In [1]:
```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

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

In [3]:
```
# Preprocess data (normalize pixel values to range [0, 1])
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
```

In [4]:
```
# Reshape data for input layer (28x28 pixels)
x_train = x_train.reshape(x_train.shape[0], 28 * 28)
x_test = x_test.reshape(x_test.shape[0], 28 * 28)
```

In [5]:
```
# Convert class labels to one-hot encoded vectors
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [6]:
```
# Define the Pure ANN model with less than 10,000 trainable parameters
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(28 * 28,)))  # First layer with 512 neurons
model.add(Dense(128, activation='relu'))  # Second layer with 128 neurons
model.add(Dense(10, activation='softmax'))  # Output layer with 10 neurons (one for each digit class)
```

In [7]:
```
# Compile the model
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy']
```

In [8]:
```
# Train the model
model.fit(x_train, y_train, epochs=10, batch_size=128, validation_data=(x_test, y_test))
```

```
Epoch 1/10
469/469 [==============================] - 4s 6ms/step - loss: 0.2393 - accuracy: 0.9304 - val_loss: 0.
1190 - val_accuracy: 0.9643
Epoch 2/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0895 - accuracy: 0.9726 - val_loss: 0.
0805 - val_accuracy: 0.9744
Epoch 3/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0550 - accuracy: 0.9829 - val_loss: 0.
0694 - val_accuracy: 0.9786
Epoch 4/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0391 - accuracy: 0.9880 - val_loss: 0.
0666 - val_accuracy: 0.9794
Epoch 5/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0278 - accuracy: 0.9909 - val_loss: 0.
0712 - val_accuracy: 0.9795
Epoch 6/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0218 - accuracy: 0.9935 - val_loss: 0.
0672 - val_accuracy: 0.9808
Epoch 7/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0191 - accuracy: 0.9936 - val_loss: 0.
0716 - val_accuracy: 0.9813
Epoch 8/10
469/469 [==============================] - 3s 7ms/step - loss: 0.0140 - accuracy: 0.9956 - val_loss: 0.
0717 - val_accuracy: 0.9811
Epoch 9/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0146 - accuracy: 0.9953 - val_loss: 0.
1071 - val_accuracy: 0.9749
Epoch 10/10
469/469 [==============================] - 3s 6ms/step - loss: 0.0128 - accuracy: 0.9958 - val_loss: 0.
0878 - val_accuracy: 0.9792
```

Out[8]:
```
<keras.src.callbacks.History at 0x1b63652f700>
```

In [9]:
```
# Evaluate the model on test data
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0878 - accuracy: 0.9792
Test accuracy: 0.979200005531311
```

In [10]:
```
# Count the total number of trainable parameters
total_params = model.count_params()
print('Total trainable parameters:', total_params)
```

```
Total trainable parameters: 468874
```