Lucy Zhang
02/20/23
ID: 1229576766

Project 3: Classification Using Neural Networks and Deep Learning

Introduction:

In this project, I'm required to understand the whole process of compiling different layers (Convolutional Layer, Fully-Connected Layer, Pooling Layer, Activation Layer, Loss function) of a simple Convolutional Neural Network (CNN) for the visual classification task and train the CNN with a fixed epoch number and initialization of parameters. Also, I need to compile my own evaluation code to evaluate the trained CNN to obtain the training and testing results.

Baseline Report:

The Baseline code is a machine learning model trained on the MNIST dataset to classify handwritten digits. The model was trained for 12 epochs, with a batch size of 64, using the Adadelta optimizer. The training data consisted of 60,000 images of size 28x28 with a single channel, while the test data consisted of 10,000 images. The model achieved a test accuracy of 97.85% and a test loss of 0.0701. Its performance improved throughout the training process, as evidenced by the decreasing training and validation loss and the increasing training and validation accuracy. The training accuracy reached 0.9783 at the end of the training process. By changing the kernel size to 5x5 and redoing the experiment, the test accuracy increased slightly to 97.89%, and the testing loss went down to 0.0655.

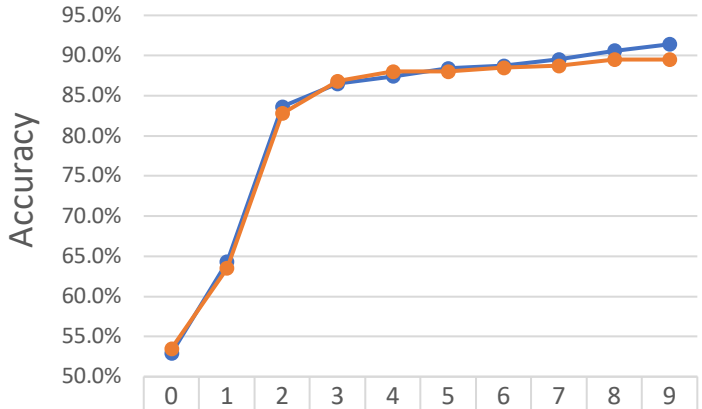
Then I tried to change the number of the feature map by adding 10 in the first and second convolutional layers after resetting the kernel, the testing accuracy increased 0.33% to 98.18%, and the testing loss came down to 0.562. Thus, both increasing kernel size and adding the feature maps approaches can improve the accuracy, and adding maps has better performance.

Evaluation:

Given a pre-defined neural network with layers included Convolutional, ReLu, MaxPooling , Flatten, and Softmax, I wrote a missing evaluation function for CNN_train_test based on the training code to evaluate the trained network. It takes in a neural network object, images for evaluation, and corresponding labels, evaluating one image at a time and extracting a batch of images and their corresponding labels. The output was calculated by calling the forward method of each layer in the network.

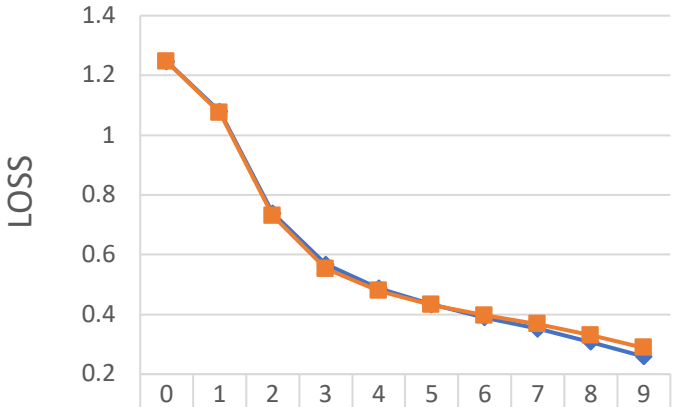
Finally, update the loss and accuracy after computing them. Since the total epoch number is 10, the result came to training accuracy 91.4% with loss 0.259 and test accuracy 89.5% with loss 0.288 after looping 10 times. The high test accuracy and high similarity of training and testing on both accuracy and loss result proved the simple CNN with different layers has high performance on image classification.

Training and Testing Accuracy vs Epochs from CCN_train_test



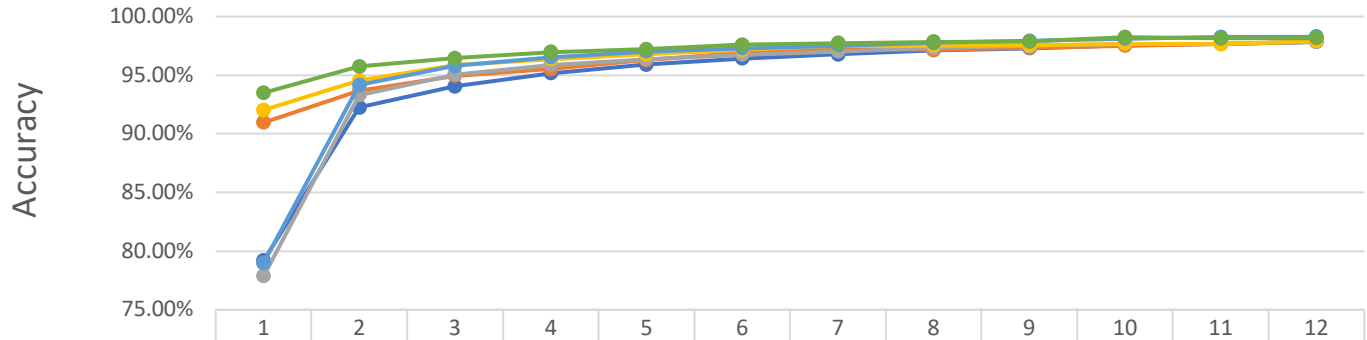
	0	1	2	3	4	5	6	7	8	9
Train	52.9	64.3	83.6	86.5	87.4	88.4	88.7	89.5	90.6	91.4
Test	53.5	63.5	82.8	86.8	88.0	88.0	88.5	88.7	89.5	89.5

