Homework

- Experiment with the code
- Add more hidden layers / nodes
- Change the activation function for the hidden layers
- Change the optimizers
- Change the number of epochs
- See the results

Import and load a dataset.

```
import tensorflow as tf
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
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

Create a neural network model, compile, train, and evaluate the model.

```
[2] model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape = (28,28)),
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(10, activation = 'softmax')
    ])
```

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

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

```
Epoch 1/5
1875/1875 [==============================] - 5s 2ms/step - loss: 0.2335 - accuracy: 0.9317
Epoch 2/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0984 - accuracy: 0.9700
Epoch 3/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0683 - accuracy: 0.9784
Epoch 4/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0513 - accuracy: 0.9842
Epoch 5/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0408 - accuracy: 0.9863
313/313 [==============================] - 1s 1ms/step - loss: 0.0963 - accuracy: 0.9722
[0.09628631174564362, 0.9721999764442444]
```

Add more hidden layers with different hidden nodes, and train, evaluate the model.

```
[4] model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape = (28,28)),
        tf.keras.layers.Dense(256, activation = 'relu'),
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(64, activation = 'relu'),
        tf.keras.layers.Dense(10, activation = 'softmax')
    ])
```

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

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

```
Epoch 1/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2101 - accuracy: 0.9371
Epoch 2/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0906 - accuracy: 0.9717
Epoch 3/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0646 - accuracy: 0.9793
Epoch 4/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0502 - accuracy: 0.9836
Epoch 5/5
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0416 - accuracy: 0.9869
313/313 [==============================] - 1s 2ms/step - loss: 0.0780 - accuracy: 0.9770
[0.0780177041888237, 0.9769999980926514]
```

Change the activation function, train and evaluate the model.

```
[6] model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape = (28,28)),
        tf.keras.layers.Dense(256, activation = 'relu'),
        tf.keras.layers.Dense(256, activation = 'sigmoid'),
        tf.keras.layers.Dense(128, activation = 'relu'),
        tf.keras.layers.Dense(128, activation = 'sigmoid'),
        tf.keras.layers.Dense(10, activation = 'softmax')
    ])
```

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

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

```
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2827 - accuracy: 0.9163
Epoch 2/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1064 - accuracy: 0.9686
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0753 - accuracy: 0.9775
Epoch 4/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0546 - accuracy: 0.9836
Epoch 5/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0444 - accuracy: 0.9865
313/313 [==============================] - 1s 2ms/step - loss: 0.0990 - accuracy: 0.9746
[0.09900996834039688, 0.9746000170707703]
```

Change the optimizer function, train and evaluate the model.

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

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

```
Epoch 1/5
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0294 - accuracy: 0.9915
Epoch 2/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0241 - accuracy: 0.9933
Epoch 3/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0220 - accuracy: 0.9942
Epoch 4/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0218 - accuracy: 0.9947
Epoch 5/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0190 - accuracy: 0.9952
313/313 [==============================] - 1s 2ms/step - loss: 0.1134 - accuracy: 0.9805
[0.1133800521492958, 0.9804999828338623]
```

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

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

```
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0138 - accuracy: 0.9965
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0131 - accuracy: 0.9966
Epoch 3/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0125 - accuracy: 0.9968
Epoch 4/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0119 - accuracy: 0.9969
Epoch 5/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0115 - accuracy: 0.9970
313/313 [==============================] - 1s 2ms/step - loss: 0.1073 - accuracy: 0.9822
[0.10727093368768692, 0.982200026512146]
```

Change the loss function, train and evaluate the model.

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

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

```
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 27.3046 - accuracy: 0.1032
Epoch 2/5
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.1031
Epoch 3/5
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.1009
Epoch 4/5
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.1017
Epoch 5/5
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0989
313/313 [==============================] - 1s 2ms/step - loss: 27.2503 - accuracy: 0.0900
[27.25031280517578, 0.09000000357627869]
```

Increase the epoch, train and evaluate the model.

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

model.fit(x_train, y_train, epochs = 10)
model.evaluate(x_test, y_test)
```

```
Epoch 1/10
1875/1875 [==============================] - 8s 4ms/step - loss: 27.3046 - accuracy: 0.0996
Epoch 2/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0986
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.1001
Epoch 4/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0979
Epoch 5/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0998
Epoch 6/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0993
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0988
Epoch 8/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0991
Epoch 9/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0992
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 27.3046 - accuracy: 0.0978
313/313 [==============================] - 1s 2ms/step - loss: 27.2503 - accuracy: 0.0980
[27.25031280517578, 0.09799999743700027]
```