**Homework-3 Report**

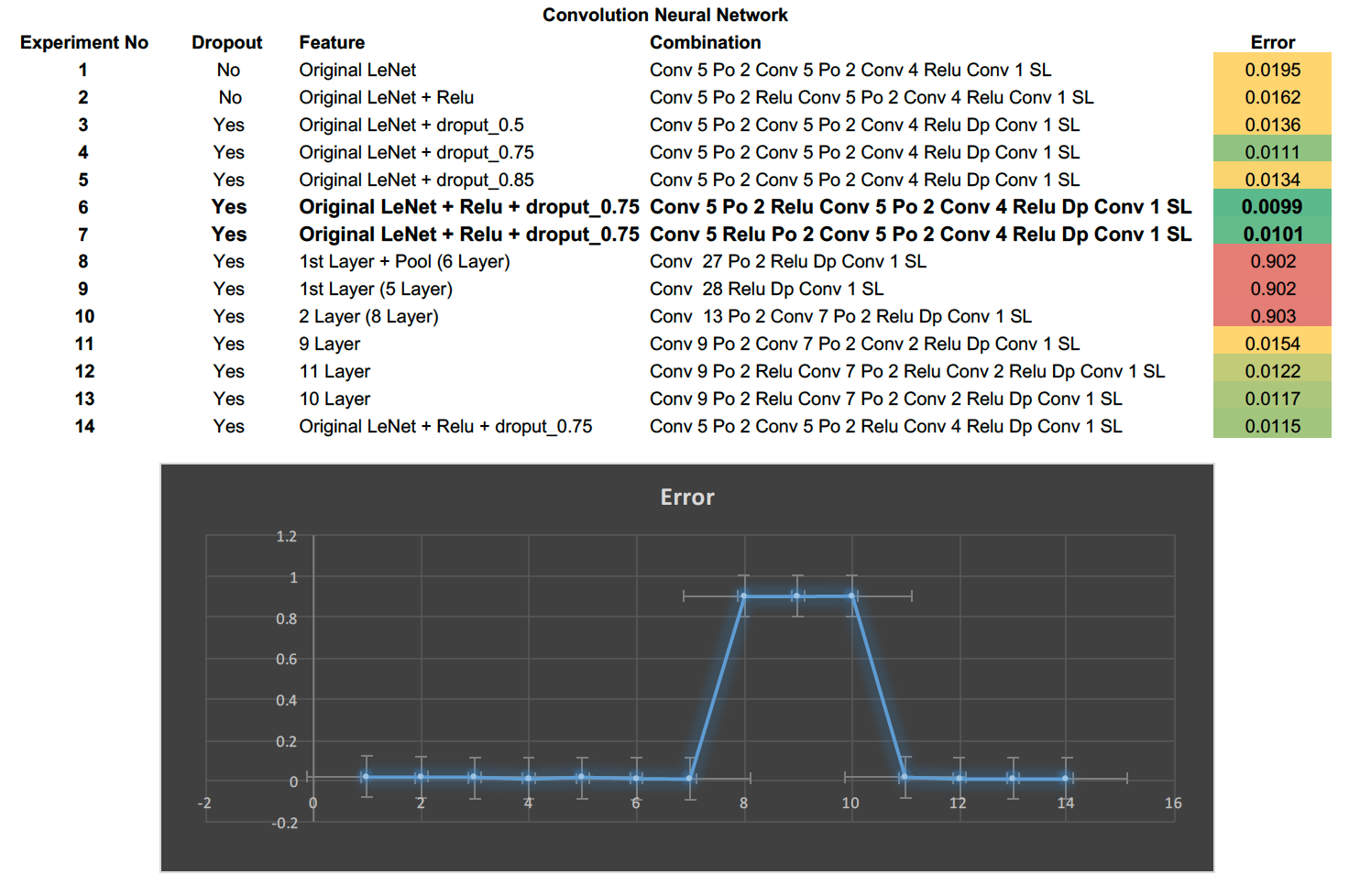
**MatConvNet –**

**Algorithm:**

1. Download cnn\_mnist\_6156.m and run the original
2. Change Topology of Neural Network [Adding & removing pair of convolutional and max- pooling layers]
3. Add Dropout Layer to Neural Network [Change the dropout rate between 0.5 to 1]
4. Add Relu layer after pairs of convolutional and max- pooling layers
5. Compare the result

