Basic Imports

```
import tensorflow as tf
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt
from tensorflow.examples.tutorials.mnist import input_data
```

Importing the MNIST dataset

```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(type(x_train))
x_train = x_train.astype('float32')
x_train /= 255
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
# y_train = y_train.reshape(y_train.shape[0],1)

# splitSz = x_train.shape[0]
# splitSz = int(1.0*splitSz)

# print(y_train.shape)
# x_train, x_test = x_train[:splitSz,:], x_train[splitSz:,:]
# y_train, y_test = y_train[:splitSz], y_train[splitSz:]

print(y_train.shape)
print(y_train[:10])
print(x_test.shape)
```

```
<class 'numpy.ndarray'>
(60000,)
[5 0 4 1 9 2 1 3 1 4]
(10000, 28, 28, 1)
```

Model

1. 7x7 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. fully connected layer with 1024 output units.
6. ReLU Activation Layer.

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation="relu"))

```

```
model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/r
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        320
_________________________________________________________________
batch_normalization_v1 (Batc (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              6423552
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 6,434,250
Trainable params: 6,434,186
Non-trainable params: 64
_________________________________________________________________
```

```
print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=10)
```

```
(60000,)
Epoch 1/10
60000/60000 [==============================] - 226s 4ms/sample - loss: 0.2494 - acc: 0
Epoch 2/10
60000/60000 [==============================] - 226s 4ms/sample - loss: 0.0396 - acc: 0
Epoch 3/10
60000/60000 [==============================] - 226s 4ms/sample - loss: 0.0235 - acc: 0
Epoch 4/10
60000/60000 [==============================] - 226s 4ms/sample - loss: 0.0178 - acc: 0
Epoch 5/10
60000/60000 [==============================] - 227s 4ms/sample - loss: 0.0176 - acc: 0
Epoch 6/10
60000/60000 [==============================] - 226s 4ms/sample - loss: 0.0135 - acc: 0
Epoch 7/10
60000/60000 [==============================] - 227s 4ms/sample - loss: 0.0104 - acc: 0
Epoch 8/10
60000/60000 [==============================] - 230s 4ms/sample - loss: 0.0123 - acc: 0
Epoch 9/10
60000/60000 [==============================] - 227s 4ms/sample - loss: 0.0089 - acc: 0
Epoch 10/10
60000/60000 [==============================] - 227s 4ms/sample - loss: 0.0089 - acc: 0
<tensorflow.python.keras.callbacks.History at 0x7f0b32470748>
```

```
score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
10000/10000 [==============================] - 8s 805us/sample - loss: 4.7237 - acc: 0

 Test Acc =  0.7065
```

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(32, (3, 3), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation="relu"))
model.add(tf.keras.layers.Dense(1024, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()

print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=5)

score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
```

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(32, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()

print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=5)

score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 28, 28, 32)        1600
_________________________________________________________________
batch_normalization_v1_2 (Ba (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1024)              6423552
_________________________________________________________________
dense_6 (Dense)              (None, 10)                10250
=================================================================
Total params: 6,435,530
Trainable params: 6,435,466
Non-trainable params: 64
_________________________________________________________________
(60000,)
Epoch 1/5
60000/60000 [==============================] - 267s 4ms/sample - loss: 0.2479 - acc: 0
Epoch 2/5
60000/60000 [==============================] - 268s 4ms/sample - loss: 0.0415 - acc: 0
Epoch 3/5
60000/60000 [==============================] - 268s 4ms/sample - loss: 0.0266 - acc: 0
Epoch 4/5
60000/60000 [==============================] - 267s 4ms/sample - loss: 0.0235 - acc: 0
Epoch 5/5
60000/60000 [==============================] - 266s 4ms/sample - loss: 0.0196 - acc: 0
10000/10000 [==============================] - 12s 1ms/sample - loss: 0.3357 - acc: 0.

 Test Acc =  0.9791
```

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(64, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()

print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=5)

score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/r
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        3200
_________________________________________________________________
batch_normalization_v1 (Batc (None, 28, 28, 64)        256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 12544)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              12846080
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 12,859,786
Trainable params: 12,859,658
Non-trainable params: 128
_________________________________________________________________
(60000,)
Epoch 1/5
60000/60000 [==============================] - 513s 9ms/sample - loss: 0.3795 - acc: 0
Epoch 2/5
60000/60000 [==============================] - 505s 8ms/sample - loss: 0.0364 - acc: 0
Epoch 3/5
60000/60000 [==============================] - 503s 8ms/sample - loss: 0.0278 - acc: 0
Epoch 4/5
60000/60000 [==============================] - 507s 8ms/sample - loss: 0.0240 - acc: 0
Epoch 5/5
60000/60000 [==============================] - 507s 8ms/sample - loss: 0.0205 - acc: 0
10000/10000 [==============================] - 18s 2ms/sample - loss: 0.3191 - acc: 0.

 Test Acc =  0.9801
```

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(32, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.Conv2D(32, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()

print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=5)

score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        1600
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 32)        50208
_________________________________________________________________
batch_normalization_v1_1 (Ba (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              6423552
_________________________________________________________________
dense_3 (Dense)              (None, 10)                10250
=================================================================
Total params: 6,485,738
Trainable params: 6,485,674
Non-trainable params: 64
_________________________________________________________________
(60000,)
Epoch 1/5
60000/60000 [==============================] - 806s 13ms/sample - loss: 0.1211 - acc:
Epoch 2/5
60000/60000 [==============================] - 783s 13ms/sample - loss: 0.0459 - acc:
Epoch 3/5
60000/60000 [==============================] - 799s 13ms/sample - loss: 0.0294 - acc:
Epoch 4/5
60000/60000 [==============================] - 791s 13ms/sample - loss: 0.0268 - acc:
Epoch 5/5
60000/60000 [==============================] - 787s 13ms/sample - loss: 0.0212 - acc:
10000/10000 [==============================] - 33s 3ms/sample - loss: 0.3530 - acc: 0.

