Neural Network Architectures

All the raw program output can be found in the *Resut-NNA.txt* file.

For this assignment, the cross validation will use 5-fold.

1 Determine Convolution Neural Network Architecture

1.1 Find the Best Architecture

In here, we trying to create the *Neural Network* based on *LeNet-5*. The structure will look like:

1. Convolution 2D layer with batch normalization (kernel size = 5, stride = 1)
    a. Use ReLU
    b. Batch normalization
2. Max Pooling Layer (kernel size = 2, stride = 2, no padding)
3. Convolution 2D layer with batch normalization (kernel size = 3, stride = 1)
    a. Use ReLU
    b. Batch normalization
4. Max Pooling Layer (kernel size = 2, stride = 2, no padding)
5. 2 Fully connect layers
    a. Use Sigmoid
    b. 40% Dropout during training
6. Output layer with one node
    a. Use Sigmoid

To determine some module type we should use in the structure and during the training, such as optimizer, convolution layer output channel and fully connect layer structure, we need a parameter sweep to get the best model architectures.

Here is the sweeping range:

- Optimizer (learning rate = 0.01)
    - Adam
    - SGD (no momentum)
- Convolution layer output channel
    - First layer and second layer [(6, 12), (6, 16), (8, 16)]
- Fully connect layer structure
    - fully connect layer nodes [(20, 10), (30, 20), (120, 84)]

With 500 iteration training on all different parameters, we can get 94.9917% cross validation accuracy with error bound [94.2826%, 95.7009%] on **Training Set**, when using the following parameters:

- Optimizer Adam
- Convolution layer output channel (6, 16)
- Fully connect layer structure (120, 84)

1.2 Process Output

For Fully connect layers output channel: (20, 10), we can have the following output:

Convolution Layer	**(6, 12)**	**(6, 16)**	**(8, 16)**
Adam Optimizer	94.2212%	94.0837%	93.9461%
SGD Optimizer	67.2812%	67.4739%	57.7600%

For Fully connect layers output channel: (30, 20), we can have the following output:

Convolution Layer	**(6, 12)**	**(6, 16)**	**(8, 16)**
Adam Optimizer	94.3588%	94.9092%	94.3313%
SGD Optimizer	60.3742%	83.5443%	92.5702%

For Fully connect layers output channel: (120, 84), we can have the following output:

Convolution Layer	**(6, 12)**	**(6, 16)**	**(8, 16)**
Adam Optimizer	94.7716%	94.9917%	94.1387%
SGD Optimizer	84.1772%	84.4249%	75.4540%

2 Feature Engineering

Our **Training Set** only have 3634 samples to help our model to do the training, we can do the following data augmentation:

- Image mirroring
- Transform Image
  - Randomly resize (output size 30 x 30, from origin image scale (0.8 to 1.0)
  - Randomly rotation (15 degrees)
  - Randomly change the brightness, contrast and saturation of an image
  - Crops the Image at the center (output 24 x 24)

By using the cross validation on the new **Training Set** with data augmentation, we can have the following output:

Iteration	**500**	**600**	**700**
Original Training Set (3634 Training Samples)	94.3588%	93.9736%	94.5239%
With Mirror (7268 Training Samples)	95.0881%	95.1018%	94.7441%
Mirror with transform batch 1 (14536 Training Samples)	94.8610%	95.1156%	95.1844%

From this result we can know that the **Training Set** with Mirror data and without Transform data and training 600 iteration will have 95.1018% cross validation accuracy with error bound [94.4001%, 95.8036%]. By using this model to predict the **Test Set**, we can get 96.1221% accuracy with error bound [95.0352%, 97.2091%].

And the **Training Set** with Mirror data and with Transform data (batch 1) and training 700 iteration will have similar but better performance than previous one, 95.1844% cross validation accuracy with error bound [94.4883%, 95.8805%]. Also, by using this model to predict the **Test Set**, we can get 96.4521% accuracy with error bound [95.4107%, 97.4936%].

Overall, we can know that with more training data, the model will performance better than just use the original data. And with more iTransformed data will performance better than just include the Mirror data.

