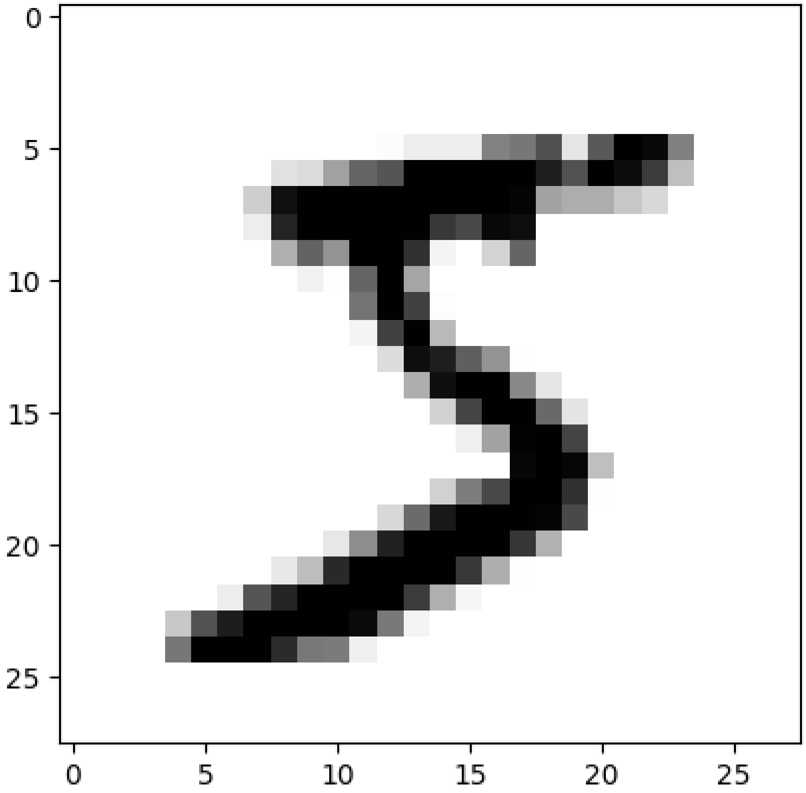
```
In [3]: import tensorflow as tf
```

```
In [ ]: mnist = tf.keras.datasets.mnist
        (x_train, y_train), (x_test, y_test) = mnist.load_data()
        x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
In [5]: import matplotlib.pyplot as plt
        plt.imshow(x_train[0], cmap=plt.cm.binary)
```

```
Out[5]: <matplotlib.image.AxesImage at 0x24331c22490>
```



```
In [6]: model = tf.keras.models.Sequential([
          tf.keras.layers.Flatten(input_shape=(28, 28)),
          tf.keras.layers.Dense(128, activation='relu'),
          tf.keras.layers.Dropout(0.2),
          tf.keras.layers.Dense(10)
        ])
```

```
In [7]: predictions = model(x_train[:1]).numpy()
        print(predictions)
```

```
[[-0.15393575  0.5745611   0.23484835 -0.08647239 -0.00528421 -0.36834687
  -0.69348824  0.46609634  0.0031645  -0.29634845]]
```

In [8]:

```
tf.nn.softmax(predictions).numpy()
```

Out[8]:

```
array([[0.082894  , 0.17175338, 0.12228408, 0.08867926, 0.09617931,
        0.06689683, 0.04832796, 0.15409893, 0.09699535, 0.07189093]],
      dtype=float32)
```

In [9]:

```
#convert output probabilities to a single prediction
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

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

In [10]:

```
model.fit(x_train, y_train, epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2930 - acc
uracy: 0.9146
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1401 - acc
uracy: 0.9580
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.1057 - acc
uracy: 0.9678
Epoch 4/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0863 - acc
uracy: 0.9734
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0744 - acc
uracy: 0.9769
```

Out[10]:

```
<keras.callbacks.History at 0x24333e98a90>
```

In [11]:

```
model.evaluate(x_test,  y_test, verbose=2)
```

```
313/313 - 1s - loss: 0.0749 - accuracy: 0.9773 - 562ms/epoch - 2ms/step
```

Out[11]:

```
[0.0748772993683815, 0.9772999882698059]
```

In [12]:

```
probability_model = tf.keras.Sequential([
  model,
  tf.keras.layers.Softmax()
])
```

In [14]:

```
print (probability_model(x_test[:5]))
```

```
tf.Tensor(
[[1.88280183e-08 5.15292975e-10 3.06334641e-06 2.51267717e-04
  5.36535226e-12 2.30961295e-08 5.32454804e-14 9.99742806e-01
  6.51713847e-08 2.76385776e-06]
 [6.54264056e-07 1.00209814e-04 9.99838710e-01 4.07953194e-05
  9.04130930e-15 1.65799938e-05 1.48702611e-07 8.70862789e-13
  2.92654840e-06 6.28578162e-11]
 [5.53513217e-08 9.99066651e-01 2.31224622e-04 1.61659500e-05
  2.98411414e-05 2.94180700e-06 2.11715778e-05 4.38720250e-04
  1.90352614e-04 2.90085632e-06]
 [9.99890327e-01 1.57914889e-10 1.35624941e-05 7.71929365e-08
  2.36984215e-06 1.46909388e-05 7.38265589e-05 1.51520396e-06
  1.23019079e-08 3.53591327e-06]
 [1.83616783e-06 2.47155019e-10 2.72418720e-06 4.14668939e-08
  9.98704433e-01 3.01004479e-06 3.81676273e-06 1.90626051e-05
  5.73740067e-07 1.26453629e-03]], shape=(5, 10), dtype=float32)
```

In [15]:

```
print (y_test[:5])
```

```
[7 2 1 0 4]
```