In [1]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
```

In [2]:
```
# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize pixel values to [0, 1]
x_train, x_test = x_train / 255.0, x_test / 255.0

# One-hot encode the labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mn
ist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

In [3]:
```
# Define the model
model = Sequential([
    Flatten(input_shape=(28, 28)),       # Flatten 28x28 input images into 1D vecto
    Dense(128, activation='relu'),       # Hidden layer with 128 neurons
    Dense(10, activation='softmax')      # Output layer with 10 neurons (one per cl
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: Us
erWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer in the m
odel instead.
  super().__init__(**kwargs)
```

In [4]:
```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy
```

In [5]:
```
# Train the model
history = model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.2
```

```
Epoch 1/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8622 - loss: 0.4747 - val_a
ccuracy: 0.9548 - val_loss: 0.1599
Epoch 2/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - accuracy: 0.9599 - loss: 0.1352 - val_a
ccuracy: 0.9673 - val_loss: 0.1138
Epoch 3/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9738 - loss: 0.0903 - val_a
ccuracy: 0.9722 - val_loss: 0.0954
Epoch 4/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9804 - loss: 0.0634 - val_a
ccuracy: 0.9732 - val_loss: 0.0912
Epoch 5/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9863 - loss: 0.0476 - val_a
ccuracy: 0.9696 - val_loss: 0.0989
```

In [6]:

```
# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9709 - loss: 0.0977
Test Accuracy: 97.46%
```