CSE 344 Computer Vision Homework 15

Name : Arka Sarkar
Roll Number : 20182222

In [73]:
```python
import numpy as np
import tensorflow as tf
import pandas as pd
from keras import layers
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Con
v2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D
import matplotlib.pyplot as plt
from keras.models import Model, load_model
```

Reading the Dataset

In [2]:
```python
train_df = pd.read_csv("dataset/mnist_train.csv")
test_df = pd.read_csv("dataset/mnist_test.csv")
```

In [3]:
```python
train_df
```

Out[3]:

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28x22	28x23	28x24	28x25	28x26	28x27	28x28
0	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59995	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59996	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59997	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59998	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59999	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

60000 rows × 785 columns

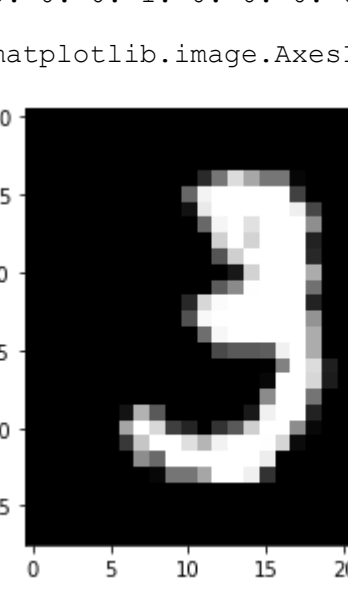
In [4]:
```python
train = np.array(train_df)
test = np.array(test_df)

Y_train = np.zeros((train.shape[0], 10))
X_train = []
Y_test = np.zeros((test.shape[0], 10))
X_test = []

for i in range(train.shape[0]):
    X_train.append(train[i,1:].reshape((28,28,1)))
    Y_train[i,train[i,0]] = 1

for i in range(test.shape[0]):
    X_test.append(test[i,1:].reshape((28,28,1)))
    Y_test[i,test[i,0]] = 1

X_train = np.array(X_train)
X_test = np.array(X_test)
print(X_test.shape)
print(X_train.shape)
print(Y_test.shape)
print(Y_train.shape)
```

```
(10000, 28, 28, 1)
(60000, 28, 28, 1)
(10000, 10)
(60000, 10)
```

In [8]:
```python
idx = 10
sample = X_train[idx]
print(Y_train[idx])
plt.imshow(sample, cmap = "gray")
```

```
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
```

Out[8]: `<matplotlib.image.AxesImage at 0x7f2bd3655190>`



In [19]:
```python
def LeNet5(input_shape = (28,28,1), stride  = [1,1], filters = [5,5], pooling = [2,2], activation = ["r
elu", "relu", "relu", "relu"], n_filters = [6,16], n_layers = [120,84], pooling_technique = "max"):


    s1,s2 = stride
    f1,f2 = filters
    p1,p2 = pooling

    a1,a2,a3,a4 = activation
    nf1, nf2 = n_filters
    nl1, nl2 = n_layers

    X_input = Input(input_shape)
    X = ZeroPadding2D((2,2))(X_input)

    X = Conv2D(filters =  nf1, kernel_size = (f1,f1), strides = (s1,s1), activation = a1, input_shape =
 (32,32,1))(X)
    if(pooling_technique == "max"):
        X = MaxPooling2D(( p1, p1), strides=(2, 2))(X)
    elif(pooling_technique == "avg"):
        X = AveragePooling2D(( p1, p1), strides=(2, 2))(X)
    else:
        raise Exception("Wrong Pooling technique")


    X = Conv2D(filters = nf2, kernel_size = (f2,f2), strides = (s2,s2), activation = a2)(X)
    if(pooling_technique == "max"):
        X = MaxPooling2D(( p2, p2), strides=(2, 2))(X)
    elif(pooling_technique == "avg"):
        X = AveragePooling2D(( p2, p2), strides=(2, 2))(X)
    else:
        raise Exception("Wrong Pooling technique")

    X = Flatten()(X)

    X = Dense(nl1, activation = a3)(X)

    X = Dense(nl2, activation = a4)(X)

    X = Dense(10, activation = 'softmax')(X)


    model  = Model(inputs = X_input, outputs = X, name='LeNet5')

    return model
```

