

MNIST Based Handwritten Digits Recognition

ECE 539 Course Project Report

Linjun Li

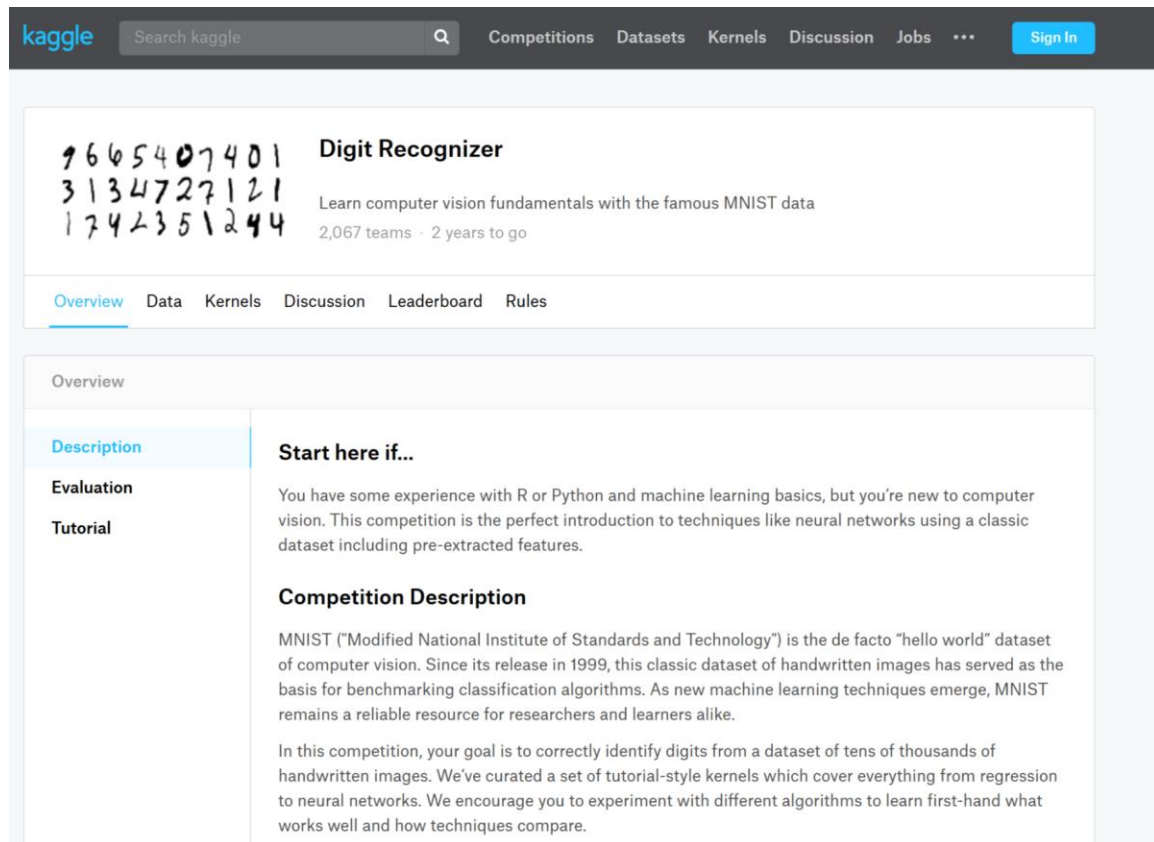
907 920 6059

1.Introduction

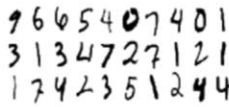
In this project, a handwritten digits recognition system was implemented with the famous MNIST data set. This is not a new topic and the after several decades, the MNIST data set is still very popular and important for evaluation and validation of new algorithms.

Handwritten digits recognition problem has been studied by researchers since 1998 with almost all the algorithms designed by then and even until now. The test error rate decreased from 12% in 1988 by linear classifier to 0.23% in 2012 by convolutional nets, and these days more and more data scientists and machine learning experts are trying to develop and validate unsupervised learning methods such as auto-encoder and deep learning model.

Besides, there are also a Kaggle project based on MNIST, which is perfect resource to begin with.



The screenshot shows the Kaggle website interface for the 'Digit Recognizer' competition. At the top is the Kaggle navigation bar with a search bar and links to Competitions, Datasets, Kernels, Discussion, Jobs, and a Sign In button. Below the navigation bar, the competition header features the title 'Digit Recognizer', a description 'Learn computer vision fundamentals with the famous MNIST data', and statistics '2,067 teams · 2 years to go'. A row of tabs includes Overview, Data, Kernels, Discussion, Leaderboard, and Rules. The 'Overview' section is active, displaying a table with 'Description', 'Evaluation', and 'Tutorial' rows. The 'Description' row is expanded, showing a 'Start here if...' section for newcomers and a 'Competition Description' section detailing the MNIST dataset and the goal of the competition.

