**DETAYLI ANLATIM**

Raporuma, model sonucu ile başlamak isterim. Model doğruluğu 1.00 ve sonuçlar veriler ile birçok kez test edildi.

insan yüzü, adam, insan, gülümsemek, gülüş, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Model Oluşturma Adımları:**

1. **Gerekli Kütüphanelerin Yüklenmesi**

Model eğitimine kullanılacak kütüphaneleri ekleyerek başladım.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import DenseNet121, MobileNet

from tensorflow.keras.models import Model, load\_model

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

from tensorflow.keras.optimizers import Adam, SGD

from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, classification\_report, roc\_curve, auc

import os

1. **Veri Seti Yükleme**

Model eğitim verisi olarak Face Mask Detection Dataset’i kullandım.

<https://www.kaggle.com/datasets/omkargurav/face-mask-dataset>

1. **Önceden Eğitilmiş (Pre-trained) Modellerin Hazırlanması**

Sınıflandırıcı olarak DenseNet121 modeli oluşturdum ve modelin öğrenme kapasitesini artırmak için dense katmanı ve dropout katmanı ekledim.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

1. **Veri Setinin Eğitim/Validasyon ve Test Olarak Ayrılması ve Eğitim/Validasyon Setinin Bölünmesi ve Modelin Eğitilmesi**

Veri eğitim için bölündü.

Resim dosyaları ve etiketleri oluşturuldu.

**Tüm veri seti (%100)** → **%80 eğitim, %20 test** olarak ayrıldı.

**Eğitim seti (%80)** tekrar **bölündü → %80 train, %20 validation**

Özetle;

**Eğitim Seti** → %64 (%80’in %80’i)

**Validasyon Seti** → %16 (%80’in %20’si)

**Test Seti** → %20 (test için ayrılan)

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Transfer Learning yaklaşımıyla önceden eğitilmiş bir model kullanılarak sınıflandırma yapıldı. Modelin eğitiminde erken durdurma ve öğrenme oranı azaltma gibi teknikler uygulandı. Eğitim sürecinin analizinde loss ve accuracy eğrileri incelendi. Elde edilen sonuçlar modelin performansını değerlendirirken overfitting olup olmadığını anlamak için kullanıldı.

Model Eğitimi Süreci:

* Başlangıçta 100 epok boyunca eğitim planlandı.
* Erken durdurma (Early Stopping) etkinleştiği için model 36. epokta durdu.
* ReduceLROnPlateau sayesinde öğrenme hızı (learning rate) düşürülerek modelin stabil hale gelmesi sağlandı.
* Eğitim boyunca accuracy ve loss değerlerinde istikrarlı bir iyileşme gözlemlendi.

Model Performansı:

* Eğitim doğruluğu (training accuracy) başlangıçta %82 seviyesindeyken ilerleyen epoc’larda %98-99 seviyelerine ulaştı.
* Validasyon doğruluğu (validation accuracy) başlangıçta %98 seviyelerinde başladı ve %99 civarında sabitlendi.
* Validasyon kaybı (validation loss) düşerek stabil bir hale geldi, bu da modelin overfitting yapmadığını gösteriyor. Model sonuçaları ile kontrol edilerek doğruluğu test edildi.

Test Sonuçları:

* Model, test setinde %100 doğruluk oranına ulaştı.
* Precision, recall ve f1-score değerleri 1.00, yani model hiçbir hatalı sınıflandırma yapmamış.
* Makro ve ağırlıklı ortalama da çok iyi sonuçlar veriyor.

**4. Test Verisi İle Tahmin (Prediction) ve Sonuç Analizi**

Eğitilen modelin en iyi kayıtlı ağırlıkları yüklendikten sonra test seti üzerinde tahminler gerçekleştirildi. **Model, maske tespitinde çok iyi bir performans sergiliyor. Test sonuçlarına göre hata oranı yok denecek kadar az ve tüm metrikler (%99.87-100 aralığında) mükemmel seviyede.**

Modelin doğruluğunu değerlendirmek için aşağıdaki metrikler kullanıldı:

**ROC eğrisi:**

metin, çizgi, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

* AUC (Area Under Curve) değeri = 1.00
* Eğri neredeyse mükemmel bir sınıflandırıcıyı temsil edecek şekilde köşelere oturmuş.
* False Positive Rate düşük, True Positive Rate ise %100’e yakın. Modelin iki sınıfı ayırt etme yeteneği çok iyi seviyede ve AUC değeri **1.00** olduğu için model ideal bir sınıflandırıcı olarak değerlendirilebilir.

**Confusion Matrix (Karışıklık Matrisi)**

metin, ekran görüntüsü, diyagram, dikdörtgen içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

* Model maskeli ve maskesiz yüzleri ayırt etmekte neredeyse hatasız.
* Yanlış sınıflandırma sadece 1 örnekte olmuş, yani bir maskesiz kişi yanlışlıkla maskeli olarak etiketlenmiş.
* Diğer tüm tahminler doğru.

