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 ResNet101

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

# Dizinler

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

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

IMG\_SIZE = 224

NUM\_CLASSES = 2

# Etiketleme

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

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)

# Verileri yükle

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

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

# Normalizasyon ve One-hot encoding

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

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

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

# ResNet101 modeli

print("\n--- ResNet101 Modeli Eğitiliyor ---\n")

base\_model = ResNet101(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'])

# Eğitim

history = model.fit(

X\_train, y\_train,

epochs=50,

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

batch\_size=32,

verbose=1

)

# Tahminler 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))

# Sonuçlar

print("\nResNet101 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('ResNet101 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('ResNet101 Modeli Kayıp Grafiği')

plt.show()

--- ResNet101 Modeli Eğitiliyor ---

Downloading data from <https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet101_weights_tf_dim_ordering_tf_kernels_notop.h5>

**171446536/171446536** ━━━━━━━━━━━━━━━━━━━━ **8s** 0us/step

Epoch 1/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **54s** 483ms/step - accuracy: 0.5923 - loss: 0.7486 - val\_accuracy: 0.6316 - val\_loss: 0.6395

Epoch 2/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **53s** 189ms/step - accuracy: 0.6198 - loss: 0.6642 - val\_accuracy: 0.6316 - val\_loss: 0.6231

Epoch 3/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 190ms/step - accuracy: 0.6212 - loss: 0.6407 - val\_accuracy: 0.6316 - val\_loss: 0.6178

Epoch 4/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.6529 - loss: 0.6147 - val\_accuracy: 0.6316 - val\_loss: 0.6070

Epoch 5/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 187ms/step - accuracy: 0.6428 - loss: 0.6150 - val\_accuracy: 0.6316 - val\_loss: 0.5928

Epoch 6/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **13s** 189ms/step - accuracy: 0.6517 - loss: 0.5952 - val\_accuracy: 0.6316 - val\_loss: 0.5858

Epoch 7/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 186ms/step - accuracy: 0.6780 - loss: 0.5840 - val\_accuracy: 0.6316 - val\_loss: 0.5751

Epoch 8/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 187ms/step - accuracy: 0.6520 - loss: 0.5848 - val\_accuracy: 0.6316 - val\_loss: 0.5679

Epoch 9/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 189ms/step - accuracy: 0.6603 - loss: 0.5817 - val\_accuracy: 0.6316 - val\_loss: 0.5610

Epoch 10/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.6568 - loss: 0.5679 - val\_accuracy: 0.6316 - val\_loss: 0.5550

Epoch 11/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 188ms/step - accuracy: 0.6805 - loss: 0.5590 - val\_accuracy: 0.8421 - val\_loss: 0.5497

Epoch 12/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 187ms/step - accuracy: 0.6734 - loss: 0.5623 - val\_accuracy: 0.6372 - val\_loss: 0.5405

Epoch 13/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 188ms/step - accuracy: 0.7016 - loss: 0.5458 - val\_accuracy: 0.6316 - val\_loss: 0.5363

Epoch 14/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 189ms/step - accuracy: 0.7144 - loss: 0.5403 - val\_accuracy: 0.6316 - val\_loss: 0.5322

Epoch 15/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **13s** 190ms/step - accuracy: 0.7236 - loss: 0.5258 - val\_accuracy: 0.6617 - val\_loss: 0.5229

Epoch 16/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.7364 - loss: 0.5322 - val\_accuracy: 0.8365 - val\_loss: 0.5169

Epoch 17/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 186ms/step - accuracy: 0.7427 - loss: 0.5235 - val\_accuracy: 0.7444 - val\_loss: 0.5121

Epoch 18/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 188ms/step - accuracy: 0.7334 - loss: 0.5287 - val\_accuracy: 0.8496 - val\_loss: 0.5061

Epoch 19/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.7418 - loss: 0.5231 - val\_accuracy: 0.8459 - val\_loss: 0.5029

Epoch 20/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **13s** 190ms/step - accuracy: 0.7733 - loss: 0.5113 - val\_accuracy: 0.8571 - val\_loss: 0.4967

Epoch 21/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 186ms/step - accuracy: 0.7997 - loss: 0.5038 - val\_accuracy: 0.8590 - val\_loss: 0.4926

Epoch 22/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 188ms/step - accuracy: 0.7860 - loss: 0.5009 - val\_accuracy: 0.8440 - val\_loss: 0.4881

Epoch 23/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 189ms/step - accuracy: 0.7740 - loss: 0.5153 - val\_accuracy: 0.8590 - val\_loss: 0.4837

