```
import tensorflow as tf # Import tensorflow library
import matplotlib.pyplot as plt # Import matplotlib library
```

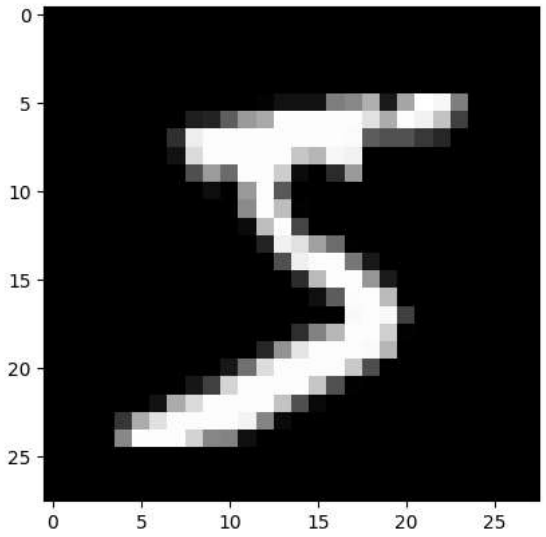
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

learn = tf.compat.v1.estimator

tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
```

```
mnist = tf.keras.datasets.mnist # Object of the MNIST dataset
(x_train, y_train),(x_test, y_test) = mnist.load_data() # Load data
```

```
plt.imshow(x_train[0], cmap="gray") # Import the image
plt.show() # Plot the image
```




```
# Normalize the train dataset
x_train = tf.keras.utils.normalize(x_train, axis=1)
# Normalize the test dataset
x_test = tf.keras.utils.normalize(x_test, axis=1)
```

```
#Build the model object
model = tf.keras.models.Sequential()
# Add the Flatten Layer
model.add(tf.keras.layers.Flatten())
# Build the input and the hidden layers
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
# Build the output layer
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax))
```

```
# Compile the model
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
```

```
model.fit(x=x_train, y=y_train, epochs=5) # Start training process
```

```
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2638 - accuracy: 0.9223
Epoch 2/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1081 - accuracy: 0.9662
Epoch 3/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0730 - accuracy: 0.9770
Epoch 4/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0546 - accuracy: 0.9822
Epoch 5/5
```

```
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0422 - accuracy: 0.9859
<keras.src.callbacks.History at 0x7f63054355d0>
```

```
# Evaluate the model performance
test_loss, test_acc = model.evaluate(x=x_test, y=y_test)
# Print out the model accuracy
print('\nTest accuracy:', test_acc)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0816 - accuracy: 0.9761

Test accuracy: 0.9761000275611877
```

```
predictions = model.predict([x_test]) # Make prediction
```

```
313/313 [==============================] - 1s 3ms/step
```

```
print(np.argmax(predictions[1000])) # Print out the number
```

```
9
```

```
plt.imshow(x_test[1000], cmap="gray") # Import the image
plt.show() # Show the image
```