail.mcgill.ca

Abstract

The task of this project is to identify the largest hand written digit in images from a variation of the well known MNIST dataset, each contains multiple digits. At the step of finding the largest digit in an image, we applied functions from OpenCV library to preprocess the image. Then, for digit recognition part, we compared the performance of Convolutional Neural Networks with traditional machine learning models. Also, we conducted experiments on neural network architectures, and hyperparameter tuning. The best performance we discovered using an majority ensemble of three most-accurate Convolutional Neural Network models with test accuracy 97.700%*. The other significant finding is that rather than doing careful preprocessing to select out the largest digit, using the original images after simple thresholding improved the model drastically.*

1. Introduction

Digit recognition is a classic problem in image classification, the purpose of which is to locate digits appearing in an image and recognise it as one of the digits 0-9. This is a technique that has been widely applied in many real-world problems.

In this project we focus on dealing with modified MNIST dataset, which consists of grayscale images containing multiple hand-written images, as well as their labels indicating the largest digit in this image. Our goal is to perform digit recognition using machine learning models. We constructed a preprocessing method to be performed on our dataset, and we tested our models both with and without preprocessing of the raw data. Our best accuracy is achieved with very simple preprocessing. Details on our preprocessing used will be discussed in Sections 3.2 and 4.

We experimented with multiple approaches, including basic machine learning models such as Random Forest[3], k-Nearest-Neighbours[5] and Logistic Regression; as well as more complicated models, Convolutional Neural Networks (CNN). CNN performs in general better than the basic machine learning models in terms of validation accuracy. We also experimented with different architectures of CNN, including self-built CNNs and pre-built CNNs such as LeNet[11] and VGGNet[13], and among all these models out best performing CNN is an ensemble of modified version of VGG-16 and self-built model, and it achieves 97.700% accuracy on the test set on Kaggle Leaderboard.

2. Related Work

With the rise of computational power, recent practice of Convolutional Neural Network(CNN) has garnered tremendous success in a variety of domains in computer vision, namely object localization, object detection, image segmentation, image captioning, autonomous vehicle, etc.

The very first time that CNN has caught people's eyes is the success of AlexNet[10]. In 2012, Alex Krizhevesky and others proposed a deeper and wider CNN model compared to LeNet and won the most difficult ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual Recognition Challenge[6]. AlexNet achieved state-of-the-art recognition accuracy against all the traditional machine learning and computer vision approaches. However, it consists of many fully connected layers which are extremely computationally expensive, so that accounts for why it is mostly not in use as of today.

Following AlexNet is the hype of deep learning. A large number of architectures have been invented, such as GoogleNet[15], VGGNet[14], and ResNet[7], etc, and most of them have outperformed AlexNet on classification task of MNIST, and even moving on to larger dataset, such as CIFAR-10[9] and ImageNet, for tasks other than just classification.

Aside from deep learning approach, traditional machine learning techniques can also be applied to computer vision tasks. Simon Bernard[1] had already experimented with Random Forest[3] on digit classification on MNIST. He earned an accuracy of around 93%, which is actually a very good performance, although not

comparable with deep learning models.

3. Dataset and Setup

3.1. Dataset

In this project we use a modified MNIST dataset which consists of 40,000 grayscale images, each is of size 64×64 and contains multiple hand-written digits 0-9. We split the dataset into 2: training set, which contains 32,000 images, and validation set, which contains 8,000 images. Labels are also provided for each image in a csv file.

We are also given a dataset of 10,000 images that are used as test set, and out Kaggle competition model is tested on this set of images. Labels for test set samples are not given.

Our task is to build a multi-class classification model, which finds the largest digit in the given images in terms of the size of bounding square, and it predicts the class that this digit belongs to (one of the integers 0-9).

3.2. Preprocessing

We used OpenCV library[2] functions to perform blurring, thresholding, drawing contours, and finding bounding squares according to the contours during preprocessing. We noticed that for the images that contain too many components (i.e. potential candidates for the digits) after thresholding, and for the images in which there are digits located very closely to each other, blurring is likely to merge components together, which could lead to incorrectly drawn bounding square, therefore we performed blurring only to images with less than 4 components. Number of components here was obtained by $connectedComponentsWithStats()$ function from OpenCV library. To select the largest digit in an image, we calculated the area of bounding square using the length of larger side of the bounding square as the side length of our bounding square. The digit in the largest bounding square (in terms of area) is the digit we are working with. Due to the fact that the cropped digits are relatively small, we pasted each of them on blank images of size 32×32 which is our resultant image. Figure 1 gives an example of preprocessed image.

