COMP 551 Fall 2020

Mini Project 3: Multi-label Classification of Image Data

Written Report

Group 6

Haowei Qiu (260762269) Liang Zhao (260781081) Ningchen Ma (260784506)

1 Abstract

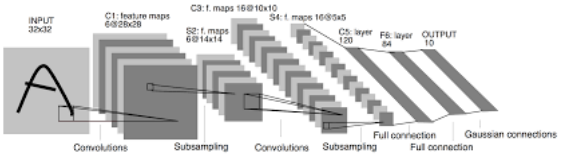
In this project, we implemented a Convolutional Neural Network(CNN) Model with leNet as our model architecture and Stochastic gradient descent(SGD) as our optimizer. The model is then trained with a modified MNIST dataset by dividing the dataset into training and validation datasets. The time-consumed, validation loss and accuracy of our model are then compared on our dataset to investigate the effects of using different hyper-parameters and optimizers. In our case with leNet architecture, we concluded that the SGD optimizer with hyper-parameters including train percentage = 0.9, epochs = 200, learning rate = 0.1, batch size = 128 and momentum = 0.9 works the best — with accuracy = 99.46% and ranked $24th$ in the Kaggle competition(Dec 13th, 2020).

2 Introduction

This project implements a Convolutional Neural Network(CNN) Model with architecture leNet and optimizer Stochastic gradient descent(SGD). The dataset used is a modified MNIST dataset with 5 digit numbers in one grayscale image. The dataset is processed by cross-validation and sliced to training and validation parts. We then convert the data to tensorflow in order to use computational graphs which can be executed much more efficiently than in Python. To further speed up the training procedure, the main computations are performed in GPU. For predicting the 5 numbers in each image, a tensor stack is used to cooperate with our CNN model. The average training time for running 200 epochs is 90 mins in our model. This project also invested that how time-consumed and accuracy are affected by optimizer and hyper-parameters. By using the leNet architecture, the accuracy of predicting the validation sets fluctuates after 150 epochs. Lastly, to get the model with the highest accuracy, a simple termination condition is implemented which saves the current model if $accuracy > 0.9945$.

3 LeNet Architecture

LeNet is a convolutional neural network structure proposed by Yann LeCun in 1989, a family of ConvNets named LeNet trained for recognizing MNIST handwritten characters with robustness to simple geometric transformations and distortion. The key intuition here is to have low-layers alternating convolution operations with max-pooling operations. The convolution operations are based on carefully chosen local receptive fields with shared weights for multiple feature maps. Then, higher levels are fully connected layers based on a traditional MLP with hidden layers and softmax as the output layer.



leNet layers

There are 7 layers in leNet in addition to input but our modified leNet contains 11 hidden layers containing multiple pooling and convolution layers:

Layer Hidden 1: a convolution layer with six convolution kernels of 5x5, input channel 1 and output channel 32 which can prevent the information of the input image from falling out of the boundary of convolution kernel.
Layer Hidden 2: a convolution layer with six convolution kernels of 5x5 and output channel 64 with similar function to C1.
Layer Hidden 3: in the the pooling, each layer in each feature map is connected to 2x2 neighborhoods in the corresponding feature map in Hidden 2. Then it follows a convolution layer with kernel size 3 and output channel 64.
Layer Hidden 4: contains convolutional layer with 64 output channel and kernel size 3 and a BatchNorm function where mean and standard-deviation of 'data' are calculated.
Layer Hidden 5: a pooling layer similar to hidden 3
Layer Hidden 6: applies a linear transformation with 9216 input features and 256 output features
Layer Hidden 7: applies the rectified linear unit function
Layer Hidden 8: applies a linear transformation with 256 input features and 128 output features
Layer Hidden 9: applies the rectified linear unit function
Layer Hidden 10: applies a linear transformation with 128 input features and 84 output features.
Layer Hidden 11: is fully connected to hidden 10, and 84 feature graphs are output.

4 Discussion

Justification on hyper-parameters: leNet architecture: Our idea is inspired by the graph in lecture slide CNN, p18. We first tried to implement AlexNet but at the end found that leNet worked better in our case even though that AlexNet was inspired by leNet.
optimizer: We compared Adadelta, SGD and Adam as our candidate optimizer and by experiment we found that SGD gave us the highest accuracy.
Other hyper-parameters we set the parameters according to experiments and logic found in project 2.
Validation and train performance with epochs: When epochs increase, validation and training accuracy increase and validation error decreases; but after 150 epochs with accuracy around 99.3%, there is no obvious relation between accuracy and epochs. Interestingly, when the epoch reaches around 200, sometimes validation accuracy drops to 0, this might be caused by NaN values.
Further improvement: while we are reading the paper about leNet, we found that leNet inspired the creation of two famous CNN architecture: AlexNet and VGG. Surprisingly, our modified leNet worked better than AlexNet. If we have more time, we would investigate more on AlexNet and hopefully we could find the correct way to implement it and get better results.
Statement of Contributions Liang Zhao contributed to the CNN model architecture, Ningchen Ma contributed to testing and Kaggle competition and Haowei Qiu contributed to testing of Hyper-parameters and data-processing. All 3 group members participated in the mini project report writing.
Appendix

Architecture	Modified leNet
Optimizer	Stochastic gradient descent(SGD)
hidden layers	11
epochs	200
learning rate	0.1
batch size	128
momentum	0.9

Table 1. Best performing model