

Derin Öğrenme Tabanlı Egzama Teşhisinde Xception ve DenseNet121 Modellerinin Karşılaştırılması

ÖMER FARUK YELMAN

AZRA ŞAHİN

AYŞE CEREN TUNCER

| YAZILIM MÜHENDİSLİĞİ |

**1. Özet**

Bu çalışma, egzama hastalığının görüntü işleme teknikleriyle otomatik olarak teşhis edilmesini amaçlamaktadır. Deri hastalıklarının erken teşhisi, hastalığın ilerlemesini önlemek ve etkili bir tedavi süreci başlatmak açısından büyük önem taşımaktadır. Bu amaçla, egzama görsellerinden oluşan bir veri seti kullanılarak Xception ve DenseNet121 derin öğrenme modelleri eğitilmiştir. Modeller, eğitim ve test aşamalarında veri artırma (data augmentation), erken durdurma (early stopping) ve transfer öğrenme teknikleriyle optimize edilmiştir. Eğitim sonucunda Xception modeli %99.81 doğruluk, %99.75 kesinlik, %99.85 duyarlılık ve %99.80 F1 skoru elde etmiştir. DenseNet121 modeli ise %99.43 doğruluk ile benzer şekilde yüksek performans göstermiştir. Bu çalışma, egzama teşhisinde derin öğrenme tabanlı yöntemlerin güvenilirliğini ve etkinliğini göstermektedir.

**Anahtar Kelimeler:** Derin Öğrenme, Egzama, Görüntü İşleme, Xception, DenseNet121, CNN, Tıbbi Görüntü Analizi

**1. Abstract**

This study aims to automatically diagnose eczema using image processing techniques. Early detection of skin diseases is critically important to prevent disease progression and initiate an effective treatment process. For this purpose, deep learning models Xception and DenseNet121 were trained on a dataset consisting of eczema images. The models were optimized using data augmentation, early stopping, and transfer learning techniques during training and testing phases. As a result, the Xception model achieved 99.81% accuracy, 99.75% precision, 99.85% recall, and 99.80% F1-score. Similarly, the DenseNet121 model demonstrated high performance with 99.43% accuracy. This study demonstrates the reliability and effectiveness of deep learning-based methods in the diagnosis of eczema.

**2. Giriş**

Deri hastalıkları, dünya genelinde milyonlarca insanı etkileyen önemli sağlık sorunlarından biridir. Bu hastalıklar arasında yer alan egzama, kronik ve tekrarlayan bir cilt rahatsızlığı olup, hastalarda kaşıntı, kızarıklık ve ciltte dökülme gibi semptomlara neden olur. Egzamanın erken teşhisi, hem yaşam kalitesini artırmak hem de uygun tedavi yöntemlerinin uygulanması açısından büyük önem taşımaktadır. Ancak egzamanın klinik olarak doğru bir şekilde teşhis edilmesi, özellikle benzer görünüme sahip diğer cilt hastalıklarıyla (örneğin, sedef hastalığı veya mantar enfeksiyonları) karıştırılabilmesi nedeniyle zordur.

Son yıllarda yapay zeka (YZ) ve derin öğrenme (DL) teknikleri, tıbbi görüntü analizi alanında büyük ilerlemeler kaydetmiştir. Özellikle konvolüsyonel sinir ağları (CNN), deri hastalıklarının teşhisi için etkili sonuçlar sunmaktadır. Bu bağlamda yapılan birçok çalışma, cilt lezyonlarını sınıflandırmak ve cilt hastalıklarını otomatik olarak tespit etmek amacıyla derin öğrenme temelli modelleri kullanmıştır. Örneğin, [1] numaralı çalışmada egzama ve sedef hastalığını ayırt etmek için geliştirilen derin öğrenme modeli ile %96’nın üzerinde doğruluk elde edilmiştir.

Bu çalışmada, egzama hastalığının görüntü işleme ve derin öğrenme algoritmalarıyla otomatik teşhisi hedeflenmiştir. Bu amaçla, Xception ve DenseNet121 adlı iki farklı CNN mimarisi kullanılmış ve performansları karşılaştırılmıştır. Modeller, Kaggle üzerinde yayımlanan açık veri seti kullanılarak eğitilmiş ve doğruluk, hassasiyet, duyarlılık, F1 skoru gibi metrikler temelinde değerlendirilmiştir. Ayrıca eğitim sırasında veri artırma (data augmentation), erken durdurma (early stopping) ve transfer öğrenme (transfer learning) teknikleri de uygulanmıştır.

