Rohan Nyati

500075940

R177219148

Batch - 5 (Ai & Ml)

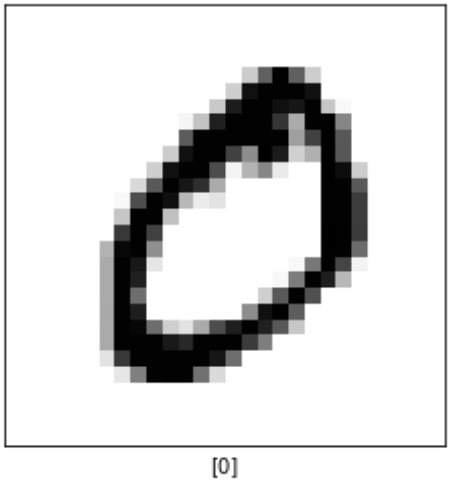
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
# Load themnist pre-shuffled train data and test data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
print("x_train shape:", x_train.shape, "y_train shape:", y_train.shape,"x_test shape:", x
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
x_train shape: (60000, 28, 28) y_train shape: (60000,) x_test shape: (10000, 28, 28)
```

```
np.random.seed(0)
```

```
# Show one of the images from the training dataset
plt.xticks([])
plt.yticks([])
plt.xlabel([y_train[1]])
plt.imshow(x_train[1],cmap=plt.cm.binary)
```

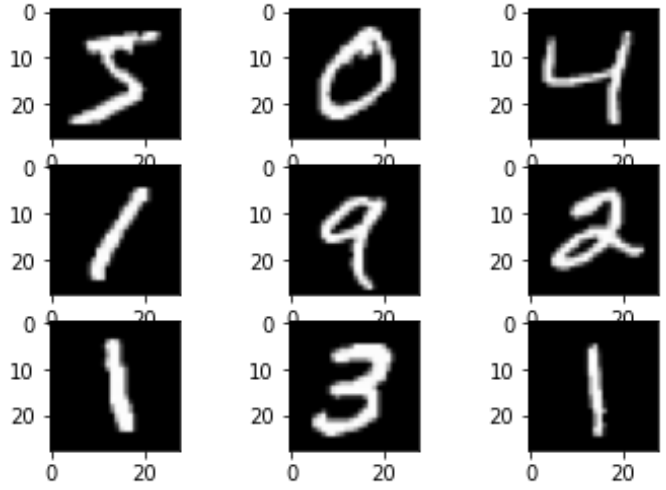
```
<matplotlib.image.AxesImage at 0x7f6fa74f8490>
```




```
from tensorflow.keras.datasets import mnist
from matplotlib import pyplot as plt
# load dataset
(trainX, trainy), (testX, testy) = mnist.load_data()
# summarize loaded dataset
print('Train: X=%s, y=%s' % (trainX.shape, trainy.shape))
print('Test: X=%s, y=%s' % (testX.shape, testy.shape))
```

```
# plot first few images
for i in range(9):
  plt.subplot(330 + 1 + i)
  plt.imshow(trainX[i],cmap=plt.get_cmap('gray'))
# show the figure
plt.show()
```

```
Train: X=(60000, 28, 28), y=(60000,)
Test: X=(10000, 28, 28), y=(10000,)
```



```
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
```

```
model = tf.keras.Sequential()
# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(2,2),strides=(1, 1), padding='s
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(2,2),strides=(1, 1), padding='s
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(10, activation='softmax'))
# Take a look at the model summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 64)        320

 max_pooling2d (MaxPooling2D  (None, 14, 14, 64)       0
 )
```

```
 dropout (Dropout)           (None, 14, 14, 64)        0

 conv2d_1 (Conv2D)           (None, 14, 14, 32)        8224

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 32)         0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 32)          0

 flatten (Flatten)           (None, 1568)              0

 dense (Dense)               (None, 256)               401664

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 412,778
Trainable params: 412,778
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
x_train = x_train.reshape(-1,28, 28,1)#Reshape for CNN
x_test = x_test.reshape(-1,28, 28, 1)
```

```
model_log=model.fit(x_train, y_train,
          batch_size=60,
          epochs=10,
          verbose=1,
          validation_split=.3)
```

```
Epoch 1/10
700/700 [==============================] - 58s 82ms/step - loss: 0.3924 - accuracy:
Epoch 2/10
700/700 [==============================] - 51s 73ms/step - loss: 0.1460 - accuracy:
Epoch 3/10
700/700 [==============================] - 50s 72ms/step - loss: 0.1133 - accuracy:
Epoch 4/10
700/700 [==============================] - 50s 72ms/step - loss: 0.0919 - accuracy:
Epoch 5/10
700/700 [==============================] - 50s 72ms/step - loss: 0.0821 - accuracy:
Epoch 6/10
700/700 [==============================] - 50s 72ms/step - loss: 0.0732 - accuracy:
Epoch 7/10
700/700 [==============================] - 50s 72ms/step - loss: 0.0661 - accuracy:
Epoch 8/10
```

```
700/700 [==============================] - 50s 72ms/step - loss: 0.0593 - accuracy:
Epoch 9/10
700/700 [==============================] - 50s 72ms/step - loss: 0.0573 - accuracy:
Epoch 10/10
700/700 [==============================] - 50s 71ms/step - loss: 0.0527 - accuracy:
```

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test, verbose=0)
# Print test accuracy
print('\n', 'Test accuracy:', score[1])
```

```
 Test accuracy: 0.9905999898910522
```

```
predictions = model.predict(x_test)
predictions[0]
```

```
array([5.1499034e-13, 1.4007135e-09, 5.1558011e-09, 2.6321667e-08,
       1.6978053e-12, 1.6660723e-12, 5.8933968e-18, 9.9999988e-01,
       2.5531477e-10, 1.3288975e-07], dtype=float32)
```

```
np.argmax(predictions[0])
```

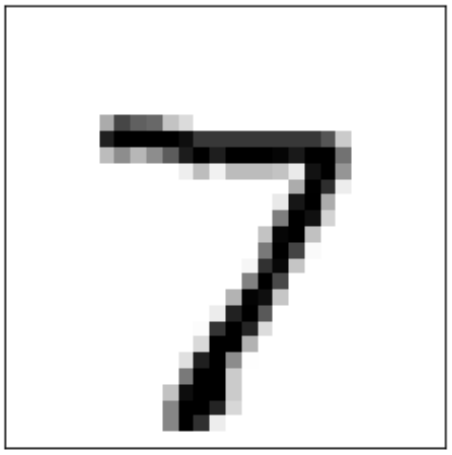
```
7
```

```
x_test = x_test.reshape(-1,28, 28)
x_test.shape
```

```
(10000, 28, 28)
```

```
# Show one of the images from the test dataset
plt.xticks([])
plt.yticks([])
plt.xlabel([y_train[0]])
plt.imshow(x_test[0],cmap=plt.cm.binary)
```

```
<matplotlib.image.AxesImage at 0x7f6fa2d74450>
```



```
[5]
```