4. Proposed Approach

4.1. Models

In this project we experimented with various models, and below are the main ones that we consider significant to mention:

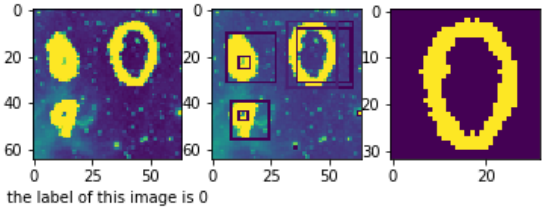

Figure 1. An example of preprocessing

Basic Machine Learning Models

4.1.1 Random Forest

Random forest is an ensemble of decision trees. By bagging a large amount of decision trees, the variance and the possibility of overfitting can be significantly reduced. We built our random forest using SciKit Learn Library[12]. Each decision tree contains nodes (i.e. tests) which pick a number of random features, where the max number of features are set automatically. The best test at each node is decided using Gini impurity. Our random forest consists of 1,000 decision trees. Preprocessing is applied to the dataset.

4.1.2 k- Nearest Neighbours (kNN)

This method involves no learning, it stores all the training examples and when given a test sample, the model finds the k training sample points (in our model $k = 6$) that are "closest" to this test example. The predicted result of this test sample is the majority class of the 6 nearest training sample's labels. We built this model using SciKit Learn Library. Preprocessing is applied to the dataset.

4.1.3 Logistic Regression

This method builds multiple logistic functions to distinguish one class from the rest from training. When given a test sample, the model calculates the probability of this sample belonging to each class and labels it with the most probable one. We built this model using SciKit Learn Library. Preprocessing is applied to the dataset.

Convolutional Neural Networks

Convolutional neural network models are our main focus of this project. We performed experiments on

these models both with and without preprocessing, and the numerical results can be found in Table 3.

In all our CNN models mentioned below, from Keras Library we used Categorical Cross Entropy criterion and Adam Optimizer[8], and we also applied two callbacks, *ReducedLROnPlateau* and *LearningRateScheduler*, to perform learning rate reduction during training process.

Data augmentation is known for its ability to expand training set and thus generalizes the model. However, our tests showed that this takes extremely long training time, therefore we decided not to apply data augmentation to our models.

4.1.4 LeNet-5

LeNet-5 is a one of the most well-known CNN architectures, its simple structure results in less training time. LeNet 5 consists of 5 layers, including 2 convolutional layers, 2 subsampling layers and 2 fully connected layers. However, LeNet-5 gives us relatively low accuracy, and in our experiments we treated it as a baseline to test the performance of the various CNN architectures we built. It is worth noting that we did not add any dropout layers or perform any batch normalization in the neural network.

4.1.5 "Model 1"

We built several different CNNs using Keras Library[4]. Our best performing model (we will use "Model 1" to indicate in the following context) has an architecture that resembles VGGNet's "CONV-CONV-POOL" pattern as shown in the Table 1. The "FC-1024" (fully connected layer with 1024 neurons) and the "softmax" part of the model act together as a classifier to process the features created by the previous layers and return a final result of a predicted class. A dropout layer is added after each max-pooling layer to avoid overfitting and batch normalization is done before feeding into every CONV layer with nonlinear activation function "ReLU". We used mini-batches of size 32 and 25 epochs to train the model.

4.1.6 Modified VGG-16

This model is built based largely on the VGG-16 model which has very simple structure (3×3 convolutional layers). The following are the some major modifications we applied in the VGG-16 model:

- We modified the layer structures. In our modified model we used 2 convolutional layers with depth 32, 2 with depth 64, 3 with depth 128 and 6 with depth 256.
- We used only one fully connected layer, instead of the 2 fully connected layers in VGG-16.

