import tensorflow as tf

from tensorflow.keras.models import Sequential #type:ignore

from tensorflow.keras.layers import Dense, Dropout #type:ignore

from tensorflow.keras.utils import to\_categorical #type:ignore

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report

import pandas as pd

# Load the OCR dataset

# Replace 'letter-recognition.data' with the actual path to your dataset

data = pd.read\_csv("C:\\Users\\bhumi\\OneDrive\\Desktop\\letter-recognition.csv", header=None)

# The dataset's first column contains labels (letters A-Z), encoded as characters.

# Convert the letter labels to numerical values.

data[0] = data[0].apply(lambda x: ord(x) - ord('A'))

# Split features and labels

X = data.iloc[:, 1:].values # Features (16 numerical attributes)

y = data.iloc[:, 0].values # Labels (letters A-Z as integers)

# One-hot encode the labels

y = to\_categorical(y, num\_classes=26)

# Split into training and testing datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Normalize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Build the deep neural network model

model = Sequential([

Dense(128, activation='relu', input\_shape=(X\_train.shape[1],)),

Dropout(0.3),

Dense(64, activation='relu'),

Dropout(0.3),

Dense(26, activation='softmax') # Output layer with 26 classes

])

# Compile the model

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

history = model.fit(X\_train, y\_train,

epochs=30,

batch\_size=32,

validation\_split=0.2,

verbose=1)

# Evaluate the model on the test data

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test, verbose=0)

print(f"Test Accuracy: {test\_accuracy \* 100:.2f}%")

# Generate predictions

predictions = model.predict(X\_test)

y\_pred = predictions.argmax(axis=1)

y\_true = y\_test.argmax(axis=1)

# Classification report

print("Classification Report:\n")

print(classification\_report(y\_true, y\_pred, target\_names=[chr(i) for i in range(ord('A'), ord('Z') + 1)]))

# Save the model

model.save("ocr\_multiclass\_model.h5")

# Visualize training history

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))

# Plot accuracy

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

# Plot loss

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.show()

Multiclass classification with deep neural networks is a powerful technique for OCR

tasks. By leveraging architectures tailored to the dataset, preprocessing appropriately, and

optimizing training, high accuracy can be achieved. The OCR Letter Recognition dataset

offers an excellent starting point for exploring multiclass classification.