Evaluating convolutional neural networks in MNIST classification

Mohamed Osman

December 4, 2018

Introduction

This report will be focusing on image classification using convolutional neural networks. Three variations of the algorithm will be used as the models for classifying the data, and their performances will be measured using 3-fold cross validation on a training set to get an idea of how they might perform on new data. From there, the models will be re-trained on a section of the data set and tested on the rest, attaining empirical losses for each model that will be graphed along with their true-risk confidence bounds. Then, to conclude, the models will be ranked from best to worst given their performances in this report.

The data set that will be used is MNIST, which includes 70,000 images of 28x28 monochromatic handwritten digits, with each pixel representing a value between 0 and 255. The goal for each of the models is to take each image and classify them according to their "true" intentional digit, which can be a number between 0 and 9. 60,000 samples will be used as the training set, and the remanining 10,000 samples will be used as the hold-out set.

Model Descriptions

There are three convolutional neural network models that will be used in this report. All three of them were built with mean squared error loss functions and stochastic gradient descent optimizers. They will be trained with a batch size of 32 and an epoch of 2.

CNN 1

This model consists of an input layer, a convolution layer with 32 3x3 filters, and an output layer takes a flattened matrix-to-vector input and maps the result to a 10-size feature that classifies the image based on the feature position with the greatest value. It is simple by design and makes use of only linear activation functions.

CNN 2

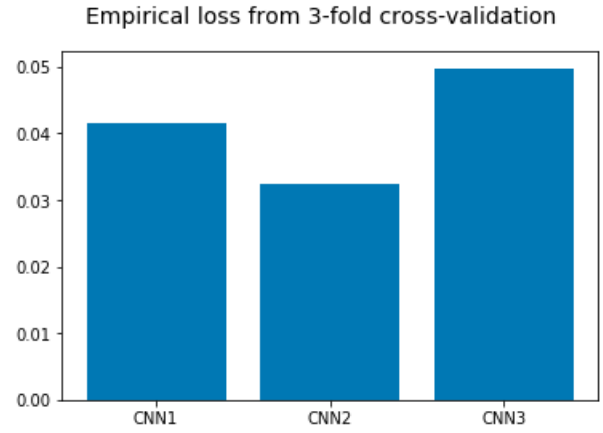
This model is a bit more complex. It has a pair of layers that convolute the input using 32 3x3 filters and downsamples the result with a 2x2 filter. It also uses rectified linear units at the end of the convolution layer and the output layer to better represent the non-linear nature of the original data.

CNN 3

This model has the most going for it. It has the same design as CNN 2, except it adds a second pair of convolutional and downsampling layers, where the new convolution layer uses 3x3 64 filters. They are placed before the output layer.

Cross Validation

To see how the models might perform on fresh data, 3-fold cross validation is used. It splits the training set into three folds, and each fold will be used as a testing set while the remaining folds will be used to train the models. This results in three empirical loss values per model, which will be averaged by the amount of folds (which is, in this case, three) to get a single averaged loss per model. The average losses for each model are graphed on the following bar chart.

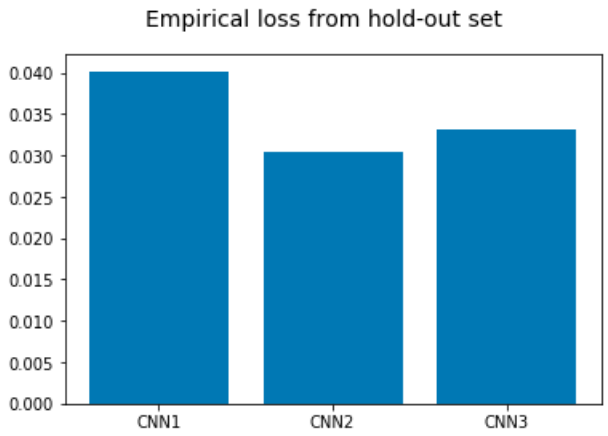



```
Cross-validation losses for each model:
CNN1: 0.04162362614572048
CNN2: 0.03240672734379768
CNN3: 0.04974569374720256
```

Through this process, the CNN 2 achieved the lowest loss, followed by CNN 1 and CNN 3.

