Convolutional Neural Network Optimization Problem – FRE9733 Week 11 Homework

In previous weeks, we have analyzed multilayer perceptron neural networks. This week, we will be analyzing and optimizing Convolutional Neural Networks (CNN).

In our problem, we have MNIST dataset which has a set of 60, 000 greyscale images, 28 × 28 pixels in size. These images represent digits 0 − 9. So, we have 10 output possibilities at the end of the network. Due to multi class classification, we have softmax activation function at the output. Our network is really similar to Lenet-5 network which is Yann LeCun's groundbreaking architecture in this area of neural networks. In his paper, he has 2 convolutional layers which consists of conv. Layer and pooling layers with filter sizes (f=5,s=1, filter size=6) and (f=5,s=1, filter size=16). At the end of these two layers, we have one fully connected layer (FC) which is connected to softmax output layer. The diagram of this convolutional neural network can be seen below.

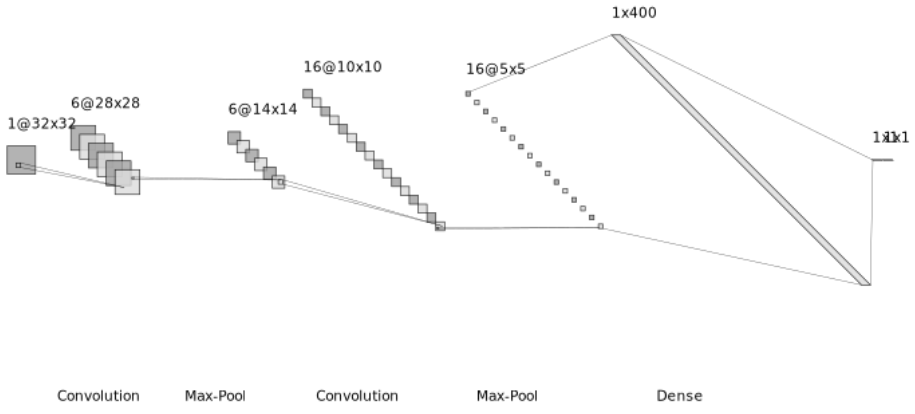

Figure 1: CNN Network Example

We should not forget that we only have 1 channel in our example unlike 3 channels in original rgb images. Also, we are assuming our image pixels (size) is 32x32 as it is stated in the original paper.

In this report, we will try to get 95% accuracy while generating and training the model with at least possible parameters. To do this, we are going to change the filter sizes (f), fully connected layer size (FC) and also filter number in each conv. layer. We do not change the stride size. Even before starting analysis, we can expect that FC size should be the most important one as it causes to enlarge the parameter size most if we compare it with convolution layers. This is the main purpose why convolutional neural networks are built. Basically, they allow us to not connect each individual feature in image to neural network layer. If this was the case, we could get millions of parameters. In the second part of the report, we are going to use our best model from the first part and we will try to compare the

performances with different activations functions as well as different pooling techniques which have two options that are max pooling and average pooling.

CNN Hyperparameter Tuning

In each test section below, we have tried different values for filter size, number (f, #filters) and Fully Connected (FC) layer size as we mentioned in the first part of the report. We can observe the difference for this numbers in the corresponding table and we can also observe the entropy, accuracy, precision, recall and F1 score for corresponding table and CNN network.

We will try to minimize entropy while we try to maximize accuracy. *The main goal is to hold accuracy over 95% while we shrink the overall parameter size.* First 4 models are with filter size (f) selection of (5x5) and the rest is f of (3x3).

i) Models with filter size (f) = 5

1) First Model

This model is the default model that is specified before the report.

