CLASSIFICATION USING NEURAL NETWORKS AND DEEP LEARNING

Francis Mendoza

ASUID: 1213055998

CSE575- Statistical Machine Learning

fmendoz7@asu.edu

1. INTRODUCTION

a. I used the same MNIST dataset from project 1 to train a Convolutional Neural Network (CNN) using Tensorflow, Python, and Keras to classify digits. I first ran a modified version of the boilerplate code provided (keeping only the lr parameter for Adadelta, as the original copy did not run) and reported both the loss and the accuracy. In the second part of the project, I redid the experiment with the original boilerplate by changing the kernel size, and plot the learning errors as well as reported error and accuracy. I then redid the experiment a third time by changing the number of feature maps in the first and second convolutional layers, while reporting the same output of testing error, accuracy on the test set, and a plot of the learning error.

i. `jupyter nbconvert --to script Mendoza-CSE575_Project3.ipynb`

2. PART #1: Assessing Original Boilerplate

i. Part #1 involved running the original boilerplate and evaluating a plot for the learning error over epochs, testing loss, and testing accuracy.

ii. The original boilerplate made the following assumptions:

1. Input size of the image is 28x28
2. First Convolutional Layer contains:
    a. 6 Feature Maps
    b. Kernels are 3x3
    c. Stride of 1 (by default if omitted)
3. First Max Pooling Layer
    a. Pooling is 2x2
    b. Stride of 1 (by default if omitted)
4. Second Convolutional Layer
    a. 16 Feature Maps
    b. Kernels are 3x3
    c. Stride of 1 (by default if omitted)
5. Second Max Pooling Layer
    a. Pooling is 2x2
    b. Stride of 1 (by default if omitted)
6. First Hidden Layer

a. 120 nodes
b. ReLU activation function

7. Second Hidden Layer
   a. 84 nodes
   b. ReLU activation function
8. Softmax Layer with 10 output nodes

iii. However, the original boilerplate did not run successfully, so the following modification was made:

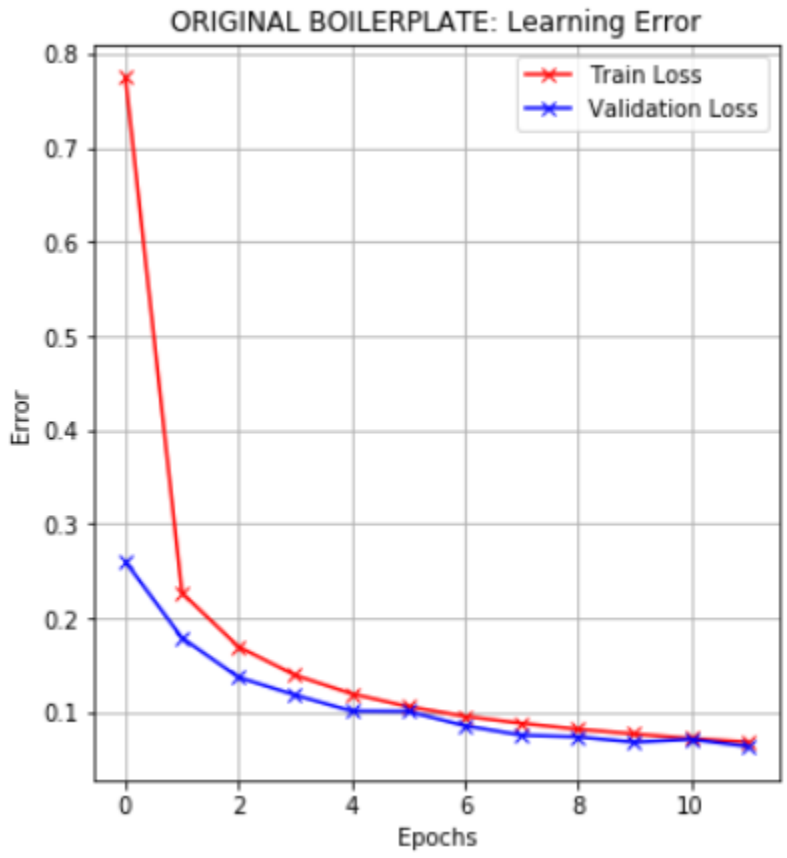
MODIFIED CODE TO RUN: Just kept lr=0.1 for Adadelta method