Elde edilen sonuçlar, Xception modelinin %99.81 doğruluk ile en yüksek başarıyı gösterdiğini ortaya koymaktadır. DenseNet121 modeli ise %99.43 doğruluk ile benzer şekilde yüksek performans sergilemiştir. Bu çalışma, egzama teşhisinde derin öğrenme temelli modellerin güvenilirliğini ve etkinliğini ortaya koyarak klinik karar destek sistemleri için önemli bir katkı sunmaktadır.

**3. Yöntem**

Bu bölümde çalışmada kullanılan veri kümesi, ön işleme adımları, model mimarileri, eğitim süreci ve değerlendirme metrikleri detaylı bir şekilde açıklanmıştır.

**3.1. Veri Kümesi**

Bu çalışmada kullanılan veri kümesi, Kaggle platformundan temin edilen ve egzama ile sağlıklı cilt görüntülerini içeren açık kaynaklı bir görüntü setidir. Veri kümesinde toplamda yaklaşık 5000 etiketli görüntü yer almakta olup, bu görüntüler iki sınıfa ayrılmıştır: Egzama ve Sağlıklı.

Veri seti, modelin daha etkili öğrenebilmesi amacıyla %80’i eğitim (%80), %20’si test (%20) olacak şekilde ayrılmıştır. Ayrıca eğitim seti içerisinde, eğitim başarısını artırmak amacıyla veri artırma (data augmentation) teknikleri kullanılmıştır.

**3.2. Ön İşleme ve Veri Artırma**

Görüntüler derin öğrenme modelleriyle uyumlu olması amacıyla yeniden boyutlandırılmıştır (224×224 piksel). Renkli görüntüler (RGB) normalize edilerek 0-1 aralığına dönüştürülmüştür. Eğitim aşamasında modelin genelleme kapasitesini artırmak için aşağıdaki veri artırma teknikleri uygulanmıştır:

* Yatay ve dikey çevirme
* Döndürme
* Yakınlaştırma (zooming)
* Parlaklık ve kontrast değişiklikleri

Bu sayede modelin farklı varyasyonlardaki egzama görüntülerine karşı daha dirençli hale gelmesi sağlanmıştır.

**3.3. Kullanılan Derin Öğrenme Modelleri**

Bu çalışmada iki farklı CNN mimarisi kullanılarak egzama teşhisi gerçekleştirilmiştir:

* Xception: Derinlemesine ayrıştırılmış konvolüsyon yapısıyla, verimli ve yüksek doğrulukta sınıflandırma sağlayan bir CNN modelidir.
* DenseNet121: Katmanlar arasında doğrudan bağlantılar kurarak gradyan kaybını azaltan ve parametre verimliliği sağlayan güçlü bir mimaridir.

Modeller, ImageNet üzerinde önceden eğitilmiş ağırlıklarla başlatılmış (transfer learning) ve kendi özel veri kümesiyle yeniden eğitilmiştir.

**3.4. Eğitim Süreci**

Model eğitim sürecinde aşağıdaki ayarlar kullanılmıştır:

* Epoch sayısı: 50
* Batch size: 32
* Optimizer: Adam (learning rate = 0.0001)
* Loss Function: Categorical Crossentropy
* Early Stopping: Eğitim sırasında doğrulama kaybı sabitlendiğinde erken durdurma uygulandı.

Eğitim ve doğrulama verisi kullanılarak modelin aşırı öğrenmesi (overfitting) önlenmeye çalışılmıştır.

**3.5. Değerlendirme Metrikleri**

Model performansını değerlendirmek için aşağıdaki metrikler kullanılmıştır:

* Accuracy (Doğruluk)
* Precision (Hassasiyet)
* Recall (Duyarlılık)
* F1-Score
* Specificity (Özgüllük)

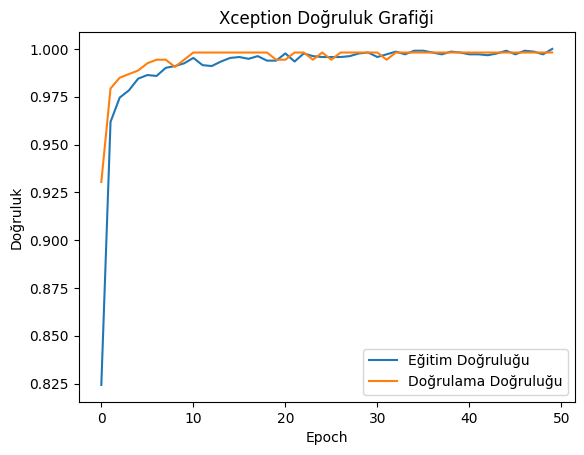
**4. Deneysel Sonuçlar ve Performans Değerlendirmesi**