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=5,#filter=20,s=1)	(28,28,20)	15680	520
POOL 1	(14,14,20)	3920	0
CONV2(f=5,#filter=50,s=1)	(10,10,50)	5000	5050
POOL2	(5,5,50)	1250	0
FC3 (layer size=50)	(50,1)	50	62550
Softmax(10 outputs)	(10,1)	10	510
		Total Parameters	68630

```
Fitting set entropy: 0.07791565563598016


========================Evaluation Metrics=========================
 # of classes:    10
 Accuracy:        0.9745
 Precision:       0.9744
 Recall:          0.9746
 F1 Score:        0.9744
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

2) Second Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=5,#filter=6,s=1)	(28,28,6)	4704	156

POOL 1	(14,14,6)	1176	0
CONV2(f=5,#filter=16,s=1)	(10,10,16)	1600	1616
POOL2	(5,5,16)	400	0
FC3 (layer size=40)	(40,1)	40	16040
Softmax(10 outputs)	(10,1)	10	410
		Total Activation Size	Total Parameters
		8954	18222

```
Fitting set entropy: 0.1547228034938558


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0.9731
 Precision:       0.9730
 Recall:          0.9734
 F1 Score:        0.9731
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

3) Third Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=5,#filter=4,s=1)	(28,28,4)	3136	104
POOL 1	(14,14,6)	1176	0
CONV2(f=5,#filter=8,s=1)	(10,10,8)	800	208
POOL2	(5,5,8)	200	0
FC3 (layer size=12)	(12,1)	12	2412
Softmax(10 outputs)	(10,1)	10	130
		Total Activation Size	Total Parameters
		6358	2854

```
Fitting set entropy: 0.1301116024468991


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0.9648
 Precision:       0.9650
 Recall:          0.9642
 F1 Score:        0.9645
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

4) Fourth Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=5,#filter=4,s=1)	(28,28,4)	3136	104
POOL 1	(14,14,6)	1176	0
CONV2(f=5,#filter=8,s=1)	(10,10,8)	800	208
POOL2	(5,5,8)	200	0
FC3 (layer size=10)	(10,1)	10	2010
Softmax(10 outputs)	(10,1)	10	110
		Total Activation Size	Total Parameters
		6356	2432

```
Fitting set entropy: 0.06762403122934985


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0.9521
 Precision:       0.9515
 Recall:          0.9522
 F1 Score:        0.9516
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

ii) Models with filter size (f) = 3

1) First Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=3,#filter=6,s=1)	(30,30,6)	5400	60
POOL 1	(15,15,6)	1350	0
CONV2(f=3,#filter=16,s=1)	(13,13,16)	2704	160
POOL2	(6,6,16)	576	0
FC3 (layer size=40)	(40,1)	40	23080
Softmax(10 outputs)	(10,1)	10	410
		Total Activation Size	Total Parameters
		11104	23710

```
Fitting set entropy: 0.1166825923950931


=======================Evaluation Metrics=======================
 # of classes:    10
 Accuracy:        0.9566
 Precision:       0.9578
 Recall:          0.9565
 F1 Score:        0.9565
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

2) Second Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=3,#filter=6,s=1)	(30,30,6)	5400	60
POOL 1	(15,15,6)	1350	0
CONV2(f=3,#filter=16,s=1)	(13,13,16)	2912	160
POOL2	(6,6,16)	576	0
FC3 (layer size=20)	(20,1)	20	11540
Softmax(10 outputs)	(10,1)	10	210
		Total Activation Size	Total Parameters
		11292	11970

```
Fitting set entropy: 0.46389680245723947


=======================Evaluation Metrics=======================
 # of classes:    10
 Accuracy:        0.9689
 Precision:       0.9694
 Recall:          0.9686
 F1 Score:        0.9688
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