 Test Acc =  0.9778
```

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(32, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.BatchNormalization())
```

```
model.add(tf.keras.layers.Conv2D(32, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()

print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=5)

score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 32)        1600
_________________________________________________________________
batch_normalization_v1_2 (Ba (None, 28, 28, 32)        128
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 32)        50208
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              6423552
_________________________________________________________________
dense_5 (Dense)              (None, 10)                10250
=================================================================
Total params: 6,485,738
Trainable params: 6,485,674
Non-trainable params: 64
_________________________________________________________________
(60000,)
Epoch 1/5
60000/60000 [==============================] - 772s 13ms/sample - loss: 1.5519 - acc:
Epoch 2/5
60000/60000 [==============================] - 760s 13ms/sample - loss: 0.0421 - acc:
Epoch 3/5
60000/60000 [==============================] - 760s 13ms/sample - loss: 0.0281 - acc:
Epoch 4/5
60000/60000 [==============================] - 773s 13ms/sample - loss: 0.0223 - acc:
Epoch 5/5
60000/60000 [==============================] - 771s 13ms/sample - loss: 0.0195 - acc:
10000/10000 [==============================] - 33s 3ms/sample - loss: 0.3756 - acc: 0.

 Test Acc =  0.9765
```

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(64, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.Conv2D(32, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
```

```
model.add(tf.keras.layers.Dense(1024, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()

print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=5)

score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/r
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        3200
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        100384
_________________________________________________________________
batch_normalization_v1 (Batc (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              6423552
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 6,537,514
Trainable params: 6,537,450
Non-trainable params: 64
_________________________________________________________________
(60000,)
Epoch 1/5
60000/60000 [==============================] - 1285s 21ms/sample - loss: 0.1055 - acc:
Epoch 2/5
60000/60000 [==============================] - 1302s 22ms/sample - loss: 0.0434 - acc:
Epoch 3/5
60000/60000 [==============================] - 1298s 22ms/sample - loss: 0.0350 - acc:
Epoch 4/5
60000/60000 [==============================] - 1282s 21ms/sample - loss: 0.0287 - acc:
Epoch 5/5
60000/60000 [==============================] - 1263s 21ms/sample - loss: 0.0213 - acc:
10000/10000 [==============================] - 52s 5ms/sample - loss: 0.2341 - acc: 0.

 Test Acc =  0.9853
```

```
1
```

Link

1. input layer: [.,784]
2. layer: Conv1 -> ReLu -> MaxPool: [.,14,14,36]
3. layer: Conv2 -> ReLu -> MaxPool: [.,7,7,36]
4. layer: Conv3 -> ReLu -> MaxPool: [.,4,4,36]
5. layer: FC -> ReLu: [.,576]
6. output layer: FC -> ReLu: [.,10]

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(36, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Conv2D(36, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Conv2D(36, (7, 7), strides=1, activation='relu', padding="same"
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
# model.add(tf.keras.layers.BatchNormalization())
# model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation="relu"))

model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
model.summary()

print(y_train.shape)

model.fit(x_train, y_train, batch_size=64, epochs=5)

score = model.evaluate(x_test, y_test)
print('\n', 'Test Acc = ', score[1])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 28, 28, 64)        3200
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 32)        100384
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 32)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 32)          50208
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 32)          0
```

```python
model2 = tf.keras.Sequential()
model2.add(tf.keras.layers.Conv2D(8, (7, 7), strides=1, activation='relu', padding="same"
model2.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
# model2.add(tf.keras.layers.Conv2D(36, (7, 7), strides=1, activation='relu', padding="sa
# model2.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
# model2.add(tf.keras.layers.Conv2D(36, (7, 7), strides=1, activation='relu', padding="sa
# model2.add(tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=2))
model2.add(tf.keras.layers.Flatten())
model2.add(tf.keras.layers.Dense(1024, activation="relu"))

model2.add(tf.keras.layers.Dense(10, activation='softmax'))

model2.compile(optimizer='adam', loss='sparse_categorical_crossentropy')
model2.summary()

print(y_train.shape)

some = model2.fit(x_train, y_train, batch_size=64, epochs=1)

score = model2.evaluate(x_test, y_test)
pred = model2.predict(x_test)
# print('\n', 'Test Acc = ', score[1])
print(score)
# print('\n'," Predictions: ", pred)
print(some)
```

•••

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 28, 28, 8)         400
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 8)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1568)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              1606656
_________________________________________________________________
dense_5 (Dense)              (None, 10)                10250
=================================================================
Total params: 1,617,306
Trainable params: 1,617,306
```

```
1
```

```
(60000,)
42112/60000 [====================>.........] - ETA: 26s - loss: 0.1809
```