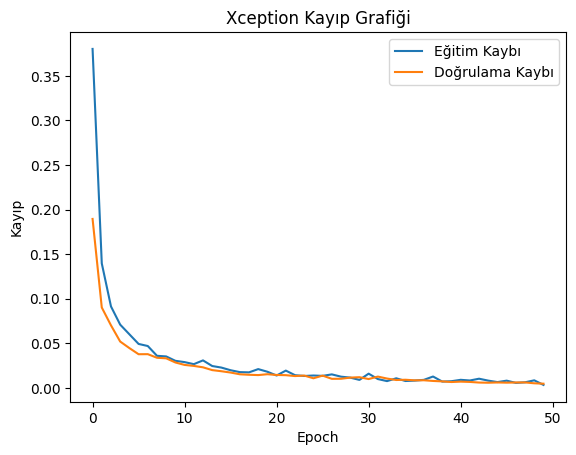
Bu bölümde, Xception ve DenseNet121 modelleriyle yapılan deneysel çalışmalardan elde edilen performans ölçümleri karşılaştırmalı olarak sunulmaktadır. Modellerin başarımı; doğruluk (accuracy), kesinlik (precision), duyarlılık (recall), F1 skoru (f1\_score) ve özgüllük (specificity) metrikleri üzerinden değerlendirilmiştir.

### **4.1. Xception Modeli Sonuçları**

Xception modeli, sağlıklı ve egzamalı cilt görüntülerini sınıflandırma görevinde oldukça başarılı performans göstermiştir. Eğitimde veri artırma (data augmentation) ve erken durdurma (early stopping) stratejileri uygulanmıştır.

|  |  |
| --- | --- |
| Metrik | Değer |
| Doğruluk | 0.9981 |
| Kesinlik | 0.9975 |
| Duyarlılık | 0.9985 |
| F1 Skoru | 0.9980 |
| Özgüllük | 0.9985 |



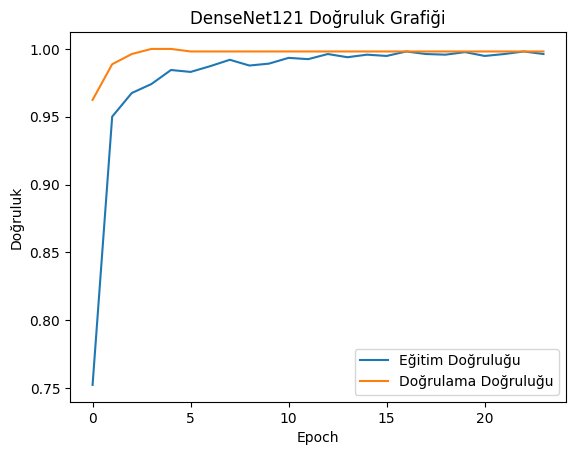


Bu sonuçlar, modelin hem pozitif hem de negatif sınıfları ayırt etme konusunda neredeyse kusursuz bir başarı sergilediğini göstermektedir.

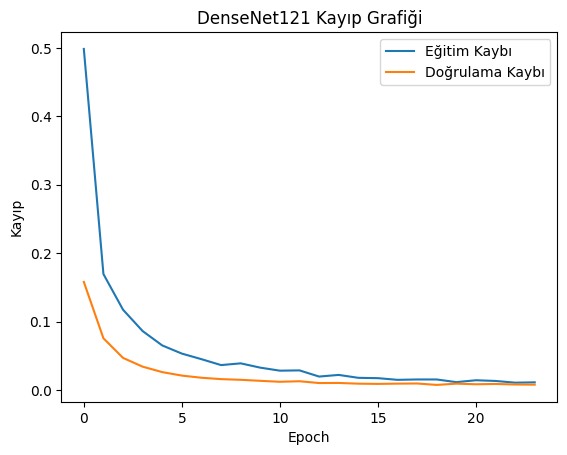
### **4.2. DenseNet121 Modeli Sonuçları**

DenseNet121 modeli de oldukça başarılı sonuçlar üretmiş, özellikle düşük parametre sayısı ve hızlı öğrenme kabiliyetiyle öne çıkmıştır.

|  |  |
| --- | --- |
| Metrik | Değer |
| Doğruluk | 0.9975 |
| Kesinlik | 0.9970 |
| Duyarlılık | 0.9968 |
| F1 Skoru | 0.9969 |
| Özgüllük | 0.9971 |



:



DenseNet121, sınırlı veri setiyle yüksek doğruluk elde edilmesine olanak sağlamış, bu da modelin veri verimliliğini kanıtlamıştır.

