import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

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

from tensorflow.keras.applications import DenseNet121

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.optimizers import Adam

# Verilerin bulunduğu dizinler

train\_dir = '/content/drive/MyDrive/egzema/train'

test\_dir = '/content/drive/MyDrive/egzema/test'

# Görüntü boyutu ve sınıf sayısı

IMG\_SIZE = 224

NUM\_CLASSES = 2

# Klasör adlarına karşılık gelen etiketler

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

# Verileri yükleme fonksiyonu

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)

# Normalize et

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

# One-hot encoding

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

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

# Modeli oluştur

print("\n--- DenseNet121 Modeli Eğitiliyor ---\n")

base\_model = DenseNet121(weights='imagenet', include\_top=False, input\_shape=(IMG\_SIZE, IMG\_SIZE, 3))

base\_model.trainable = False

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(128, activation='relu')(x)

x = Dropout(0.5)(x)

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

model = Model(inputs=base\_model.input, outputs=predictions)

model.compile(optimizer=Adam(learning\_rate=0.0001), loss='categorical\_crossentropy', metrics=['accuracy'])

# Eğitimi başlat

history = model.fit(

X\_train, y\_train,

epochs=50,

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

batch\_size=32,

verbose=1

)

# Tahmin ve metrikler

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

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

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

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)

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))

# Sonuçları yazdır

print("\nDenseNet121 Model Sonuçları:")

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}")

# Grafikler

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('DenseNet121 Modeli Doğruluk Grafiği')

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('DenseNet121 Modeli Kayıp Grafiği')

plt.show()

--- DenseNet121 Modeli Eğitiliyor ---

Downloading data from <https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5>

**29084464/29084464** ━━━━━━━━━━━━━━━━━━━━ **2s** 0us/step

Epoch 1/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **80s** 626ms/step - accuracy: 0.7813 - loss: 0.4571 - val\_accuracy: 0.9699 - val\_loss: 0.1090

Epoch 2/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **31s** 92ms/step - accuracy: 0.9426 - loss: 0.1676 - val\_accuracy: 0.9831 - val\_loss: 0.0575

Epoch 3/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 96ms/step - accuracy: 0.9732 - loss: 0.0889 - val\_accuracy: 0.9925 - val\_loss: 0.0400

Epoch 4/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 95ms/step - accuracy: 0.9880 - loss: 0.0601 - val\_accuracy: 0.9981 - val\_loss: 0.0282

Epoch 5/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 92ms/step - accuracy: 0.9831 - loss: 0.0491 - val\_accuracy: 0.9981 - val\_loss: 0.0222

Epoch 6/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 93ms/step - accuracy: 0.9891 - loss: 0.0379 - val\_accuracy: 0.9981 - val\_loss: 0.0178

Epoch 7/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 95ms/step - accuracy: 0.9884 - loss: 0.0365 - val\_accuracy: 0.9981 - val\_loss: 0.0153

Epoch 8/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 93ms/step - accuracy: 0.9947 - loss: 0.0236 - val\_accuracy: 0.9981 - val\_loss: 0.0131

Epoch 9/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 94ms/step - accuracy: 0.9943 - loss: 0.0233 - val\_accuracy: 0.9981 - val\_loss: 0.0116

Epoch 10/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **8s** 116ms/step - accuracy: 0.9917 - loss: 0.0207 - val\_accuracy: 0.9981 - val\_loss: 0.0102

Epoch 11/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 95ms/step - accuracy: 0.9953 - loss: 0.0204 - val\_accuracy: 0.9981 - val\_loss: 0.0092

Epoch 12/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 95ms/step - accuracy: 0.9988 - loss: 0.0122 - val\_accuracy: 0.9962 - val\_loss: 0.0093

Epoch 13/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 97ms/step - accuracy: 0.9984 - loss: 0.0120 - val\_accuracy: 0.9981 - val\_loss: 0.0078

Epoch 14/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **7s** 98ms/step - accuracy: 0.9963 - loss: 0.0117 - val\_accuracy: 1.0000 - val\_loss: 0.0075

Epoch 15/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 96ms/step - accuracy: 0.9981 - loss: 0.0107 - val\_accuracy: 0.9981 - val\_loss: 0.0066

Epoch 16/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 94ms/step - accuracy: 0.9998 - loss: 0.0065 - val\_accuracy: 0.9981 - val\_loss: 0.0062

Epoch 17/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 94ms/step - accuracy: 0.9973 - loss: 0.0127 - val\_accuracy: 1.0000 - val\_loss: 0.0060

Epoch 18/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **8s** 115ms/step - accuracy: 0.9992 - loss: 0.0101 - val\_accuracy: 0.9981 - val\_loss: 0.0057

Epoch 19/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 94ms/step - accuracy: 0.9979 - loss: 0.0078 - val\_accuracy: 1.0000 - val\_loss: 0.0051

Epoch 20/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 94ms/step - accuracy: 0.9999 - loss: 0.0068 - val\_accuracy: 1.0000 - val\_loss: 0.0048

Epoch 21/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 96ms/step - accuracy: 0.9994 - loss: 0.0065 - val\_accuracy: 1.0000 - val\_loss: 0.0046