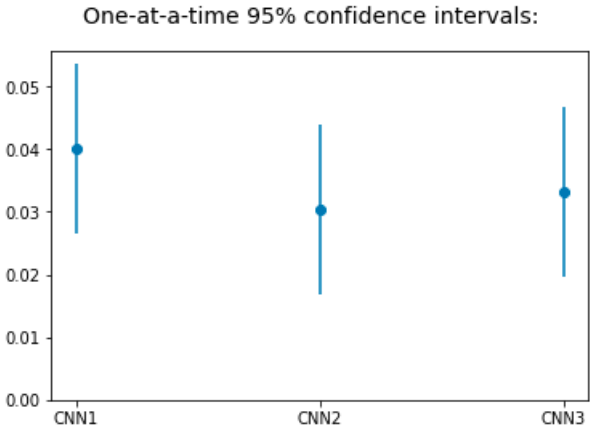
Training and Testing

The models are now re-trained using the entire training set and tested on the hold-out set. The empirical losses for each model are graphed on the following chart.

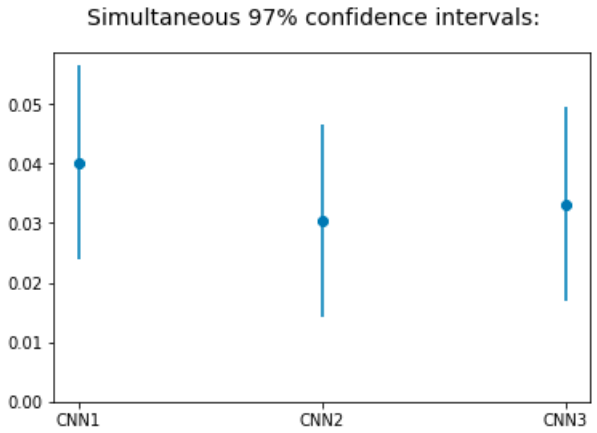



```
Hold-out set losses for each model:
CNN1: 0.04017598705291748
CNN2: 0.030390585446357726
CNN3: 0.03313038868308067
```

Using Hoeffding's inequality, confidence bounds were created for each model out of the given losses. These bounds provide estimates of where the true loss of a model might be, given a certain level of confidence. The bounds were plotted with a 95% one-at-a-time confidence, and a wider 97% simultaneous confidence (99% for each model).




```
One-at-a-time 95% Conf. Interval values for each model:
CNN1: [0.026594971895511285, 0.05375700221032367]
CNN2: [0.016809570288951532, 0.04397160060376392]
CNN3: [0.019549373525674478, 0.046711403840486866]
```




```
Simultaneous 97% Conf. Interval values for each model:
CNN1: [0.023899750745730187, 0.056452223360104775]
CNN2: [0.014114349139170434, 0.046666821753545015]
CNN3: [0.01685415237589338, 0.04940662499026796]
```

Compared to the losses given by the cross-validation test, all three models performed better. Relatively, CNN 2 achieved the lowest loss, followed by CNN 3 and CNN 1. The cross-validation test also had CNN 2 achieving the lowest loss, but the results for CNN 1 and CNN 3 were different. But, if cross-validation was the sole predictor without looking at the test data, it would have been right in showing that CNN 2 achieves the lowest loss out of the group.

Conclusion

The evaluation of the models for this experiment relies on looking at the confidence bounds and ranking the models based on the upper-bound height of their bounds, from lowest to highest. CNN 2 and CNN 3 have similar overlapping bounds, but, since CNN 2 has a lower upper bound, it is the better performer for this test. CNN 1 has a bound that is much higher than both CNN 2 and CNN 3, which makes it the worst-performing model for this test.

Although CNN 2 performed the best for this experiment, more model-selection experiments using new data will be needed to choose a better classifier for this problem. Such examples include using models with more optimized layers, and using cross-validation with more folds to reduce bias.