3 More Improvement

3.1 Parameter sweep

In order to have better performance on our model, we can use the **Training Set** with Mirror and with transform batch 1 to do another round of parameter sweep on these parameters:

- Convolution layer kernel size
  - First layer and second layer [(5, 3), (5, 5)]
- Pooling Layer
  - Use Max Pooling or Average Pooling
- Learning rate
  - In range [0.01, 0.1]

With all different parameters and 700 iteration on training, we can still get 95.5696% cross validation accuracy with error bound [94.9006%, 96.2386%] on **new Training Set**, when using the following parameters:

- Convolution layer kernel size (5, 3)
- Pooling Layer Average Pooling
- Learning rate 0.01

For Pooling Layers use Max Pooling, we can have the following output:

Learning Rate	**0.01**	**0.1**
Convolution Layer Kernel Size (5, 3)	95.1844%	92.7353%
Convolution Layer Kernel Size (5, 5)	94.9161%	93.1549%

For Pooling Layers use Average Pooling, we can have the following comparation:

Pooling	**Max**	**Average**
Learning Rate 0.01	95.1844%	95.5696%

By using this model to predict the **Test Set**, we can get 95.9571% accuracy with error bound [94.8482%, 97.0660%]. Even this model has better performance on cross validation, but by using to predict the testing data it's worse than previous one. This may because of the random initialization process of the model by using different random value as the default weight and bias.

In later we can try update both of these 2 models to try to have better outcome.

3.2 Update the Methods

Also, we can update methods used in the *Neural Network*, such as:

- Make fully connected layers use ReLU method
- Make output layer use SoftMax method to compute the final result

But none of them can have better performance as before, not only that, the model will not converge to the optimal, and just randomly give the result back.

3.3 Update Dropout Probability

We can also update the dropout probability in the fully connected layer.

3.3.1 Max Pooling Model

In here we do the cross validation to check the model with *Max Pooling*. By checking the dropout probability for fully connected layer, we can have the following output:

	0.05	**0.1**	**0.15**	**0.2**	**0.4**
Iteration 300	94.8335%	94.8679%	94.9436%	94.7578%	95.2944%
Iteration 400	94.8266%	95.0055%	94.8748%	94.8954%	95.0055%
Iteration 500	94.8679%	95.0399%	94.9161%	94.5102%	95.3288%
Iteration 600	94.5927%	95.1844%	95.1569%	94.9711%	95.3770%
Iteration 700	94.1731%	94.7854%	94.6203%	94.8473%	**95.5696%**
Iteration 800	94.4964%	94.9298%	94.7303%	94.9298%	95.2738%
Iteration 900	94.8679%	95.1569%	94.8748%	94.8266%	95.3357%
Iteration 1000	94.7303%	95.0468%	94.9642%	95.1225%	95.3082%

From this we can know that when dropout probability for fully connected layer is 0.4 with 700 iterations, we can have best performance as 95.5696% cross validation accuracy with error bound [94.9006%, 96.2386%] on **new Training Set**. and by using this model to predict the **Test Set**, we can get 96.4521% accuracy with error bound [95.4107%, 97.4936%].

3.3.2 Average Pooling Model

In here we do the cross validation to check the model with *Average Pooling*. By checking the dropout probability for fully connected layer, we can have the following output:

	0.05	**0.1**	**0.15**	**0.2**
Iteration 300	94.9780%	95.1018%	94.9436%	94.7716%
Iteration 400	95.1500%	95.2050%	95.2532%	94.8886%
Iteration 500	94.7234%	94.8817%	94.7716%	94.9986%
Iteration 600	**95.3701%**	94.7922%	95.4045%	94.7578%
Iteration 700	95.0124%	95.0399%	94.9161%	94.2488%
Iteration 800	95.2325%	95.1637%	94.7647%	95.0881%
Iteration 900	95.2050%	94.8886%	95.3564%	95.1912%
Iteration 1000	95.1636%	95.3288%	95.1156%	94.7854%

From this we can know that when dropout probability for fully connected layer is 0.05 with 600 iterations, we can have best performance as 95.3701% cross validation accuracy with error bound [94.6898%, 96.0533%] on **new Training Set**. And by using this model to predict the **Test Set**, we can get 96.3696% accuracy with error bound [95.8836%, 97.8458%].

