Title of the Assignment: Multiclass classification using Deep Neural Networks: Example: Use the OCR letter recognition dataset

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam

# Load the dataset
file_path = "letter-recognition.csv"
data = pd.read_csv(file_path)

# Check the column names
print("Column Names:", data.columns)

# Preprocess the data

y = data.iloc[:, 0]  # Target
X = data.drop(columns=[y.name])  # Features

# Encode target labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
test_size=0.2, random_state=42)

# Build the model
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(len(label_encoder.classes_), activation='softmax')  # Output layer
with number of classes
])

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

# Train the model
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_accuracy)
```

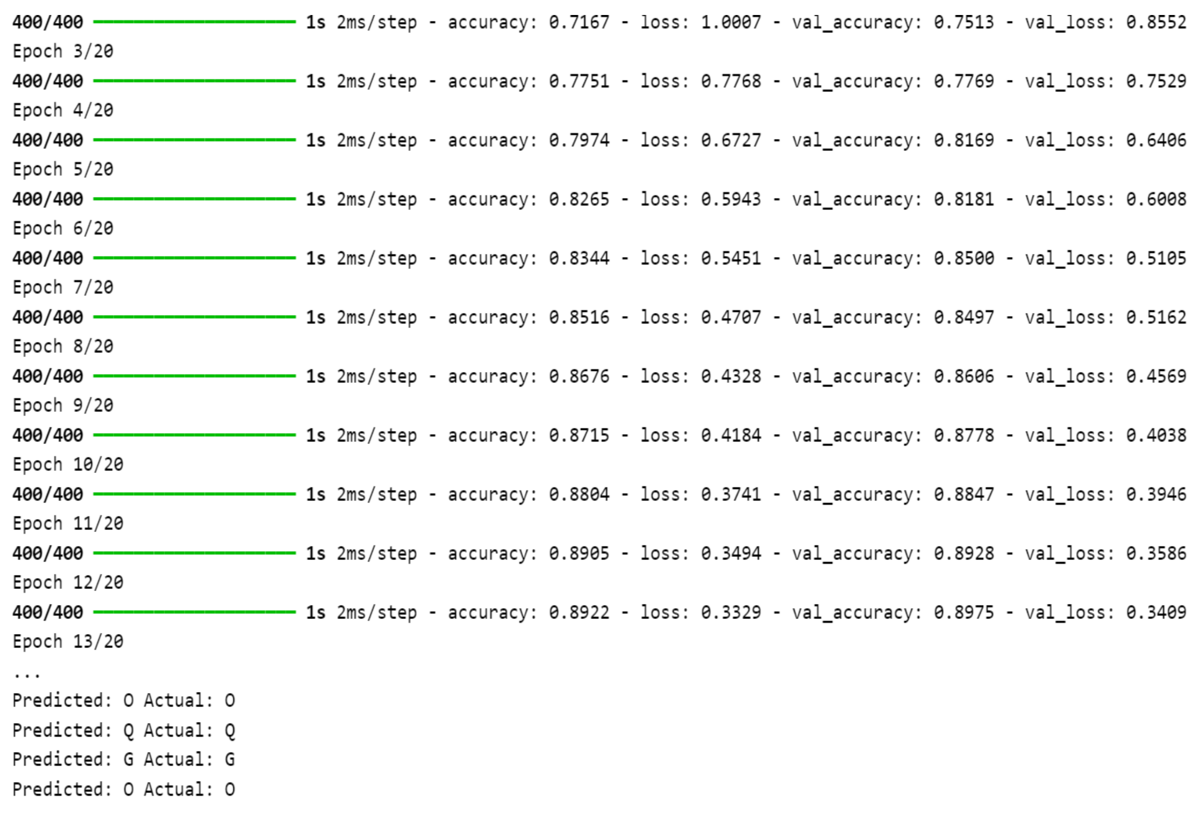
```python
# Make predictions
predictions = model.predict(X_test)

# Decode the predicted labels
predicted_labels = label_encoder.inverse_transform(np.argmax(predictions,
axis=1))

# Decode the actual labels
actual_labels = label_encoder.inverse_transform(y_test)

# Print some predicted and actual results
print("Some Predicted and Actual Results:")
for i in range(10):
    print("Predicted:", predicted_labels[i], "Actual:", actual_labels[i])
```

OUTPUT:



```
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7167 - loss: 1.0007 - val_accuracy: 0.7513 - val_loss: 0.8552
Epoch 3/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7751 - loss: 0.7768 - val_accuracy: 0.7769 - val_loss: 0.7529
Epoch 4/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7974 - loss: 0.6727 - val_accuracy: 0.8169 - val_loss: 0.6406
Epoch 5/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8265 - loss: 0.5943 - val_accuracy: 0.8181 - val_loss: 0.6008
Epoch 6/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8344 - loss: 0.5451 - val_accuracy: 0.8500 - val_loss: 0.5105
Epoch 7/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8516 - loss: 0.4707 - val_accuracy: 0.8497 - val_loss: 0.5162
Epoch 8/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8676 - loss: 0.4328 - val_accuracy: 0.8606 - val_loss: 0.4569
Epoch 9/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8715 - loss: 0.4184 - val_accuracy: 0.8778 - val_loss: 0.4038
Epoch 10/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8804 - loss: 0.3741 - val_accuracy: 0.8847 - val_loss: 0.3946
Epoch 11/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8905 - loss: 0.3494 - val_accuracy: 0.8928 - val_loss: 0.3586
Epoch 12/20
400/400 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8922 - loss: 0.3329 - val_accuracy: 0.8975 - val_loss: 0.3409
Epoch 13/20
...
Predicted: O Actual: O
Predicted: Q Actual: Q
Predicted: G Actual: G
Predicted: O Actual: O
```