Table 1. Architectures of CNNs

ConvNet Configuration		
LeNet-5	Model1	Modified VGG16
input: 64×64 (or 32×32) grayscale image		
conv5-6	conv3-32	conv3-32
	conv3-32	conv3-32
Avgpool	maxpool	maxpool
conv5-16	conv3-64	conv3-64
	conv3-64	conv3-64
Avgpool	maxpool	maxpool
conv5-120	conv3-128	conv3-128
	conv3-128	conv3-128
		conv3-128
	maxpool	maxpool
	conv3-256	conv3-256
	conv3-256	conv3-256
		conv3-256
	maxpool	maxpool
	conv3-512	conv3-256
	conv3-512	conv3-256
		conv3-256
	maxpool	maxpool
FC-84	FC-1024	FC-1024
soft-max		

- The dimension of fully connected layer in our modified model, which is 1024, is smaller than the original VGG-16 fully connected layer.
- We added dropout layers and batch normalisation layers following the convolutional layers.

Although the original VGG-16 model used in ImageNet takes a painful amount of time to train, this modified model on the MNIST dataset takes reasonable training time, which we will discuss in the results section.

5. Results

In this section, we will discuss the results of experiments conducted on both machine learning models and CNN models.

Our CNN models performed significantly better than the basic machine learning models. The best machine-learning model achieved validation accuracy below 90%, however all our CNN models achieved validation accuracy over 92%.

Table 2. Machine Learning Models Experiments

Model Name	Hyper Parameters	Mean Accuracy
RANDOM FOREST	N ESTIMATOR:1000	0.8906
-	GINI INDEX	-
-	MAX FEATURES: AUTO	-
K-NN	K: 6	0.8805
LOGISTIC REG	C: 1.0	0.6966
-	SOLVER: LBFGS	-
-	MAX ITER: 800	-

Table 3. CNNs Experiments. Notice for the "with preprocessing" column, "NO" means we only performed thresholding to the images to get rid of the noise, and "YES" means we first picked out the digit with the largest bounding square, then put it on a 32×32 blank image to effectively reduce the training time.

Model Name	With Preprocessing	Training Accuracy	Validation Accuracy
LENET-5	YES	0.9672	0.9251
	NO	0.9781	0.7672
MODEL1	YES	0.9668	0.9481
	NO	0.9866	0.9714
MODIFIED VGG16	YES	0.9333	0.9401
	NO	0.9855	0.9731

We will now divide our models into the basic-machine-learning group and CNN group, and we will discuss them respectively.

Basic Machine Learning Models

As in Table 2, the choice of hyperparameters and mean accuracy is presented. These hyperparameters are chosen via 2-fold cross validations on each of the machine learning models, in order to be compared with our CNN models.

We can see that random forest model has a mean accuracy of 89.06%, which is the best among all the experiments in this group, due to the fact that the prediction of the model is made based 1000 estimator.

k-NN also yielded a descent performance with a mean validation accuracy of 88.05%, with $k = 6$. This model is also fast to train as there is no parameter to be learnt.

Our logistic regression model has a validation accuracy of only 69.66% which is less impressive among all we have tested. Also we noticed that our logistic regression model converges extremely slowly during training.

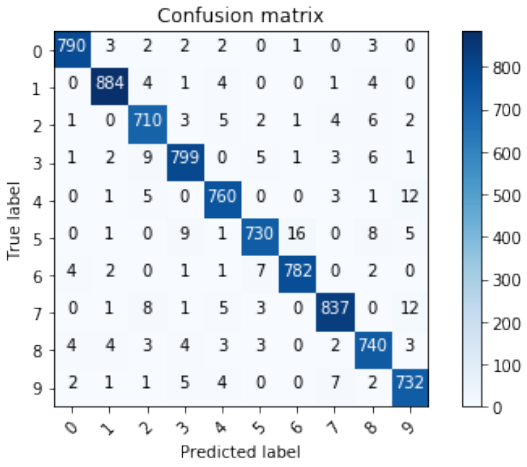

Figure 2. Confusion matrix of our best model

Convolutional Neural Networks

We run experiments on our CNN models, and the corresponding numerical results can be found in Table 3.