**Accuracy:**

Accuracy = (TP + TN) / (TP + TN + FP + FN)  
Accuracy = (760 + 750) / (760 + 750 + 1 + 0) = 0.9993 🡪99.93%

Modelin doğruluk oranı **%99.93** olup, test setinde neredeyse tüm tahminleri doğru yapmıştır.

**Precision:**

Precision (PPV) = TP / (TP + FP)  
Precision = 750 / (750 + 1) = 0.9987 🡪99.87%

Modelin maskeli olarak tahmin ettiği örneklerin %99.87’si gerçekten maskeli. Yanlış maskeli tahmini sayısı oldukça düşük.

**Recall:**

Recall (TPR) = TP / (TP + FN)  
Recall = 750 / (750 + 0) = 1.00 🡪100%

Model maskeli kişileri %100 doğrulukla tespit edebiliyor. Hiçbir maskeli kişi yanlışlıkla maskesiz olarak sınıflandırılmamış.

**Specificity:**

Specificity (TNR) = TN / (TN + FP)  
Specificity = 760 / (760 + 1) = 0.9987 🡪99.87%

Model, maskesiz kişileri ayırt etmede de %99.87 oranında başarılıdır.

**F1-score:**

F1-score = 2 × (Precision × Recall) / (Precision + Recall)  
F1-score = 2 × (0.9987 × 1.00) / (0.9987 + 1.00) 🡪0.9993 🡪99.93%

Precision ve Recall değerlerinin dengeli olması nedeniyle F1-score da oldukça yüksek (%99.93).

**Model Sayısal Çıktıları:**

runfile('/Users/haticeaksu/Desktop/untitled2.py', wdir='/Users/haticeaksu/Desktop')

Found 4833 validated image filenames belonging to 2 classes.

Found 1209 validated image filenames belonging to 2 classes.

Found 1511 validated image filenames belonging to 2 classes.

Epoch 1/100

/opt/anaconda3/lib/python3.11/site-packages/keras/src/trainers/data\_adapters/py\_dataset\_adapter.py:120: UserWarning: Your `PyDataset` class should call `super().\_\_init\_\_(\*\*kwargs)` in its constructor. `\*\*kwargs` can include `workers`, `use\_multiprocessing`, `max\_queue\_size`. Do not pass these arguments to `fit()`, as they will be ignored.

self.\_warn\_if\_super\_not\_called()

152/152 ━━━━━━━━━━━━━━━━━━━━ 0s 610ms/step - accuracy: 0.8209 - loss: 0.4265 /opt/anaconda3/lib/python3.11/site-packages/PIL/Image.py:981: UserWarning: Palette images with Transparency expressed in bytes should be converted to RGBA images

warnings.warn(

152/152 ━━━━━━━━━━━━━━━━━━━━ 125s 780ms/step - accuracy: 0.8215 - loss: 0.4254 - val\_accuracy: 0.9810 - val\_loss: 0.0619 - learning\_rate: 0.0010

Epoch 2/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 119s 775ms/step - accuracy: 0.9538 - loss: 0.1129 - val\_accuracy: 0.9851 - val\_loss: 0.0432 - learning\_rate: 0.0010

Epoch 3/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 656s 4s/step - accuracy: 0.9718 - loss: 0.0730 - val\_accuracy: 0.9818 - val\_loss: 0.0462 - learning\_rate: 0.0010

Epoch 4/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 125s 810ms/step - accuracy: 0.9797 - loss: 0.0580 - val\_accuracy: 0.9917 - val\_loss: 0.0404 - learning\_rate: 0.0010

Epoch 5/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 129s 837ms/step - accuracy: 0.9799 - loss: 0.0579 - val\_accuracy: 0.9835 - val\_loss: 0.0540 - learning\_rate: 0.0010

Epoch 6/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 128s 832ms/step - accuracy: 0.9772 - loss: 0.0507 - val\_accuracy: 0.9884 - val\_loss: 0.0365 - learning\_rate: 0.0010

Epoch 7/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 126s 819ms/step - accuracy: 0.9785 - loss: 0.0594 - val\_accuracy: 0.9909 - val\_loss: 0.0309 - learning\_rate: 0.0010

Epoch 8/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 125s 808ms/step - accuracy: 0.9818 - loss: 0.0547 - val\_accuracy: 0.9851 - val\_loss: 0.0387 - learning\_rate: 0.0010

Epoch 9/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 123s 801ms/step - accuracy: 0.9820 - loss: 0.0567 - val\_accuracy: 0.9868 - val\_loss: 0.0419 - learning\_rate: 0.0010