Epoch 24/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 189ms/step - accuracy: 0.7992 - loss: 0.4931 - val\_accuracy: 0.8515 - val\_loss: 0.4790

Epoch 25/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.7910 - loss: 0.5011 - val\_accuracy: 0.8421 - val\_loss: 0.4751

Epoch 26/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **23s** 226ms/step - accuracy: 0.8067 - loss: 0.4772 - val\_accuracy: 0.8459 - val\_loss: 0.4711

Epoch 27/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **18s** 189ms/step - accuracy: 0.8200 - loss: 0.4710 - val\_accuracy: 0.8496 - val\_loss: 0.4672

Epoch 28/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 187ms/step - accuracy: 0.7843 - loss: 0.4801 - val\_accuracy: 0.8477 - val\_loss: 0.4642

Epoch 29/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 187ms/step - accuracy: 0.8278 - loss: 0.4700 - val\_accuracy: 0.8515 - val\_loss: 0.4604

Epoch 30/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.8135 - loss: 0.4705 - val\_accuracy: 0.8515 - val\_loss: 0.4566

Epoch 31/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 188ms/step - accuracy: 0.8158 - loss: 0.4638 - val\_accuracy: 0.8477 - val\_loss: 0.4534

Epoch 32/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 189ms/step - accuracy: 0.8147 - loss: 0.4563 - val\_accuracy: 0.8496 - val\_loss: 0.4528

Epoch 33/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 189ms/step - accuracy: 0.8185 - loss: 0.4677 - val\_accuracy: 0.8534 - val\_loss: 0.4472

Epoch 34/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **13s** 190ms/step - accuracy: 0.8045 - loss: 0.4727 - val\_accuracy: 0.8327 - val\_loss: 0.4465

Epoch 35/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 186ms/step - accuracy: 0.8324 - loss: 0.4612 - val\_accuracy: 0.8571 - val\_loss: 0.4419

Epoch 36/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 188ms/step - accuracy: 0.8290 - loss: 0.4496 - val\_accuracy: 0.8534 - val\_loss: 0.4393

Epoch 37/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 189ms/step - accuracy: 0.8174 - loss: 0.4490 - val\_accuracy: 0.8515 - val\_loss: 0.4493

Epoch 38/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **13s** 189ms/step - accuracy: 0.8318 - loss: 0.4493 - val\_accuracy: 0.7857 - val\_loss: 0.4445

Epoch 39/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 186ms/step - accuracy: 0.7998 - loss: 0.4520 - val\_accuracy: 0.8534 - val\_loss: 0.4355

Epoch 40/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 186ms/step - accuracy: 0.8290 - loss: 0.4613 - val\_accuracy: 0.8365 - val\_loss: 0.4306

Epoch 41/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 190ms/step - accuracy: 0.8048 - loss: 0.4517 - val\_accuracy: 0.8684 - val\_loss: 0.4372

Epoch 42/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.8359 - loss: 0.4450 - val\_accuracy: 0.8609 - val\_loss: 0.4285

Epoch 43/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 189ms/step - accuracy: 0.8411 - loss: 0.4361 - val\_accuracy: 0.8571 - val\_loss: 0.4250

Epoch 44/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **23s** 226ms/step - accuracy: 0.8288 - loss: 0.4363 - val\_accuracy: 0.8553 - val\_loss: 0.4258

Epoch 45/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **18s** 189ms/step - accuracy: 0.8251 - loss: 0.4391 - val\_accuracy: 0.8459 - val\_loss: 0.4209

Epoch 46/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.8524 - loss: 0.4226 - val\_accuracy: 0.8158 - val\_loss: 0.4244

Epoch 47/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.8272 - loss: 0.4379 - val\_accuracy: 0.8402 - val\_loss: 0.4172

Epoch 48/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 188ms/step - accuracy: 0.8435 - loss: 0.4222 - val\_accuracy: 0.8571 - val\_loss: 0.4165

Epoch 49/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **21s** 189ms/step - accuracy: 0.8231 - loss: 0.4356 - val\_accuracy: 0.8571 - val\_loss: 0.4175

Epoch 50/50

**67/67** ━━━━━━━━━━━━━━━━━━━━ **20s** 190ms/step - accuracy: 0.8454 - loss: 0.4210 - val\_accuracy: 0.8102 - val\_loss: 0.4195

**17/17** ━━━━━━━━━━━━━━━━━━━━ **17s** 531ms/step

ResNet101 Model Sonuçları:

Accuracy : 0.8102

Precision : 0.7984

Recall : 0.8157

F1 Score : 0.8028

Specificity: 0.8157



