```
# Implementing feedforward neural networks with Keras and TensorFlow
# import the necessary packages
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,Dropout
import matplotlib.pyplot as plt


# grab the MNIST dataset (if this is your first time using this
# dataset then the 11MB download may take a minute)

print("[INFO] accessing MNIST...")
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape((x_train.shape[0], 28, 28, 1)).astype('float32') /
255
x_test = x_test.reshape((x_test.shape[0], 28, 28, 1)).astype('float32') / 255
```

```
[INFO] accessing MNIST...
```

```
model = Sequential()

# Convolutional layers
model.add(Conv2D(28,kernel_size=(3, 3), input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())

# Fully connected layers
model.add(Dense(200,activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(10,activation="softmax"))
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/
base_conv.py:107: UserWarning: 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 model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━
┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━
┩
│ conv2d (Conv2D)                      │ (None, 26, 26, 28)          │
280 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
│ max_pooling2d (MaxPooling2D)         │ (None, 13, 13, 28)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────
┤
```

```
| flatten (Flatten)                    | (None, 4732)                |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────
──────────┤
| dense (Dense)                        | (None, 200)                 |
946,600 |
├──────────────────────────────────────┼─────────────────────────────┼──────
──────────┤
| dropout (Dropout)                    | (None, 200)                 |
0 |
├──────────────────────────────────────┼─────────────────────────────┼──────
──────────┤
| dense_1 (Dense)                      | (None, 10)                  |
2,010 |
└──────────────────────────────────────┴─────────────────────────────┴──────
──────────┘

 Total params: 948,890 (3.62 MB)

 Trainable params: 948,890 (3.62 MB)

 Non-trainable params: 0 (0.00 B)
```

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

# Step 5: Train the model
model.fit(x_train, y_train, epochs=2)
```

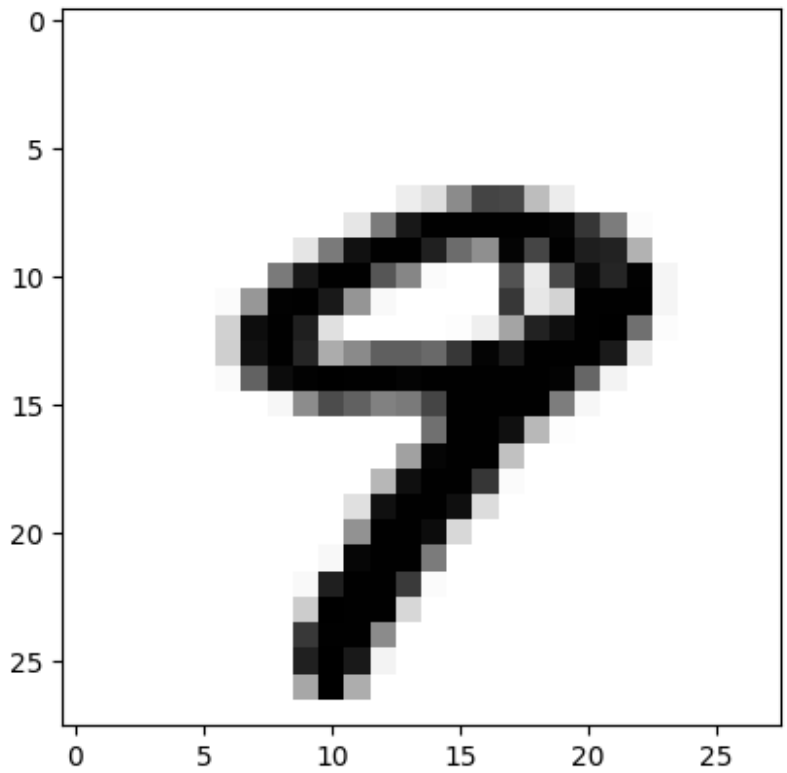
```
Epoch 1/2
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 14s 5ms/step - accuracy: 0.8935 - loss: 0.3502
Epoch 2/2
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 13s 3ms/step - accuracy: 0.9728 - loss: 0.0882

<keras.src.callbacks.history.History at 0x7a410db96fb0>
```

```
# Step 6: Evaluate the network
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_accuracy*100:.2f}%')
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9757 - loss: 0.0757
Test accuracy: 98.07%
```

```
image = x_test[9]
plt.imshow(image,cmap='Greys')
plt.show()
```




```
image=image.reshape(1,28,28,1)
prediction = model.predict(image)
print(np.argmax(prediction))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 256ms/step
9
```