In [*]:

```
# MODIFIED THIS LINE, AS ORIGINAL BOILERPLATE DID NOT RUN CORRECTLY
"""
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(lr=0.1, rho=0.95, epsilon=None, decay=0.0),
              metrics=['accuracy'])
"""

# REPLACED WITH THIS LINE INSTEAD
"""
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(lr=0.1),
              metrics=['accuracy'])
"""
```

iv. The following learning error over epochs plot was:

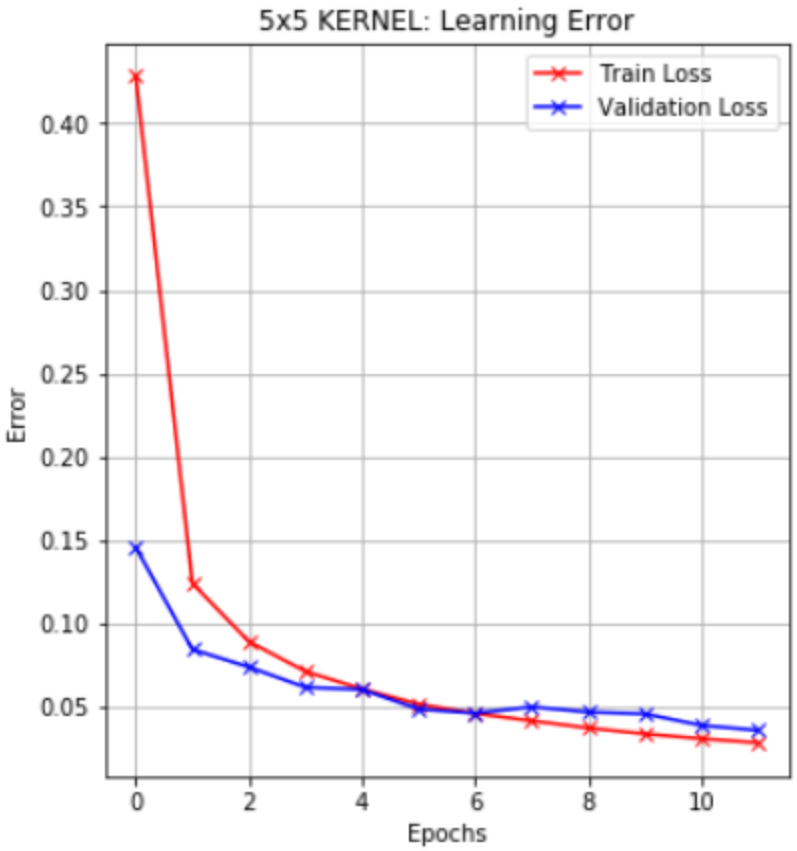


v. The testing error and testing accuracy were:

```
In [40]: score = model.evaluate(x_test, y_test, verbose=0)
         print('Test loss:', score[0])
         print('Test accuracy:', score[1])

         Test loss: 0.06357556160818785
         Test accuracy: 0.9796000123023987
```

3. **PART #2: Changing Kernel Dimensions (5x5)**
   i. Part #2 involved changing the kernels to 5x5 dimension and evaluating a plot for the learning error over epochs, testing loss, and testing accuracy.
   ii. Part #2's modifications made the following assumptions:
      1. Input size of the image is 28x28
      2. First Convolutional Layer contains:
         a. 6 Feature Maps
         b. Kernels are 5x5
         c. Stride of 1 (by default if omitted)
      3. First Max Pooling Layer
         a. Pooling is 2x2
         b. Stride of 1 (by default if omitted)
      4. Second Convolutional Layer
         a. 16 Feature Maps
         b. Kernels are 5x5
         c. Stride of 1 (by default if omitted)
      5. Second Max Pooling Layer
         a. Pooling is 2x2
         b. Stride of 1 (by default if omitted)
      6. First Hidden Layer
         a. 120 nodes
         b. ReLU activation function
      7. Second Hidden Layer
         a. 84 nodes
         b. ReLU activation function
      8. Softmax Layer with 10 output nodes
   iii. The following learning error over epochs plot was:



iv. The following testing error and testing accuracy was:

```
In [44]:  score = model2.evaluate(x_test, y_test, verbose=0)
          print('Test loss:', score[0])
          print('Test accuracy:', score[1])

          Test loss: 0.03586786532184342
          Test accuracy: 0.9886999726295471
```