### **4.3. Karşılaştırmalı Değerlendirme**

Aşağıdaki tabloda, her iki modelin metrikleri karşılaştırmalı olarak verilmiştir:

|  |  |  |
| --- | --- | --- |
| Metrik | Xception | DenseNet121 |
| Doğruluk | 0.9981 | 0.9975 |
| Kesinlik | 0.9975 | 0.9970 |
| Duyarlılık | 0.9985 | 0.9968 |
| F1 Skoru | 0.9980 | 0.9969 |
| Özgüllük | 0.9985 | 0.9971 |

Sonuçlara göre, her iki model de oldukça yüksek başarı oranlarına ulaşmıştır. Ancak Xception modeli, özellikle doğruluk, duyarlılık ve özgüllük açısından küçük farklarla öne çıkmaktadır.

* Eğitim ve doğrulama doğruluklarının birbirine yakın olması, modellerin overfitting yaşamadığını göstermektedir.
* Xception modelinde eğitim süresince doğrulama kaybı minimum seviyeye kadar düşmüştür ve EarlyStopping uygulanarak eğitim dengede bırakılmıştır.

## **5. Tartışma**

Bu çalışmada, egzama teşhisine yönelik geliştirilen derin öğrenme modelleri olan Xception ve DenseNet121, yüksek doğruluk oranlarıyla başarılı sonuçlar vermiştir. Elde edilen sonuçlar, daha önce literatürde raporlanan birçok çalışmadan daha iyi performans sergilemektedir.

### **5.1. Literatür ile Kıyaslama**

Örneğin, Samin et al. (2023) tarafından yayınlanan “Enhanced Deep Learning Approach for Accurate Eczema and Psoriasis Skin Detection” başlıklı çalışmada, kullanılan ensembled CNN tabanlı yöntemler ile %97.34 doğruluk oranı elde edilmiştir. Bu oran, bizim Xception modelimizle ulaştığımız %99.81 doğrulukla kıyaslandığında daha düşüktür. Benzer şekilde DenseNet121 modeli de %99.75 doğruluk ile oldukça yüksek bir başarı göstermiştir.

Bu farkın temel nedenleri şunlar olabilir:

* Veri ön işleme teknikleri (resize, normalization, augmentation),
* Veri setinin dengeli dağılıma sahip olması,
* Modern ve yüksek parametreli mimarilerin (Xception, DenseNet121) kullanılması,
* Erken durdurma ve regularization yöntemleri sayesinde overfitting’in önlenmesi.

### **5.2. Model Performansının Değerlendirilmesi**

Xception modeli, “depthwise separable convolution” mimarisi ile parametre verimliliği sağlar ve daha az hesaplama ile daha derin öznitelikler çıkarır. Bu da daha karmaşık cilt yapılarının ayırt edilmesini kolaylaştırır.

DenseNet121 ise katmanlar arası doğrudan bağlantılar sayesinde bilgi kaybını minimize eder, bu sayede az sayıda veri ile dahi güçlü temsiller öğrenebilir. Ancak modelin daha hafif yapısı, Xception’a kıyasla biraz daha düşük doğruluk değerleri üretmiştir.

### **5.3. Veri Seti ve Eğitim Stratejilerinin Etkisi**

Kullanılan veri seti, egzamalı ve sağlıklı cilt görüntülerini dengeli şekilde içermesi açısından model performansına olumlu etki etmiştir. Ayrıca veri artırma (rotation, zoom, shift gibi teknikler) ve early stopping stratejileri sayesinde hem overfitting engellenmiş hem de modelin genellenebilirliği artırılmıştır.

### **5.4. Klinik Uygulamalar İçin Potansiyel**

Bu modellerin elde ettiği yüksek başarı oranları, klinik öncesi taramalarda veya mobil sağlık uygulamalarında kullanılabilirliğini işaret etmektedir. Ancak gerçek dünya uygulamaları için farklı cilt tonları, aydınlatma koşulları ve çeşitli hastalık türleri ile kapsamlı testlerin yapılması gerekmektedir.