**Discussion & Results**:

* Adding dropout layer to the original LeNet decreased the error.
* Lowest error was calculated for dropout rate (0.75).
* Dropout layer decreases error due to variance as it randomly removes nodes from input & hidden layers in order to avoid over fitting.
* Decreasing the Number of layers from Original LeNet increased the error.
* **Lowest Error [0.099] was recorded for 6th Experiment in which Relu and Dropout Layer [dropout rate 0.75] was added to Original LeNet**



**Following are the result graph for above mentioned experiment:**

**Experiment No 1: [Error: 0.0195]**

**[Conv 5 Po 2 Conv 5 Po 2 Conv 4 Relu Conv 1 SL]**

****

**Experiment No 2: [Error: 0.0162]**

**[Conv 5 Po 2 Relu Conv 5 Po 2 Conv 4 Relu Conv 1 SL]**

****

**Experiment No 3: [Error: 0.0136]**

**[Conv 5 Po 2 Conv 5 Po 2 Conv 4 Relu Dp Conv 1 SL]**

****

**Experiment No 4: [Error: 0.0111]**

**[Conv 5 Po 2 Conv 5 Po 2 Conv 4 Relu Dp Conv 1 SL]**

****

**Experiment No 5: [Error: 0.0134]**

**[Conv 5 Po 2 Conv 5 Po 2 Conv 4 Relu Dp Conv 1 SL]**

****

**Experiment No 6: [Error: 0.0099]**

**[Conv 5 Po 2 Relu Conv 5 Po 2 Conv 4 Relu Dp Conv 1 SL]**

****

**Experiment No 7: [Error: 0.0101]**

**[Conv 5 Relu Po 2 Conv 5 Po 2 Conv 4 Relu Dp Conv 1 SL]**

****

**Experiment No 8: [Error: 0.902]**

**[Conv 27 Po 2 Relu Dp Conv 1 SL]**

****

**Experiment No 9: [Error: 0.902]**

**[Conv 28 Relu Dp Conv 1 SL]**

****

**Experiment No 10: [Error: 0.903]**

**[Conv 13 Po 2 Conv 7 Po 2 Relu Dp Conv 1 SL]**

****

**Experiment No 11: [Error: 0.0154]**

**[Conv 9 Po 2 Conv 7 Po 2 Conv 2 Relu Dp Conv 1 SL]**

****

**Experiment No 12: [Error: 0.0122]**

**[Conv 9 Po 2 Relu Conv 7 Po 2 Relu Conv 2 Relu Dp Conv 1 SL]**

****

**Experiment No 13: [Error: 0.0117]**

**[Conv 9 Po 2 Relu Conv 7 Po 2 Conv 2 Relu Dp Conv 1 SL]**

****

**Experiment No 14: [Error: 0.0115]**

**[Conv 5 Po 2 Conv 5 Po 2 Relu Conv 4 Relu Dp Conv 1 SL]**

****

**LibSVM -**

**Algorithm:**

1. Load the data from imdb.mat file which is created during execution of cnn\_mnist\_6156.m file
2. Transform image.data from 4 Dimensional to 2 Dimensional (784 X 20000) and store it in X
3. Take transpose of X making it X [20000,784]
4. Divide X by 255 to scale the data
5. Create a sparse Matrix out of X
6. Load images.labels into Y and make a Transpose of Y
7. Divide data in X by 255 to scale data to value between [0 to 1]
8. Create sparse matrix from X as it contains lot of rows with value 0
9. Use libsvmwrite, libsvmread functions to write & read data
10. Take first 10000 rows as train data & train label from X & Y
11. Take last 10000 rows as test data & test label from X & Y
12. Run Cross validation function on each type of Kernel to get the best cost (-c ) and gamma (-g ) value.
13. Use this cost and gamma values to train the model and then use it on test data to predict accuracy.

**Results & Discussion:**

* First run cross validation function for **–c (Range: -5 to 15) and –g (Range: -15 to 3)**  used the best value of –c & -g on test data.
* Increasing the value of –c increased training time for svm
* Result of cross validation function are show in image a presented below.
* **Radial Basis Kernel gave highest accuracy of 96.76 % on test data among all kernel.**

1. **Linear Kernel**: Highest Accuracy on test data: **93.58% [-c 0.05]**
2. **Polynomial Kernel Degree 2:** Highest Accuracy on test data: **96.59% [-c 8192 –g 0.00048 –r 0]**
3. **Polynomial Kernel Degree 4:** Highest Accuracy on test data: **96.52% [-c 0.5 –g 0.125 –r 1]**
4. **Radial Basis Kernel:** Highest Accuracy on test data: **96.76 % [-c 8 –g 0.5]**

**Following is the result of Cross Validation Function & Test Data on given kernels**

