# 

metin, yazı tipi, tipografi içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, kare, dikdörtgen, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import RMSprop

from sklearn.metrics import precision\_score, recall\_score, f1\_score, accuracy\_score

# Verilerin bulunduğu dizinler

train\_dir = '/Users/furkanerdogan/Projects/deep-learning/Fruits/fruits/train'

test\_dir = '/Users/furkanerdogan/Projects/deep-learning/Fruits/fruits/test'

# Görüntü boyutu

IMG\_SIZE = 100

NUM\_CLASSES = 10 # 10 sınıf olduğu varsayılıyor

# Klasör adlarına karşılık gelen etiketler

class\_labels = {class\_name: idx for idx, class\_name in enumerate(os.listdir(train\_dir))}

# Verileri yükleme ve yeniden boyutlandırma

def load\_images\_from\_folder(folder):

images = []

labels = []

for label\_name in os.listdir(folder): # Sınıflara göre alt klasörlerde dolaş

class\_folder = os.path.join(folder, label\_name)

label = class\_labels[label\_name] # Klasör adını etiket olarak al

for filename in os.listdir(class\_folder):

img\_path = os.path.join(class\_folder, filename)

img = cv2.imread(img\_path)

if img is not None:

img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE)) # 100x100 olarak yeniden boyutlandırma

images.append(img)

labels.append(label)

return np.array(images), np.array(labels)

# Eğitim ve test verilerini yükleme

X\_train, y\_train = load\_images\_from\_folder(train\_dir)

X\_test, y\_test = load\_images\_from\_folder(test\_dir)

# Verileri normalize etme

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

# Etiketleri one-hot encoding ile dönüştürme

y\_train = to\_categorical(y\_train, num\_classes=NUM\_CLASSES)

y\_test\_categorical = to\_categorical(y\_test, num\_classes=NUM\_CLASSES)

# Veri artırma

datagen = ImageDataGenerator(

rotation\_range=15,

width\_shift\_range=0.1,

height\_shift\_range=0.1,

shear\_range=0.1,

zoom\_range=0.1,

horizontal\_flip=True,

fill\_mode='nearest'

)

# CNN modelinin oluşturulması

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(IMG\_SIZE, IMG\_SIZE, 3)),

BatchNormalization(),

MaxPooling2D((2, 2)),

Dropout(0.3),

Conv2D(64, (3, 3), activation='relu'),

BatchNormalization(),

MaxPooling2D((2, 2)),

Dropout(0.3),

Conv2D(128, (3, 3), activation='relu'),

BatchNormalization(),

MaxPooling2D((2, 2)),

Dropout(0.4),

Conv2D(256, (3, 3), activation='relu'),

BatchNormalization(),

MaxPooling2D((2, 2)),

Dropout(0.4),

Flatten(),

Dense(512, activation='relu'),

BatchNormalization(),

Dropout(0.5),

Dense(NUM\_CLASSES, activation='softmax')

])

# Modelin derlenmesi

model.compile(optimizer=RMSprop(learning\_rate=0.0005), loss='categorical\_crossentropy', metrics=['accuracy'])

# Modelin eğitilmesi

history = model.fit(datagen.flow(X\_train, y\_train, batch\_size=32),

epochs=15, validation\_data=(X\_test, y\_test\_categorical))

# Modeli kaydetme

model.save('fruit\_classification\_model\_optimized\_2.h5')

# Eğitim ve doğrulama kayıp ve doğruluk grafikleri

plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')

plt.plot(history.history['val\_accuracy'], label='Doğrulama Doğruluğu')

plt.xlabel('Epoch')

plt.ylabel('Doğruluk')

plt.legend(loc='lower right')

plt.title('Model Doğruluğu')

plt.show()

plt.plot(history.history['loss'], label='Eğitim Kaybı')

plt.plot(history.history['val\_loss'], label='Doğrulama Kaybı')

plt.xlabel('Epoch')

plt.ylabel('Kayıp')

plt.legend(loc='upper right')

plt.title('Model Kaybı')

plt.show()

# Tahminlerin yapılması

y\_pred = model.predict(X\_test)

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

# Confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred\_classes)

print("Confusion Matrix:\n", conf\_matrix)

# Confusion matrix'i sayısal değerlerle görselleştirme

plt.figure(figsize=(10,7))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Tahmin Edilen Etiket')

plt.ylabel('Gerçek Etiket')

plt.show()

# Metriklerin hesaplanması

precision = precision\_score(y\_test, y\_pred\_classes, average='macro')

recall = recall\_score(y\_test, y\_pred\_classes, average='macro')

f1 = f1\_score(y\_test, y\_pred\_classes, average='macro')

accuracy = accuracy\_score(y\_test, y\_pred\_classes)

# Specificity hesaplama (Her sınıf için)

TN = conf\_matrix.sum() - (conf\_matrix.sum(axis=1) + conf\_matrix.sum(axis=0) - np.diag(conf\_matrix))

FP = conf\_matrix.sum(axis=0) - np.diag(conf\_matrix)

specificity = np.mean(TN / (TN + FP))

# Metriklerin yazdırılması

print(f'Accuracy: {accuracy:.4f}')

print(f'Precision: {precision:.4f}')

print(f'Recall: {recall:.4f}')

print(f'F1-Score: {f1:.4f}')

print(f'Specificity: {specificity:.4f}')

# İlk deneme’de 5 saat sonra kodun çıktısı bu şekildeydi, bilgisayar kaynaklı bir sorundan dolayı durdurmak zorunda kaldım.