Among all the models we have built in this project, the modified VGG-16 model with no preprocessing gave the best performance with validation accuracy of 97.31%. In our experiments we found that some models take extremely long time to train but have worse validation accuracy, while this modified VGG-16 model takes reasonably short training time, with around 300 seconds for each epoch.

Our "Model 1" also performed well with validation accuracy of 97.14% with no preprocessing. Although this model gives us similar validation accuracy compared with modified VGG-16 model, it takes much longer time to train, which is about 800 seconds for each epoch.

LeNet-5 model with preprocessing gives us validation accuracy of 92.51%, which is less impressive but training process takes much shorter time compared the other two CNN models we experimented with, which is only around 30 seconds for each epoch.

For "Model 1" and modified VGG-16 models, using no preprocessing helps us to improve the performance significantly, while for LeNet-5 model, no preprocessing causes bad validation accuracy.

Test Set Result

Our final submission for Kaggle Competition is made by majority voting of the predictions came from Model1 as well as Modified VGG-16. Since the test accuracy on Kaggle for these two single models are around 97.5% (for Model1) and 97.3%(for Modified VGG-16) respectively, we trained Model1 twice to make the number of voting candidates to achieve three. The majority voting

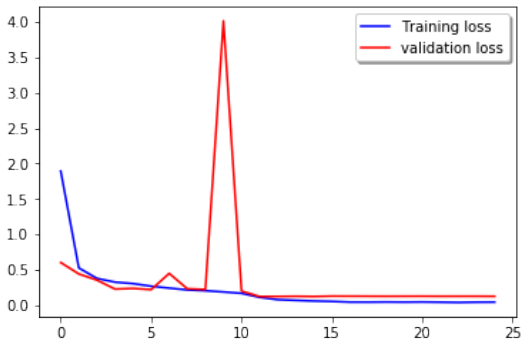

Figure 3. Loss Curve of Best Model

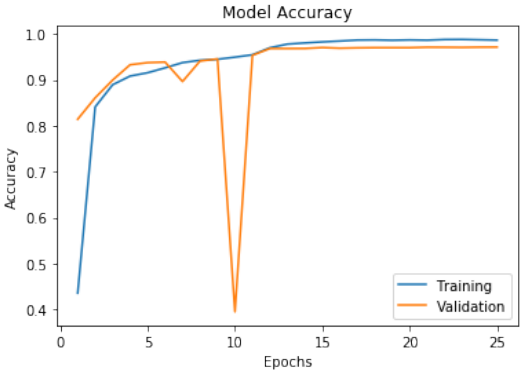

Figure 4. Accuracy curve of our best model

enhanced the test accuracy to 97.700%.

The confusion matrix, loss curve and accuracy curve can be found in Figures 2, 3 and 4.

6. Discussion and Conclusion

In this project we developed a few traditional machine learning and deep learning models in order to compare their performance on classifying the largest digit in the modified MNIST hand-written digit dataset. For that purpose, we experimented with a number of models, namely Random Forest, k-NN, Logistic Regression, and Convolutional Neural Networks with different architecture.

We have shown that traditional machine learning models can be applied to computer vision tasks with an acceptable accuracy, while CNN largely outperforms any traditional machine learning models, with a human-like accuracy.

We envision several future investigations. Clearly, MNIST and its variants have been overly used for many years, and the benchmark on it has been approaching 100%, so we need to move on to new dataset, which is more complicated and contains more objects to prove the robustness of models.

Aside, given the limited time and computational power, we were unable to perform a comprehensive experiment on each choice of optimizer, hyperparameter, architecture, etc. Also, we could try more complex ensemble methods instead of simple majority voting. In the future, we hope to reproduce and develop state-of-the-art CNN architecture in the frontier of computer vision academia, and tackle tasks more than simple classification.

7. Statement of Contributions

The three of us equally contributed to the project on both programming and write-up in LaTeX.

References

[1] S. Bernard, L. Heutte, and S. Adam. Using random forests for handwritten digit recognition. volume 2, pages 1043–1047, 10 2007.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[4] F. Chollet et al. Keras. `https://keras.io`, 2015.

[5] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, Sept. 2006.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] 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.

[13] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, page arXiv:1409.1556, Sep 2014.

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

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.