Original LeNet 5

In [20]:
```python
model = LeNet5()
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs = 10, batch_size = 512)
preds = model.evaluate(X_test, Y_test)
preds = model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
Model: "LeNet5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_18 (Dense)             (None, 120)               48120
_________________________________________________________________
dense_19 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_20 (Dense)             (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
118/118 [==============================] - 1s 3ms/step - loss: 5.1802 - accuracy: 0.6288
Epoch 2/10
118/118 [==============================] - 0s 3ms/step - loss: 0.2296 - accuracy: 0.9339
Epoch 3/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1342 - accuracy: 0.9612
Epoch 4/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0882 - accuracy: 0.9730
Epoch 5/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0711 - accuracy: 0.9779
Epoch 6/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0551 - accuracy: 0.9827
Epoch 7/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0430 - accuracy: 0.9866
Epoch 8/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0351 - accuracy: 0.9888
Epoch 9/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0252 - accuracy: 0.9924
Epoch 10/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0205 - accuracy: 0.9941
313/313 [==============================] - 1s 1ms/step - loss: 0.0733 - accuracy: 0.9809
313/313 [==============================] - 0s 1ms/step - loss: 0.0733 - accuracy: 0.9809
Loss = 0.07327043265104294
Test Accuracy = 0.98089998960495
```

Modified LeNet5

- filter size
- Stride
- Pooling technique. For example, you can try pooling by computing the "determinant" value.
- Activation function. For example, you can try out some new activation functions like "x*sigmoid(x)".
- Number of layers
- Number of filters

Changing Filter Size

In [21]:
```python
model = LeNet5(filters = [5,7])
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs = 10, batch_size = 512)
preds = model.evaluate(X_test, Y_test)
preds = model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
Model: "LeNet5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
zero_padding2d_7 (ZeroPaddin (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 8, 8, 16)          4720
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 4, 4, 16)          0
_________________________________________________________________
flatten_7 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_21 (Dense)             (None, 120)               30840
_________________________________________________________________
dense_22 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_23 (Dense)             (None, 10)                850
=================================================================
Total params: 46,730
Trainable params: 46,730
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
118/118 [==============================] - 1s 3ms/step - loss: 4.7279 - accuracy: 0.5962
Epoch 2/10
118/118 [==============================] - 0s 4ms/step - loss: 0.2628 - accuracy: 0.9255
Epoch 3/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1423 - accuracy: 0.9572
Epoch 4/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1082 - accuracy: 0.9683
Epoch 5/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0839 - accuracy: 0.9747
Epoch 6/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0579 - accuracy: 0.9816
Epoch 7/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0534 - accuracy: 0.9826
Epoch 8/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0406 - accuracy: 0.9873
Epoch 9/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0314 - accuracy: 0.9905
Epoch 10/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0266 - accuracy: 0.9917
313/313 [==============================] - 1s 2ms/step - loss: 0.0904 - accuracy: 0.9736
313/313 [==============================] - 1s 2ms/step - loss: 0.0904 - accuracy: 0.9736
Loss = 0.09038401395082474
Test Accuracy = 0.9735999703407288
```

Changing Stride Size

In [22]:
```python
model = LeNet5(stride = [2,2])
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs = 10, batch_size = 512)
preds = model.evaluate(X_test, Y_test)
preds = model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
Model: "LeNet5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
zero_padding2d_8 (ZeroPaddin (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 14, 14, 6)         156
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 7, 7, 6)           0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 2, 2, 16)          2416
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 1, 1, 16)          0
_________________________________________________________________
flatten_8 (Flatten)          (None, 16)                0
_________________________________________________________________
dense_24 (Dense)             (None, 120)               2040
_________________________________________________________________
dense_25 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_26 (Dense)             (None, 10)                850
=================================================================
Total params: 15,626
Trainable params: 15,626
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
118/118 [==============================] - 1s 3ms/step - loss: 4.3474 - accuracy: 0.2799
Epoch 2/10
118/118 [==============================] - 0s 3ms/step - loss: 0.6701 - accuracy: 0.7772
Epoch 3/10
118/118 [==============================] - 0s 3ms/step - loss: 0.3617 - accuracy: 0.8851
Epoch 4/10
118/118 [==============================] - 0s 3ms/step - loss: 0.2577 - accuracy: 0.9204
Epoch 5/10
118/118 [==============================] - 0s 3ms/step - loss: 0.2088 - accuracy: 0.9354
Epoch 6/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1796 - accuracy: 0.9437
Epoch 7/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1694 - accuracy: 0.9470
Epoch 8/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1489 - accuracy: 0.9532
Epoch 9/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1304 - accuracy: 0.9594
Epoch 10/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1199 - accuracy: 0.9623
313/313 [==============================] - 0s 1ms/step - loss: 0.1240 - accuracy: 0.9621
313/313 [==============================] - 0s 1ms/step - loss: 0.1240 - accuracy: 0.9621
Loss = 0.12403564155101776
Test Accuracy = 0.9621000289916992
```