Digit Recognizer	
	Digit Recognizer Learn computer vision fundamentals with the famous MNIST data 2,067 teams · 2 years to go
Overview Data Kernels Discussion Leaderboard Rules	
Overview	
Description	Start here if... You have some experience with R or Python and machine learning basics, but you're new to computer vision. This competition is the perfect introduction to techniques like neural networks using a classic dataset including pre-extracted features.
Evaluation	
Tutorial	Competition Description MNIST ("Modified National Institute of Standards and Technology") is the de facto "hello world" dataset of computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike. In this competition, your goal is to correctly identify digits from a dataset of tens of thousands of handwritten images. We've curated a set of tutorial-style kernels which cover everything from regression to neural networks. We encourage you to experiment with different algorithms to learn first-hand what works well and how techniques compare.

2. Work performed

In MNIST dataset, the data is already well prepared: the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field. The training set is composed of 30,000 patterns from SD-3 and 30,000 patterns from SD-1. And test set was composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1.

Because the dataset is too huge, a better idea is beginning with an available subset, which could be found on Kaggle webpage. Although the dataset has been labeled, it's still my task to design proper method to preprocessing the data including read binary files with proper programming language and choose acceptable data structure to avoid overflow. Moreover, a series of machine learning algorithms were applied to the training set and validated with the test set, and the models were tuned until the performance and accuracy is relatively acceptable.

Based on my experience on Tensorflow and Theano, I developed this recognizer with python language. And in order to enhance my engineering skills and experience on tuning real-world systems instead of toy models, I focused on some details of the models and implement part of them on my own, but I didn't reinvent everything.

The purpose of this project is to apply knowledge and technique learned during this course to a non-trivial problem, hence a series machine learning algorithms and models, including but not limited to KNN, multilayer perceptron and convolutional network were designed to complete this task. I tested different models and parameters in these models, such as various distance functions and activate functions of SLP & MLP models, and convolution kernel size and number of convolutional layers.

Finally, these models were applied to larger dataset for evaluation, including all 50,000 images in the dataset and cross validated.

4. Project Progress

4.1 Model Validation on Small Sample (From Week 8 to Week 10)

From week 8 to week 10, I basically followed the official tutorial to install Tensorflow on my computer, and try to understand how it works and ran some simple model to learn how to use it. Besides, I took out some data and implemented some models, including SVM, KNN, single layer perceptron model and multi-layer perceptron model with Matlab tool boxes to validate my idea. The best result I got was 100% accuracy on training set (which is useless) but 91.28 accuracy on test set, and the model I used was fully connected multi-layer perceptron model.

4.2 Attempt of Convolutional Network(From Week 11 to Week 12)

After week 10, I began to learn more detail about convolutional neural network and try to implement one with tutorial of Tensorflow. I did some research about the popular models researchers used on MNIST, and it seems that the most efficient one would be LeNet, and played with a lot of details about convolutional network, including the method of pooling layer, batch size and some tricks for feature visualization.

I implemented multi-layer perceptron model and convolutional neural network with Tensorflow because there are a lot of tutorial and code wrote by other researchers, and I got 99.3% accuracy when I was lucky, which is pretty close to the best result according to LeCun's Website. But there are still some model without using neural network but only with SVM & feature extract get higher accuracy than my model, and I'm planning to dive into other detail of this topic.

In week 12, I was working with implementing other method like SVM and try to get better result, I focused on analyzing the structure of the network and dataset and parameter tweaking in my models. I also digested different optimization methods from SGD to ADAM which may benefit my further research of machine learning.

5.Results and discussions

Among all the methods I tried on MNIST dataset, a committee of three convolutional networks which are ResNet-50, VGG-5, VGG-16, (inspired and modified from kkweon's work on github), has the best performance, which is 99.80% accuracy, even better than professor Lecun's committee of 35 convolutional networks with 0.23% error rate, but not always repeatable. Tweaking the kernel sizes and numbers of convolutional layers indicates that more convolutional layers are helpful to improve the accuracy while change kernel size has little effect and the accuracy even decreased sometime. Possible reason could be that convolutional layers contribute to feature extraction and feature transform.

