```
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten

# Configuration
img_width, img_height = 28, 28
input_shape = (img_width, img_height, 1)
batch_size = 1000
no_epochs = 25
no_classes = 10
validation_split = 0.2
verbosity = 1

# Load data
def load_data():
  return tensorflow.keras.datasets.mnist.load_data(path="mnist.npz")

# Model creation
def create_model():
  model = Sequential()
  model.add(Conv2D(4, kernel_size=(3, 3), activation='relu',
input_shape=input_shape))
  model.add(Conv2D(8, kernel_size=(3, 3), activation='relu'))
  model.add(Conv2D(12, kernel_size=(3, 3), activation='relu'))
  model.add(Flatten())
  model.add(Dense(256, activation='relu'))
  model.add(Dense(no_classes, activation='softmax'))
  return model

# Model compilation
def compile_model(model):
  model.compile(loss=tensorflow.keras.losses.sparse_categorical_crossentropy,
          optimizer=tensorflow.keras.optimizers.Adam(),
          metrics=['accuracy'])
  return model

# Model training
def train_model(model, X_train, y_train):
  model.fit(X_train, y_train,
         batch_size=batch_size,
         epochs=no_epochs,
         verbose=verbosity,
         shuffle=True,
         validation_split=validation_split)
  return model

# Model testing
def test_model(model, X_test, y_test):
  score = model.evaluate(X_test, y_test, verbose=0)
  print(f'Test loss: {score[0]} / Test accuracy: {score[1]}')
  return model

# Load data
(X_train, y_train), (X_test, y_test) = load_data()
```

```
# Normalize data
(X_train, X_test) = (X_train / 255.0, X_test / 255.0)

# Reshape data
(X_train, X_test) = (
  X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1),
  X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1),
)

# Create and train the model
model = create_model()
model = compile_model(model)
model = train_model(model, X_train, y_train)
model = test_model(model, X_test, y_test)
```