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import precision\_score, recall\_score, f1\_score, accuracy\_score

from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# Verilerin bulunduğu dizinler

train\_dir = '/Users/furkanerdogan/Projects/deep-learning/Fruits/fruits/train'

test\_dir = '/Users/furkanerdogan/Projects/deep-learning/Fruits/fruits/test'

# Görüntü boyutu

IMG\_SIZE = 150 # Görüntü boyutunu artırdık

NUM\_CLASSES = 10

# Klasör adlarına karşılık gelen etiketler

class\_labels = {class\_name: idx for idx, class\_name in enumerate(os.listdir(train\_dir))}

# Verileri yükleme ve yeniden boyutlandırma

def load\_images\_from\_folder(folder):

images = []

labels = []

for label\_name in os.listdir(folder):

class\_folder = os.path.join(folder, label\_name)

label = class\_labels[label\_name]

for filename in os.listdir(class\_folder):

img\_path = os.path.join(class\_folder, filename)

img = cv2.imread(img\_path)

if img is not None:

img = cv2.resize(img, (IMG\_SIZE, IMG\_SIZE))

images.append(img)

labels.append(label)

return np.array(images), np.array(labels)

# Eğitim ve test verilerini yükleme

X\_train, y\_train = load\_images\_from\_folder(train\_dir)

X\_test, y\_test = load\_images\_from\_folder(test\_dir)

# Verileri normalize etme

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

# Veri artırma için ImageDataGenerator kullanımı (sadece eğitim verisi için)

datagen = ImageDataGenerator(

rotation\_range=20,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

horizontal\_flip=True,

vertical\_flip=False,

zoom\_range=0.2,

shear\_range=0.2,

fill\_mode='nearest'

)

# Generator'ı eğitim verisiyle fit etme

datagen.fit(X\_train)

# Etiketleri one-hot encoding ile dönüştürme

y\_train = to\_categorical(y\_train, num\_classes=NUM\_CLASSES)

y\_test\_categorical = to\_categorical(y\_test, num\_classes=NUM\_CLASSES)

# İyileştirilmiş CNN modeli

model = Sequential([

# İlk Konvolüsyon Bloğu

Conv2D(64, (3, 3), padding='same', activation='relu', input\_shape=(IMG\_SIZE, IMG\_SIZE, 3)),

BatchNormalization(),

Conv2D(64, (3, 3), padding='same', activation='relu'),

BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

Dropout(0.25),

# İkinci Konvolüsyon Bloğu

Conv2D(128, (3, 3), padding='same', activation='relu'),

BatchNormalization(),

Conv2D(128, (3, 3), padding='same', activation='relu'),

BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

Dropout(0.25),

# Üçüncü Konvolüsyon Bloğu

Conv2D(256, (3, 3), padding='same', activation='relu'),

BatchNormalization(),

Conv2D(256, (3, 3), padding='same', activation='relu'),

BatchNormalization(),

MaxPooling2D(pool\_size=(2, 2)),

Dropout(0.25),

# Fully Connected Katmanlar

Flatten(),

Dense(512, activation='relu'),

BatchNormalization(),

Dropout(0.5),

Dense(256, activation='relu'),

BatchNormalization(),

Dropout(0.5),

Dense(NUM\_CLASSES, activation='softmax')

])

# Modelin derlenmesi

model.compile(

optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy']

)

# Callbacks

early\_stopping = EarlyStopping(

monitor='val\_loss',

patience=10,

restore\_best\_weights=True

)

reduce\_lr = ReduceLROnPlateau(

monitor='val\_loss',

factor=0.2,

patience=5,

min\_lr=1e-6

)

# Modelin eğitilmesi (veri artırma ile)

history = model.fit(

datagen.flow(X\_train, y\_train, batch\_size=32),

epochs=50,

validation\_data=(X\_test, y\_test\_categorical),

callbacks=[early\_stopping, reduce\_lr]

)

# Modeli kaydetme

model.save('improved\_fruit\_classification\_model.h5')

# Eğitim ve doğrulama kayıp ve doğruluk grafikleri

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')

plt.plot(history.history['val\_accuracy'], label='Doğrulama Doğruluğu')

plt.xlabel('Epoch')

plt.ylabel('Doğruluk')

plt.legend()

plt.title('Model Doğruluğu')

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Eğitim Kaybı')

plt.plot(history.history['val\_loss'], label='Doğrulama Kaybı')

plt.xlabel('Epoch')

plt.ylabel('Kayıp')

plt.legend()

plt.title('Model Kaybı')

plt.tight\_layout()

plt.show()

# Tahminlerin yapılması

y\_pred = model.predict(X\_test)

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

# Confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred\_classes)

print("Confusion Matrix:\n", conf\_matrix)

# Confusion matrix'i görselleştirme

plt.figure(figsize=(10,7))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)

plt.title('Confusion Matrix')

plt.xlabel('Tahmin Edilen Etiket')

plt.ylabel('Gerçek Etiket')

plt.show()

# Metriklerin hesaplanması

precision = precision\_score(y\_test, y\_pred\_classes, average='macro')

recall = recall\_score(y\_test, y\_pred\_classes, average='macro')

f1 = f1\_score(y\_test, y\_pred\_classes, average='macro')

accuracy = accuracy\_score(y\_test, y\_pred\_classes)

# Specificity hesaplama

TN = conf\_matrix.sum() - (conf\_matrix.sum(axis=1) + conf\_matrix.sum(axis=0) - np.diag(conf\_matrix))

FP = conf\_matrix.sum(axis=0) - np.diag(conf\_matrix)

specificity = np.mean(TN / (TN + FP))

# Metriklerin yazdırılması

print(f'Accuracy: {accuracy:.4f}')

print(f'Precision: {precision:.4f}')

print(f'Recall: {recall:.4f}')

print(f'F1-Score: {f1:.4f}')

print(f'Specificity: {specificity:.4f}')