import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

from sklearn.metrics import classification\_report, confusion\_matrix

import numpy as np

import matplotlib.pyplot as plt

import itertools

import os

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.fashion\_mnist.load\_data()

x\_train = x\_train.astype("float32") / 255.0

x\_test = x\_test.astype("float32") / 255.0

x\_train = np.expand\_dims(x\_train, -1)

x\_test = np.expand\_dims(x\_test, -1)

y\_train\_cat = to\_categorical(y\_train, 10)

y\_test\_cat = to\_categorical(y\_test, 10)

model = models.Sequential([

layers.Input(shape=(28,28,1)),

layers.Conv2D(32,3,activation="relu"),

layers.BatchNormalization(),

layers.Conv2D(32,3,activation="relu"),

layers.MaxPooling2D(),

layers.Dropout(0.25),

layers.Conv2D(64,3,activation="relu"),

layers.BatchNormalization(),

layers.Conv2D(64,3,activation="relu"),

layers.MaxPooling2D(),

layers.Dropout(0.25),

layers.Flatten(),

layers.Dense(128,activation="relu"),

layers.BatchNormalization(),

layers.Dropout(0.5),

layers.Dense(10,activation="softmax")

])

model.compile(optimizer="adam", loss="categorical\_crossentropy", metrics=["accuracy"])

os.makedirs("model\_checkpoints", exist\_ok=True)

checkpoint = ModelCheckpoint("model\_checkpoints/best\_model.h5", monitor="val\_accuracy", save\_best\_only=True, verbose=1)

early = EarlyStopping(monitor="val\_accuracy", patience=7, restore\_best\_weights=True, verbose=1)

history = model.fit(x\_train, y\_train\_cat, validation\_split=0.1, epochs=50, batch\_size=128, callbacks=[checkpoint, early])

test\_loss, test\_acc = model.evaluate(x\_test, y\_test\_cat, verbose=0)

print(f"Test loss: {test\_loss:.4f}")

print(f"Test accuracy: {test\_acc:.4f}")

y\_pred\_probs = model.predict(x\_test)

y\_pred = np.argmax(y\_pred\_probs, axis=1)

print(classification\_report(y\_test, y\_pred, digits=4))

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8,8))

plt.imshow(cm, interpolation="nearest")

plt.title("Confusion Matrix")

plt.colorbar()

tick\_marks = np.arange(10)

plt.xticks(tick\_marks, tick\_marks)

plt.yticks(tick\_marks, tick\_marks)

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], "d"), horizontalalignment="center", color="white" if cm[i, j] > thresh else "black")

plt.ylabel("True label")

plt.xlabel("Predicted label")

plt.tight\_layout()

plt.savefig("confusion\_matrix.png")

model.save("fashion\_mnist\_model.h5")

print("Model saved to fashion\_mnist\_model.h5")

print("Confusion matrix saved to confusion\_matrix.png")