3) Third Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0

CONV1(f=3,#filter=6,s=1)	(30,30,6)	5400	60
POOL 1	(15,15,6)	1350	0
CONV2(f=3,#filter=16,s=1)	(13,13,16)	2912	160
POOL2	(6,6,16)	576	0
FC3 (layer size=10)	(10,1)	10	5770
Softmax(10 outputs)	(10,1)	10	110
		Total Activation Size	Total Parameters
		11282	6100

```
Fitting set entropy: 0.09300709869878562


=======================Evaluation Metrics=======================
 # of classes:    10
 Accuracy:       0.9618
 Precision:      0.9622
 Recall:         0.9614
 F1 Score:       0.9616
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

4) Fourth Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=3,#filter=6,s=1)	(30,30,6)	5400	40
POOL 1	(15,15,6)	1350	0
CONV2(f=3,#filter=8,s=1)	(13,13,8)	1352	80
POOL2	(6,6,8)	288	0
FC3 (layer size=10)	(12,1)	10	2890
Softmax(10 outputs)	(10,1)	10	110
		Total Activation Size	Total Parameters
		9434	3120

```
Fitting set entropy: 0.15732600783415518


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0.9590
 Precision:       0.9594
 Recall:          0.9587
 F1 Score:        0.9588
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

5) Fifth Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=3,#filter=4,s=1)	(30,30,4)	3600	40
POOL 1	(15,15,4)	900	0
CONV2(f=3,#filter=8,s=1)	(13,13,8)	1352	80
POOL2	(6,6,8)	288	0
FC3 (layer size=10)	(10,1)	10	2890
Softmax(10 outputs)	(10,1)	10	110
		Total Activation Size	Total Parameters
		7184	3120

```
Fitting set entropy: 0.1679918262959325


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0.9497
 Precision:       0.9502
 Recall:          0.9498
 F1 Score:        0.9495
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

6) Sixth Model

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=3,#filter=4,s=1)	(30,30,4)	3600	40
POOL 1	(15,15,4)	900	0

CONV2(f=3,#filter=8,s=1)	(13,13,8)	1352	80
POOL2	(6,6,8)	288	0
FC3 (layer size=12)	(12,1)	12	3468
Softmax(10 outputs)	(10,1)	10	130
		Total Activation Size	Total Parameters
		7186	3718

```
Fitting set entropy: 0.20113557390351985


=======================Evaluation Metrics=======================
 # of classes:    10
 Accuracy:       0.9511
 Precision:      0.9511
 Recall:         0.9503
 F1 Score:       0.9504
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

Comments about the Results

i) Models with filter size (f) = 5

We have tried 4 different models for this part. The first model was the default model defined before the report and there will comparison between our final model and that model at the end of this report, in another section.

In the first model, we are observing significant accuracy with a low cross-entropy score which is desired in accurate models. The transition to second model is done by decreasing all defined metrics slightly and we can see the instant result from the accuracy and parameter size as the accuracy decreased moderately and the parameter size is decreased significantly. By this way, our model is more computational efficient while not losing in accuracy and cross-entropy much. We decreased number of filters even more to (4,8). When we first try with FC3 size of 12, we observe that we almost hit the 95%, so we have only small range values to test. At the end, we have found that when we select FC3 size 10, we are .001 close to 95% which is acceptable. This is the 4th model and if we examine the results, we can also see a low (really good) entropy which is expected in good predictive models.

ii) Models with filter size (f) = 3

We first tried 3x3 filter size as this was mentioned in the original paper. We made the most experiments with this filter size as you can see from the results part above.

If we compare the first three models, we can observe what full connected layer (FC) means to us. It was clearly expected to have less parameters when we have smaller size FC. However, we also get better accuracy. This affect is known as model dependent and the optimal FC layer size change form model to model. Our parameter size fall by half and the accuracy got better with decreasing FC size from 40 to 20 from 1st model to 2nd model. Again, we decrease the parameter size by half from 2nd to 3rd model while losing only small amount of accuracy.

In the 4th model, we try to decrease the 2nd convolutional layer's filter size by half in order to reduce the parameter size even more. We can see a similar result to affect of FC as the parameter size falls by half when we reduce the filter size from 16 to 8. We only lose the accuracy in the order of 0.004 which is not significant and still larger than 95%.