4 Another Model

When using the following parameter:

- Convolution layer output channel (36, 96)
- Pooling Layer Max Pooling
- Dropout rate 5%

the model can have this output on cross validation:

Iteration	**200**	**300**	**400**
Cross Validation Accuracy	95.7898%	95.2669%	94.2212%

By using this model with 200 iteration, we can have 95.7898% cross validation accuracy with error bound [94.6898%, 96.0533%] on **new Training Set**. And by using this model to predict the **Test Set**, we can get 97.1122% accuracy with error bound [96.1694%, 98.0550%].

5 Best Model

In final, we can know that after we do the data augmentation and the *Neural Network* Architecture look like the following, we can have the best performance, 97.1122% accuracy with error bound [96.1694%, 98.0550%] on **Test Set**. But because the none of the lower bound of the *Max Pooling* Model, *Average Pooling* Model or the Large Convolution Layer Model is higher than other's higher bound, we cannot say which one is better.

5.1 Data Augmentation

When training, we will use both original samples, mirrored samples, transformed original samples, and transformed mirrored samples.

- Image mirroring
- Transform Image
  - Randomly resize (output size 30 x 30, from origin image scale (0.8 to 1.0)
  - Randomly rotation (15 degrees)
  - Randomly change the brightness, contrast and saturation of an image
  - Crops the Image at the center (output 24 x 24)

5.2 Architecture

- Optimizer Adam (learning rate = 0.01)
- Convolution 2D layer with batch normalization
  - kernel size = 5, stride = 1, output channel = 36
  - Batch normalization
  - Use ReLU Method
- Max/Average Pooling Layer (kernel size = 2, stride = 2, no padding)
- Convolution 2D layer with batch normalization
  - kernel size = 3, stride = 1, output channel = 96
  - Batch normalization
  - Use ReLu Method
- Max/Average Pooling Layer (kernel size = 2, stride = 2, no padding)
- Fully connect layer
  - First Layer Nodes 120
  - Second Layer Nodes 84
  - Use Sigmoid Method
  - 40% Dropout during training on with *Max Pooling* or 5% with *Average Pooing*
- Output layer with one node
  - Use Sigmoid Method

6 Another Complexed Model to Try

With the following architecture, we doubled the input features to fully connection layer:

- Optimizer Adam (learning rate = 0.01)
- First Convolution Layers Set
  - Convolution 2D layer with batch normalization
    - kernel size = 5, stride = 1, output channel = 6
    - Batch normalization
    - Use ReLU Method
  - Max Pooling Layer (kernel size = 2, stride = 2, no padding)
  - Convolution 2D layer with batch normalization
    - kernel size = 3, stride = 1, output channel = 16
    - Batch normalization
    - Use ReLu Method
  - Max Pooling Layer (kernel size = 2, stride = 2, no padding)
- Second Convolution Layers Set
  - Convolution 2D layer with batch normalization
    - kernel size = 3, stride = 1, output channel = 6
    - Batch normalization
    - Use ReLU Method
  - Max Pooling Layer (kernel size = 2, stride = 2, no padding)
  - Convolution 2D layer with batch normalization
    - kernel size = 3, stride = 1, output channel = 16
    - Batch normalization
    - Use ReLu Method
  - Max Pooling Layer (kernel size = 2, stride = 2, no padding)
- Combine the output from 2 convolution layers sets
- Fully connect layer
  - First Layer Nodes 120
  - Second Layer Nodes 84
  - Use Sigmoid Method
  - 5% Dropout during training
- Output layer with one node
  - Use Sigmoid Method

With this model and without any parameter sweep, we can magically get 96.4521% accuracy with error bound [95.4107%, 97.4936%] on **Test Set**. If there have more time, I think after several iteration's parameter sweep, this model can become the best model.