4. PART #3: Changing Feature Maps

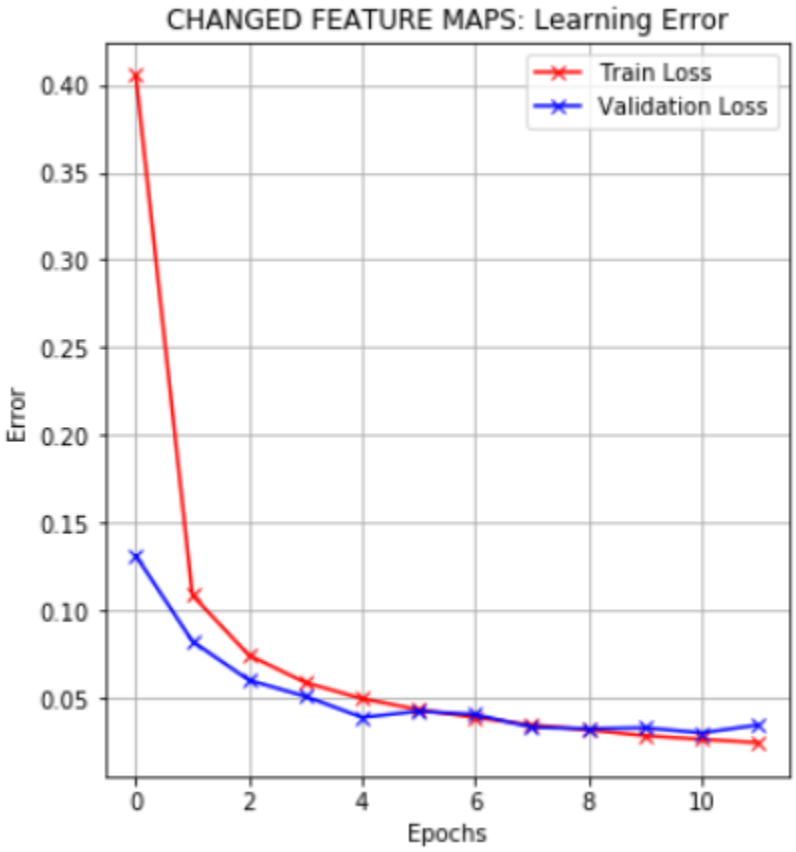
i. Part #2 involved changing the kernels to 5x5 dimension and evaluating a plot for the learning error over epochs, testing loss, and testing accuracy.

ii. Part #2's modifications made the following assumptions:

1. Input size of the image is 28x28
2. First Convolutional Layer contains:
   - **a. 8 Feature Maps**
   - **b. Kernels are 5x5**
   - c. Stride of 1 (by default if omitted)
   - **d. Added padding**
3. First Max Pooling Layer
   - a. Pooling is 2x2
   - b. Stride of 1 (by default if omitted)

4. Second Convolutional Layer
    a. **20 Feature Maps**
    b. **Kernels are 5x5**
    c. Stride of 1 (by default if omitted)
    d. **Added padding**
5. Second Max Pooling Layer
    a. Pooling is 2x2
    b. Stride of 1 (by default if omitted)
6. First Hidden Layer
    a. 120 nodes
    b. ReLU activation function
7. Second Hidden Layer
    a. 84 nodes
    b. ReLU activation function
8. Softmax Layer with 10 output nodes

iii. The following learning error over epochs plot was:



iv. The following testing error and testing accuracy was:

In [6]:
```
score = model3.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.0347609322346223
Test accuracy: 0.988099992275238
```

5. RESULTS

- **a.** The original boilerplate with the specified properties did worse than the changes to both the kernel dimensions and the feature maps
- **b.** Part #2, which changed the kernel dimensions to 5x5, had the highest test accuracy of 0.9886
- **c.** Part #3, which changed the feature maps to 8 and 20 respectively and padding as an additional hyperparameter, had the least testing loss of 0.034
- **d.** The accuracy and loss varies greatly on the proper combination of hyperparameters
- **e.** A general way to improve accuracy is to train over more epochs, although this increases your runtime