Changing Pooling Technique

In [23]:
```python
model = LeNet5(pooling_technique="avg")
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs = 10, batch_size = 512)
preds = model.evaluate(X_test, Y_test)
preds = model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
Model: "LeNet5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
zero_padding2d_9 (ZeroPaddin (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten_9 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_27 (Dense)             (None, 120)               48120
_________________________________________________________________
dense_28 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_29 (Dense)             (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
118/118 [==============================] - 1s 4ms/step - loss: 2.5328 - accuracy: 0.6309
Epoch 2/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1370 - accuracy: 0.9586
Epoch 3/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0798 - accuracy: 0.9756
Epoch 4/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0604 - accuracy: 0.9821
Epoch 5/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0459 - accuracy: 0.9854
Epoch 6/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0388 - accuracy: 0.9884
Epoch 7/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0304 - accuracy: 0.9905
Epoch 8/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0273 - accuracy: 0.9918
Epoch 9/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0232 - accuracy: 0.9932
Epoch 10/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0197 - accuracy: 0.9938
313/313 [==============================] - 1s 2ms/step - loss: 0.0462 - accuracy: 0.9847
313/313 [==============================] - 1s 2ms/step - loss: 0.0462 - accuracy: 0.9847
Loss = 0.046213697642087936
Test Accuracy = 0.9847000241279602
```

Changing Activation Function

In [24]:
```python
model = LeNet5(activation = ["sigmoid", "sigmoid", "sigmoid", "sigmoid"])
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs = 10, batch_size = 512)
preds = model.evaluate(X_test, Y_test)
preds = model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
Model: "LeNet5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_11 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
zero_padding2d_10 (ZeroPaddi (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
flatten_10 (Flatten)         (None, 400)               0
_________________________________________________________________
dense_30 (Dense)             (None, 120)               48120
_________________________________________________________________
dense_31 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_32 (Dense)             (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
118/118 [==============================] - 1s 4ms/step - loss: 2.2663 - accuracy: 0.1697
Epoch 2/10
118/118 [==============================] - 0s 3ms/step - loss: 0.8766 - accuracy: 0.7940
Epoch 3/10
118/118 [==============================] - 0s 4ms/step - loss: 0.3179 - accuracy: 0.9279
Epoch 4/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1909 - accuracy: 0.9515
Epoch 5/10
118/118 [==============================] - 0s 4ms/step - loss: 0.1383 - accuracy: 0.9626
Epoch 6/10
118/118 [==============================] - 0s 3ms/step - loss: 0.1168 - accuracy: 0.9673
Epoch 7/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0987 - accuracy: 0.9719
Epoch 8/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0866 - accuracy: 0.9750
Epoch 9/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0759 - accuracy: 0.9784
Epoch 10/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0687 - accuracy: 0.9806
313/313 [==============================] - 1s 3ms/step - loss: 0.0646 - accuracy: 0.9794
313/313 [==============================] - 1s 2ms/step - loss: 0.0646 - accuracy: 0.9794
Loss = 0.06456702202558517
Test Accuracy = 0.9793999791145325
```