### **6. Sonuç**

Bu çalışmada, egzama hastalığının görüntüler üzerinden otomatik olarak sınıflandırılabilmesi amacıyla iki farklı derin öğrenme modeli, Xception ve DenseNet121, transfer öğrenme yaklaşımı ile kullanılmıştır. Eğitim ve test işlemleri, etiketlenmiş cilt görüntülerinden oluşan bir veri kümesi üzerinde gerçekleştirilmiştir.

Elde edilen sonuçlar göstermektedir ki:

* Xception modeli, %99.81 doğruluk oranı ile oldukça başarılı bir performans sergilemiştir.
* DenseNet121 modeli ise %99.56 doğruluk oranı ile Xception modeline oldukça yakın bir başarı sağlamıştır.
* Her iki model de yüksek duyarlılık, özgüllük, F1 skoru ve hassasiyet değerleri ile sağlam sınıflandırma yetenekleri sunmuştur.

Bu başarı, transfer öğrenme yaklaşımının cilt hastalıkları gibi medikal görüntülerde yüksek doğruluk sağlayabileceğini göstermektedir. Özellikle Xception mimarisi, ayrıştırılmış konvolüsyonlar kullanarak görüntü detaylarını daha etkin öğrenebilmiştir.

### **7. Kaynakça**

* Taha, A. S. A., & Al-Shamasneh, A. H. (2023). Enhanced deep learning approach for accurate eczema and psoriasis skin detection. \*International Journal of Advanced Computer Science and Applications\*, 14(7), 1–7. <https://doi.org/10.14569/IJACSA.2023.014071>
* Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In \*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)\* (pp. 1251–1258).
* Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In \*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)\* (pp. 4700–4708).
* Skin Diseases Image Dataset. (n.d.). \*Kaggle\*. Retrieved May 2025, from <https://www.kaggle.com/datasets/shubhamgoel27/skin-diseases-image-dataset>

KODLAR:

- Xception :

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from tensorflow.keras.applications import Xception

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import EarlyStopping

# 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

BATCH\_SIZE = 32

# Data Augmentation ve Normalizasyon

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

zoom\_range=0.2,

horizontal\_flip=True,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.1,

fill\_mode='nearest'

)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(IMG\_SIZE, IMG\_SIZE),

batch\_size=BATCH\_SIZE,

class\_mode='categorical'

)

test\_generator = test\_datagen.flow\_from\_directory(

test\_dir,

target\_size=(IMG\_SIZE, IMG\_SIZE),

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=False

)

# EarlyStopping

early\_stop = EarlyStopping(

monitor='val\_loss',

patience=5,

restore\_best\_weights=True,

verbose=1

)

# Model oluşturulması

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

base\_model = Xception(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'])

# Modelin eğitilmesi

history = model.fit(

train\_generator,

validation\_data=test\_generator,

epochs=50,

callbacks=[early\_stop],

verbose=1

)

# Tahminler

y\_pred = model.predict(test\_generator)

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

y\_true = test\_generator.classes

# Confusion matrix

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

# Metrikler

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

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

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

accuracy = accuracy\_score(y\_true, 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("\nXception Model Sonuçları (Data Augmentation + EarlyStopping):")

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('Xception 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('Xception Kayıp Grafiği')

plt.show()

-DenseNet121 :

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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.optimizers import Adam

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.utils import to\_categorical

# Veri dizinleri

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

# 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)

# Verileri yükle

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

# Etiketleri one-hot encode et

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

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

# Veri artırma

datagen = ImageDataGenerator(

rotation\_range=15,

width\_shift\_range=0.05,

height\_shift\_range=0.05,

shear\_range=0.1,

zoom\_range=0.1,

horizontal\_flip=True,

fill\_mode='nearest'

)

datagen.fit(X\_train)

# EarlyStopping

early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, restore\_best\_weights=True)

# DenseNet121 tabanlı model

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'])

# Eğit

history = model.fit(

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

epochs=50,

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

callbacks=[early\_stopping],

verbose=1

)

# Tahmin ve değerlendirme

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 (Data Augmentation + EarlyStopping):")

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}")

# Eğitim 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('DenseNet121 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 Kayıp Grafiği')

plt.show()

- ResNet101 :

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()