```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)

     TensorFlow version: 2.15.0
```

```
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

     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
     11490434/11490434 [==============================] - 0s 0us/step
```

```
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)
])
```

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

     array([[ 0.36810514, -0.18530923,  0.16628006, -0.06777699,  0.15742159,
              0.48715538, -0.202894  ,  0.11354117, -0.04802385, -0.41750562]],
           dtype=float32)
```

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

     array([[0.13457096, 0.07737605, 0.10997649, 0.08702623, 0.10900656,
             0.1515843 , 0.0760273 , 0.10432674, 0.08876236, 0.06134299]],
           dtype=float32)
```

```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

```
loss_fn(y_train[:1], predictions).numpy()

     1.8866134
```

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

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

     Epoch 1/5
     1875/1875 [==============================] - 7s 3ms/step - loss: 0.3019 - accuracy: 0.9128
     Epoch 2/5
     1875/1875 [==============================] - 7s 4ms/step - loss: 0.1457 - accuracy: 0.9566
     Epoch 3/5
     1875/1875 [==============================] - 6s 3ms/step - loss: 0.1114 - accuracy: 0.9664
     Epoch 4/5
     1875/1875 [==============================] - 7s 4ms/step - loss: 0.0894 - accuracy: 0.9722
     Epoch 5/5
     1875/1875 [==============================] - 7s 4ms/step - loss: 0.0757 - accuracy: 0.9758
     <keras.src.callbacks.History at 0x796276abebc0>
```

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

     313/313 - 1s - loss: 0.0763 - accuracy: 0.9786 - 597ms/epoch - 2ms/step
     [0.07630588114261627, 0.978600025177002]
```

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

```
probability_model(x_test[:5])

     <tf.Tensor: shape=(5, 10), dtype=float32, numpy=
     array([[1.88657452e-07, 2.60073421e-08, 1.19474744e-05, 1.01849735e-04,
             2.02302342e-10, 2.56837041e-07, 4.05992412e-14, 9.99878645e-01,
             3.06588021e-07, 6.82445034e-06],
            [2.89111330e-08, 2.29390389e-05, 9.99962091e-01, 6.15386671e-06,
             4.15520737e-15, 3.63905224e-06, 1.94573158e-08, 1.81269149e-11,
             5.13773830e-06, 1.57895094e-15],
               [1.08174646e-07, 9.98168230e-01, 1.38259813e-04, 4.58060413e-06,
                2.48204597e-05, 1.03600169e-05, 9.30210444e-05, 1.29460462e-03,
                2.64401606e-04, 1.66463633e-06],
               [9.99949217e-01, 1.79552007e-09, 3.49895527e-05, 1.87561000e-07,
                1.78540915e-08, 8.14907310e-07, 1.35159689e-05, 7.98568863e-07,
                1.07971729e-07, 4.08209189e-07],
               [8.30505269e-06, 1.55753133e-09, 1.50474364e-06, 2.90527229e-08,
                9.99373615e-01, 7.10203778e-07, 7.93175332e-06, 7.46133901e-06,
                6.96371899e-06, 5.93431178e-04]], dtype=float32)>
```

Start coding or generate with AI.