In the 5th and 6th models, we try to reduce the filter size in first convolutional layer. We can see that reducing this layer's filter size reduces the accuracy significantly to under 95%. As can be observed from the 5th model, we are under 95% and to get above the specified accuracy, we add 2 more fully connected layer. As can be seen from the result at model 6, we get the least parameter size with an accuracy larger than 95%.

The Best Model

The best model decision has been made by filtering the last result that is filter size (3x3 or 5x5). So, we will compare the best results from part i) and part ii). So, the 4th model for f=5 and the 6th model is selected as the best models for corresponding filter size and we can see that f=5 selection clearly outperforms the other one. So, our best model with least parameters and accuracy that is greater than 95% will be as following,

Accuracy: 0.9521

Entropy: 0.0676

	Activation Shape	Activation Size	# Parameters
Input	(32,32,1)	1024	0
CONV1(f=5,#filter=4,s=1)	(28,28,4)	3136	104
POOL 1	(14,14,6)	1176	0
CONV2(f=5,#filter=8,s=1)	(10,10,8)	800	208
POOL2	(5,5,8)	200	0
FC3 (layer size=10)	(10,1)	10	2010
Softmax(10 outputs)	(10,1)	10	110
		Total Activation Size	**Total Parameters**
		6356	**2432**

As a side note, we have observed better results with Relu function used as activation function. So, our best model's final accuracy can be observed in the next section where we analyze the Relu function.

Additional Comparisons for Optimization CNN

1) Pooling Technique

There are two possible techniques for pooling that are MAX and AVG pooling. We have already implemented the one with Max pooling. Now, we will observe the results with average pooling.

```
Fitting set entropy: 0.3630714511868888


=======================Evaluation Metrics=======================
 # of classes:    10
 Accuracy:        0.8956
 Precision:       0.8963
 Recall:          0.8948
 F1 Score:        0.8942
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

We got a significantly worse result (worse accuracy and entropy) with AVG poling which indicate that it does not work well with our CNN model.

2) Different Activation Functions

i) Tanh function

```
Fitting set entropy: 0.3240179692500479


=======================Evaluation Metrics=======================
 # of classes:    10
 Accuracy:        0.9527
 Precision:       0.9522
 Recall:          0.9523
 F1 Score:        0.9521
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

Tanh function is used in conv. layers and at the dense layer, relu is used and the results is not bad. We got even better accuracy than our best model but lost enormous entropy. So, our best model outperforms tanh performance.

ii) Sigmoid Function

```
Fitting set entropy: 2.3099970145629034


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0.1135
 Precision:       0.1135        (9 classes excluded from average)
 Recall:          0.1000
 F1 Score:        0.2039        (9 classes excluded from average)
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

Sigmoid activation function performs really bad if we compare it with our best model. It excludes 9 classes from result which shows us that sigmoid function does not work for our digit classification problem.

iii) Relu Function (Best Model)

```
Fitting set entropy: 0.13393523227178486


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0.9627
 Precision:       0.9630
 Recall:          0.9627
 F1 Score:        0.9625
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

When we use Relu function as activation functions in conv. layers rather than identity function, we observe that we gain considerable accuracy approximately 1%. Thus, this model outperforms our best model! However, it again loses some entropy values it increased from 0.0676 to 0.1339 which is basically doubled but this is ignorable as the increase in accuracy is quite high*. We will use Relu in our best model.*

Time Comparison of Best Model and Default Model

To see the difference of parameter size ,we can compare the fitting time. Our best model has 2432 total parameters while default model has 68630 total parameters. The difference is 28 times. This is a significant difference. The fitting times can be seen below,

Our Best model with 2432 parameters: 27 Seconds

Default model with 68630 parameters: 1 Minute 25 Seconds

So, the difference is in the order of 3 which shows our best model are more computationally efficient while not losing so much in accuracy.