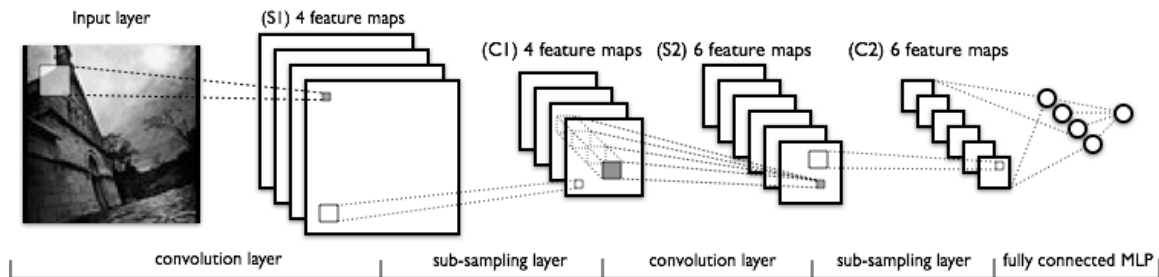


Fig.1 Structure of LeNet-5

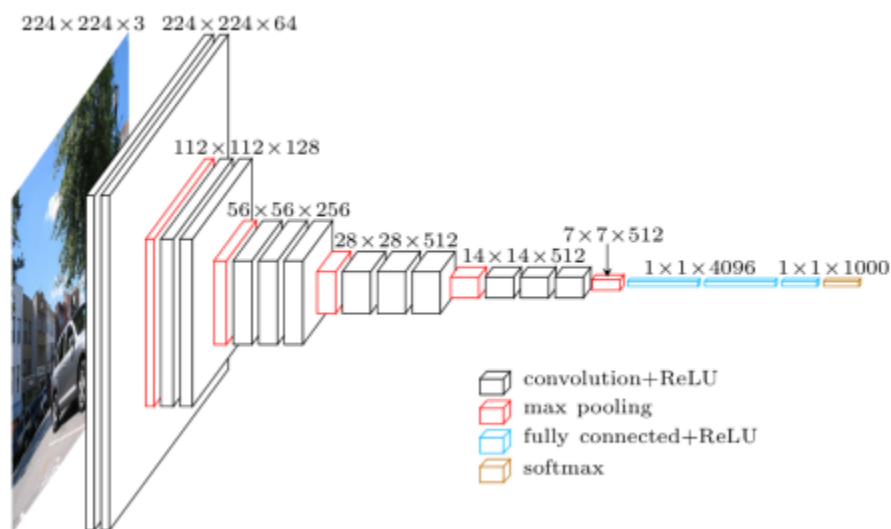


Fig.2 Structure of VGG-16

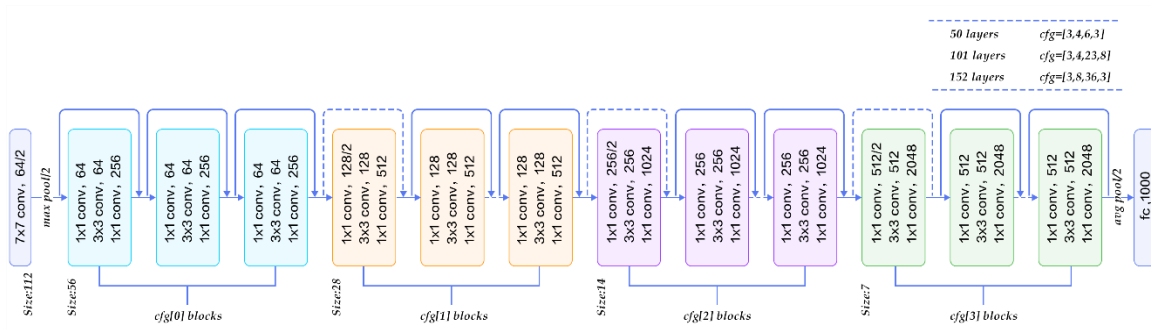
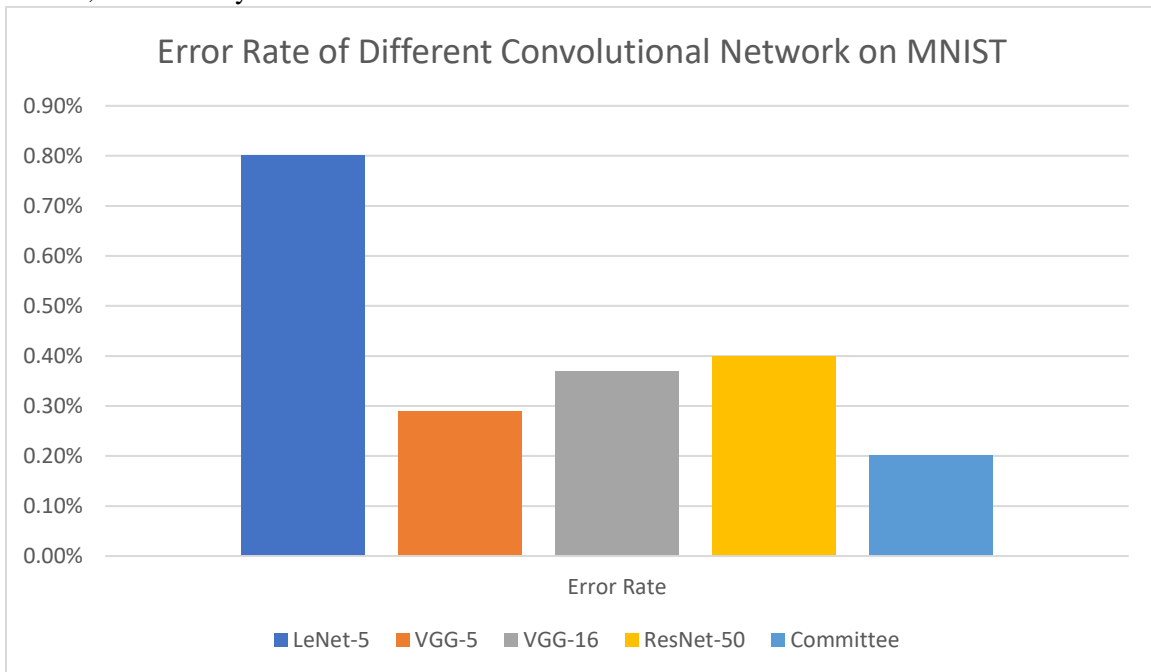