Epoch 22/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 95ms/step - accuracy: 0.9988 - loss: 0.0064 - val\_accuracy: 1.0000 - val\_loss: 0.0046

Epoch 23/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 97ms/step - accuracy: 0.9999 - loss: 0.0048 - val\_accuracy: 1.0000 - val\_loss: 0.0042

Epoch 24/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 98ms/step - accuracy: 0.9991 - loss: 0.0047 - val\_accuracy: 0.9981 - val\_loss: 0.0046

Epoch 25/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **11s** 114ms/step - accuracy: 1.0000 - loss: 0.0064 - val\_accuracy: 1.0000 - val\_loss: 0.0041

Epoch 26/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **9s** 95ms/step - accuracy: 0.9999 - loss: 0.0044 - val\_accuracy: 1.0000 - val\_loss: 0.0040

Epoch 27/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **7s** 99ms/step - accuracy: 1.0000 - loss: 0.0035 - val\_accuracy: 0.9981 - val\_loss: 0.0046

Epoch 28/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **11s** 116ms/step - accuracy: 1.0000 - loss: 0.0025 - val\_accuracy: 0.9981 - val\_loss: 0.0039

Epoch 29/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **9s** 95ms/step - accuracy: 1.0000 - loss: 0.0025 - val\_accuracy: 0.9981 - val\_loss: 0.0039

Epoch 30/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 97ms/step - accuracy: 0.9998 - loss: 0.0028 - val\_accuracy: 0.9981 - val\_loss: 0.0044

Epoch 31/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 94ms/step - accuracy: 0.9994 - loss: 0.0031 - val\_accuracy: 1.0000 - val\_loss: 0.0031

Epoch 32/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 96ms/step - accuracy: 1.0000 - loss: 0.0031 - val\_accuracy: 1.0000 - val\_loss: 0.0032

Epoch 33/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 94ms/step - accuracy: 0.9992 - loss: 0.0029 - val\_accuracy: 0.9981 - val\_loss: 0.0034

Epoch 34/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 96ms/step - accuracy: 0.9997 - loss: 0.0031 - val\_accuracy: 0.9981 - val\_loss: 0.0032

Epoch 35/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 94ms/step - accuracy: 0.9999 - loss: 0.0017 - val\_accuracy: 0.9981 - val\_loss: 0.0031

Epoch 36/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 96ms/step - accuracy: 1.0000 - loss: 0.0021 - val\_accuracy: 0.9981 - val\_loss: 0.0032

Epoch 37/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 94ms/step - accuracy: 1.0000 - loss: 0.0017 - val\_accuracy: 0.9981 - val\_loss: 0.0030

Epoch 38/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 96ms/step - accuracy: 1.0000 - loss: 0.0015 - val\_accuracy: 1.0000 - val\_loss: 0.0026

Epoch 39/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 97ms/step - accuracy: 1.0000 - loss: 0.0015 - val\_accuracy: 0.9981 - val\_loss: 0.0028

Epoch 40/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 95ms/step - accuracy: 0.9993 - loss: 0.0022 - val\_accuracy: 1.0000 - val\_loss: 0.0026

Epoch 41/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 94ms/step - accuracy: 0.9999 - loss: 0.0020 - val\_accuracy: 1.0000 - val\_loss: 0.0021

Epoch 42/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **7s** 97ms/step - accuracy: 1.0000 - loss: 0.0014 - val\_accuracy: 0.9981 - val\_loss: 0.0030

Epoch 43/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 97ms/step - accuracy: 1.0000 - loss: 0.0012 - val\_accuracy: 0.9981 - val\_loss: 0.0030

Epoch 44/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 93ms/step - accuracy: 1.0000 - loss: 0.0013 - val\_accuracy: 0.9981 - val\_loss: 0.0027

Epoch 45/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 95ms/step - accuracy: 0.9997 - loss: 0.0014 - val\_accuracy: 1.0000 - val\_loss: 0.0023

Epoch 46/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 97ms/step - accuracy: 0.9997 - loss: 0.0011 - val\_accuracy: 1.0000 - val\_loss: 0.0023

Epoch 47/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 94ms/step - accuracy: 1.0000 - loss: 0.0013 - val\_accuracy: 0.9981 - val\_loss: 0.0025

Epoch 48/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 97ms/step - accuracy: 1.0000 - loss: 9.7120e-04 - val\_accuracy: 0.9981 - val\_loss: 0.0027

Epoch 49/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **6s** 94ms/step - accuracy: 0.9999 - loss: 8.8617e-04 - val\_accuracy: 0.9981 - val\_loss: 0.0024

Epoch 50/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **10s** 94ms/step - accuracy: 1.0000 - loss: 0.0011 - val\_accuracy: 1.0000 - val\_loss: 0.0021

**17/17** ━━━━━━━━━━━━━━━━━━━━ **24s** 828ms/step

DenseNet121 Model Sonuçları:

Accuracy : 0.9981

Precision : 0.9975

Recall : 0.9985

F1 Score : 0.9980

Specificity: 0.9985