Training and Testing Loss vs Epochs from Cnn_train_test



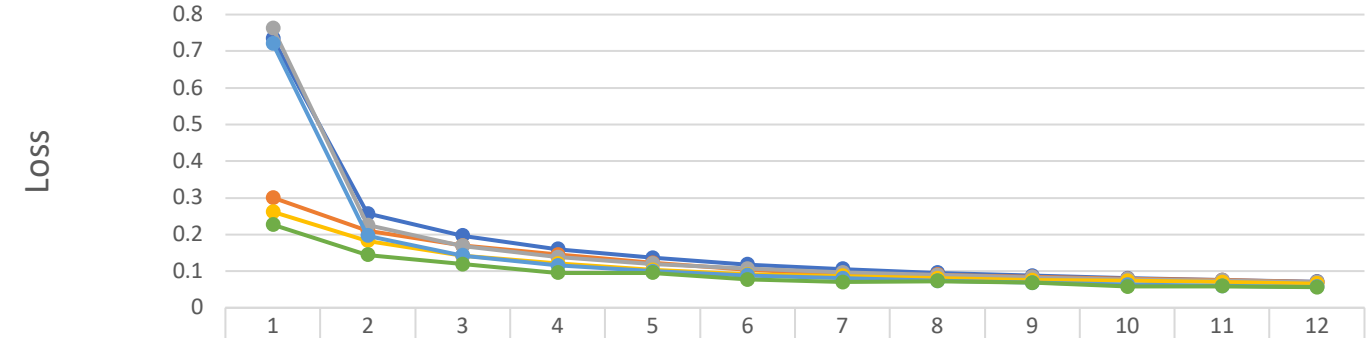
	0	1	2	3	4	5	6	7	8	9
Train	1.25	1.08	0.74	0.57	0.49	0.43	0.39	0.35	0.31	0.26
Test	1.25	1.08	0.73	0.55	0.48	0.43	0.4	0.37	0.33	0.29

Training and Testing Accuracy vs Epochs from Baseline



	1	2	3	4	5	6	7	8	9	10	11	12
Train Original	79.18%	92.25%	94.05%	95.18%	95.90%	96.41%	96.79%	97.14%	97.29%	97.56%	97.68%	97.83%
Test Original	90.96%	93.67%	94.94%	95.53%	96.29%	96.91%	97.21%	97.16%	97.34%	97.51%	97.69%	97.85%
Train 5x5 Kernel	77.86%	93.32%	95.06%	95.89%	96.36%	96.74%	97.03%	97.33%	97.50%	97.69%	97.68%	97.85%
Test 5x5 Kernel	92.03%	94.53%	95.85%	96.38%	96.77%	97.17%	97.49%	97.52%	97.55%	97.68%	97.66%	97.89%
Train Add 10 Maps	79.00%	94.16%	95.80%	96.54%	97.03%	97.29%	97.52%	97.76%	97.91%	98.11%	98.24%	98.27%
Test Add 10 Maps	93.50%	95.73%	96.45%	96.96%	97.20%	97.59%	97.72%	97.84%	97.88%	98.24%	98.13%	98.18%

Training and Testing Loss vs Epochs from Baseline



	1	2	3	4	5	6	7	8	9	10	11	12
Training Original	0.7342	0.2565	0.1962	0.1603	0.1361	0.1183	0.1056	0.0951	0.0872	0.0806	0.0755	0.0709
Test Original	0.3004	0.2101	0.1695	0.1454	0.1226	0.103	0.0907	0.0902	0.0841	0.0791	0.0748	0.0701
Training 5x5 Kernel	0.7627	0.2254	0.1683	0.1382	0.1198	0.1067	0.0965	0.0893	0.0829	0.0777	0.074	0.0697
Test 5x5 Kernel	0.2617	0.1823	0.1416	0.1202	0.1031	0.0908	0.0862	0.0803	0.0756	0.0733	0.0702	0.0655
Train Add 10 Maps	0.7206	0.1964	0.1423	0.1156	0.0994	0.088	0.0802	0.0733	0.0681	0.0632	0.0594	0.0566
Test Add 10 Maps	0.2266	0.1446	0.1187	0.0954	0.0949	0.0773	0.0702	0.0726	0.068	0.0577	0.0585	0.0562