Fig.3 ResNet-50

I've tried four different convolutional neural network and one committee model on MNIST dataset, the accuracy of which are showed below:



Other results including KNN with L3 Norm achieved 97.0% accuracy, while SLP and MLP model with softmax and cross entropy only reach 91.75% and 94.12% accuracy.

I also found numeric computing problems: it's possible that rounding method and other implementation of numeric algorithms are different in frameworks, as a result, when I implemented exactly the same model with Caffe2 and MxNet, I have to write more code than using Tensorflow and Pytorch but the accuracy and performance are worse.

6. Conclusions

Although expert committee is engineers' dirty way to solve the problem, 99.80% is even better than committee of 35 conv. net with 0.23% error rate on LeCun's website.

My experiment on KNN and SLP & MLP indicates that the more complex the better might not be true, although KNN is almost the easiest lazy learning model, it still has better performance than the so-called complex model MLP when the distance function is chosen properly.

The problems during implementation tells that devils are in the details and talk is cheap, it's not easy to solve a problem completely without practice.

Understanding the data is the most important thing. It's impossible to tell the best method to solve a specific problem before trying several reasonable methods on it. As Samuel Beckett said, "Ever tried, ever failed, no matter. Try again, fail again, fail better. Every time our algorithm fails, we get better understanding on our data and get closer to our goal."

7. References

- [1] <http://yann.lecun.com/exdb/mnist/>
- [2] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998
- [3] <https://www.kaggle.com/c/digit-recognizer>
- [4] Dan Cireşan, Ueli Meier, Juergen Schmidhuber: Multi-column Deep Neural Networks for Image Classification, CVPR, 2012
- [5] Liu, Qing, Tang Xianlun, Zhang, Na: Structure Optimized Convolutional Neural Network Based on Unsupervised Pre-training, Gongcheng Kexue Yu Jishu/Advanced Engineering Science, 2017, Vol.49, p.210-215
- [6] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, Rob Fergus: Regularization of Neural Networks using DropConnect, ICML, 2013
- [7] Jia-Ren Chang, Yong-Sheng Chen: Batch-normalized Maxout Network in Network, arXiv 2015
- [8] Ming Liang, Xiaolin Hu: Recurrent Convolutional Neural Network for Object Recognition, CVPR 2015
- [9] http://blog.csdn.net/weixin_38776853/article/details/72721077
- [10] <http://www.jeyzhang.com/cnn-learning-notes-1.html>
- [11] https://www.tensorflow.org/get_started/mnist/beginners
- [12] <http://neuralnetworksanddeeplearning.com/chap3.html#softmax>
- [13] <https://github.com/kkweon/mnist-competition>
- [14] <https://colah.github.io/posts/2014-10-Visualizing-MNIST/>