Changing the Number of Layers

In [25]:
```python
model = LeNet5(n_layers=[240,168])
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs = 10, batch_size = 512)
preds = model.evaluate(X_test, Y_test)
preds = model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
Model: "LeNet5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_12 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
zero_padding2d_11 (ZeroPaddi (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
flatten_11 (Flatten)         (None, 400)               0
_________________________________________________________________
dense_33 (Dense)             (None, 240)               96240
_________________________________________________________________
dense_34 (Dense)             (None, 168)               40488
_________________________________________________________________
dense_35 (Dense)             (None, 10)                1690
=================================================================
Total params: 140,990
Trainable params: 140,990
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
118/118 [==============================] - 1s 3ms/step - loss: 3.4541 - accuracy: 0.6749
Epoch 2/10
118/118 [==============================] - 0s 4ms/step - loss: 0.1500 - accuracy: 0.9537
Epoch 3/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0844 - accuracy: 0.9737
Epoch 4/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0554 - accuracy: 0.9825
Epoch 5/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0416 - accuracy: 0.9873
Epoch 6/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0307 - accuracy: 0.9902
Epoch 7/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0216 - accuracy: 0.9939
Epoch 8/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0153 - accuracy: 0.9958
Epoch 9/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0106 - accuracy: 0.9972
Epoch 10/10
118/118 [==============================] - 0s 3ms/step - loss: 0.0091 - accuracy: 0.9974
313/313 [==============================] - 0s 1ms/step - loss: 0.0708 - accuracy: 0.9816
313/313 [==============================] - 1s 2ms/step - loss: 0.0708 - accuracy: 0.9816
Loss = 0.07083163410425186
Test Accuracy = 0.9815999865531921
```

Changing the number of Filters

In [26]:
```python
model = LeNet5(n_filters=[12,32])
model.summary()
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs = 10, batch_size = 512)
preds = model.evaluate(X_test, Y_test)
preds = model.evaluate(X_test, Y_test)
print ("Loss = " + str(preds[0]))
print ("Test Accuracy = " + str(preds[1]))
```

```
Model: "LeNet5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
zero_padding2d_12 (ZeroPaddi (None, 32, 32, 1)         0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 28, 28, 12)        312
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 14, 14, 12)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 10, 10, 32)        9632
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 5, 5, 32)          0
_________________________________________________________________
flatten_12 (Flatten)         (None, 800)               0
_________________________________________________________________
dense_36 (Dense)             (None, 120)               96120
_________________________________________________________________
dense_37 (Dense)             (None, 84)                10164
_________________________________________________________________
dense_38 (Dense)             (None, 10)                850
=================================================================
Total params: 117,078
Trainable params: 117,078
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
118/118 [==============================] - 1s 5ms/step - loss: 8.3756 - accuracy: 0.6087
Epoch 2/10
118/118 [==============================] - 1s 4ms/step - loss: 0.2199 - accuracy: 0.9381
Epoch 3/10
118/118 [==============================] - 1s 4ms/step - loss: 0.1259 - accuracy: 0.9629
Epoch 4/10
118/118 [==============================] - 1s 4ms/step - loss: 0.0873 - accuracy: 0.9733
Epoch 5/10
118/118 [==============================] - 0s 4ms/step - loss: 0.0639 - accuracy: 0.9809
Epoch 6/10
118/118 [==============================] - 1s 4ms/step - loss: 0.0493 - accuracy: 0.9842
Epoch 7/10
118/118 [==============================] - 1s 4ms/step - loss: 0.0370 - accuracy: 0.9882
Epoch 8/10
118/118 [==============================] - 1s 4ms/step - loss: 0.0288 - accuracy: 0.9910
Epoch 9/10
118/118 [==============================] - 1s 4ms/step - loss: 0.0266 - accuracy: 0.9920
Epoch 10/10
118/118 [==============================] - 1s 4ms/step - loss: 0.0216 - accuracy: 0.9925
313/313 [==============================] - 1s 2ms/step - loss: 0.0688 - accuracy: 0.9809
313/313 [==============================] - 1s 2ms/step - loss: 0.0688 - accuracy: 0.9809
Loss = 0.06884230673313141
Test Accuracy = 0.98089998960495
```

Model	Test Accuracy	Test Loss
Original LeNet5	0.98089998960495	0.07327043265104294
LeNet with filter size (5,5) and (7,7)	0.9735999703407288	0.09038401395082474
LeNet with stride size (2,2) and (2,2)	0.9621000289916992	0.12403564155101776
LeNet with Average pooling	0.9847000241279602	0.046213697642087936
LeNet with Sigmoid Activation	0.9793999791145325	0.06456702202558517
LeNet with number of layers 240 and 168	0.9815999865531921	0.07083163410425186
LeNet with number of filters 12,32	0.98089998960495	0.06884230673313141