Epoch 10/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 124s 802ms/step - accuracy: 0.9824 - loss: 0.0503 - val\_accuracy: 0.9884 - val\_loss: 0.0405 - learning\_rate: 0.0010

Epoch 11/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 121s 788ms/step - accuracy: 0.9798 - loss: 0.0621 - val\_accuracy: 0.9917 - val\_loss: 0.0297 - learning\_rate: 0.0010

Epoch 12/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 126s 817ms/step - accuracy: 0.9815 - loss: 0.0516 - val\_accuracy: 0.9901 - val\_loss: 0.0368 - learning\_rate: 0.0010

Epoch 13/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 131s 848ms/step - accuracy: 0.9877 - loss: 0.0347 - val\_accuracy: 0.9901 - val\_loss: 0.0372 - learning\_rate: 0.0010

Epoch 14/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 128s 828ms/step - accuracy: 0.9814 - loss: 0.0551 - val\_accuracy: 0.9884 - val\_loss: 0.0363 - learning\_rate: 0.0010

Epoch 15/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 125s 812ms/step - accuracy: 0.9855 - loss: 0.0368 - val\_accuracy: 0.9926 - val\_loss: 0.0346 - learning\_rate: 0.0010

Epoch 16/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 124s 807ms/step - accuracy: 0.9807 - loss: 0.0571 - val\_accuracy: 0.9859 - val\_loss: 0.0481 - learning\_rate: 0.0010

Epoch 17/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 125s 806ms/step - accuracy: 0.9874 - loss: 0.0343 - val\_accuracy: 0.9909 - val\_loss: 0.0380 - learning\_rate: 1.0000e-04

Epoch 18/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 216s 1s/step - accuracy: 0.9854 - loss: 0.0379 - val\_accuracy: 0.9892 - val\_loss: 0.0398 - learning\_rate: 1.0000e-04

Epoch 19/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 442s 3s/step - accuracy: 0.9846 - loss: 0.0415 - val\_accuracy: 0.9909 - val\_loss: 0.0337 - learning\_rate: 1.0000e-04

Epoch 20/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 127s 826ms/step - accuracy: 0.9829 - loss: 0.0410 - val\_accuracy: 0.9934 - val\_loss: 0.0344 - learning\_rate: 1.0000e-04

Epoch 21/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 127s 824ms/step - accuracy: 0.9871 - loss: 0.0386 - val\_accuracy: 0.9917 - val\_loss: 0.0344 - learning\_rate: 1.0000e-04

Epoch 22/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 494s 3s/step - accuracy: 0.9852 - loss: 0.0405 - val\_accuracy: 0.9892 - val\_loss: 0.0409 - learning\_rate: 1.0000e-05

Epoch 23/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 124s 808ms/step - accuracy: 0.9819 - loss: 0.0487 - val\_accuracy: 0.9917 - val\_loss: 0.0314 - learning\_rate: 1.0000e-05

Epoch 24/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 195s 1s/step - accuracy: 0.9826 - loss: 0.0447 - val\_accuracy: 0.9901 - val\_loss: 0.0363 - learning\_rate: 1.0000e-05

Epoch 25/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 141s 918ms/step - accuracy: 0.9858 - loss: 0.0420 - val\_accuracy: 0.9901 - val\_loss: 0.0372 - learning\_rate: 1.0000e-05

Epoch 26/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 131s 851ms/step - accuracy: 0.9873 - loss: 0.0415 - val\_accuracy: 0.9876 - val\_loss: 0.0351 - learning\_rate: 1.0000e-05

Epoch 27/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 131s 851ms/step - accuracy: 0.9825 - loss: 0.0499 - val\_accuracy: 0.9892 - val\_loss: 0.0368 - learning\_rate: 1.0000e-06

Epoch 28/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 131s 850ms/step - accuracy: 0.9851 - loss: 0.0418 - val\_accuracy: 0.9892 - val\_loss: 0.0404 - learning\_rate: 1.0000e-06

Epoch 29/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 127s 824ms/step - accuracy: 0.9818 - loss: 0.0467 - val\_accuracy: 0.9901 - val\_loss: 0.0376 - learning\_rate: 1.0000e-06

Epoch 30/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 128s 828ms/step - accuracy: 0.9820 - loss: 0.0510 - val\_accuracy: 0.9909 - val\_loss: 0.0409 - learning\_rate: 1.0000e-06

Epoch 31/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 121s 782ms/step - accuracy: 0.9849 - loss: 0.0411 - val\_accuracy: 0.9901 - val\_loss: 0.0404 - learning\_rate: 1.0000e-06

Epoch 32/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 125s 815ms/step - accuracy: 0.9847 - loss: 0.0448 - val\_accuracy: 0.9868 - val\_loss: 0.0389 - learning\_rate: 1.0000e-07

Epoch 33/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 121s 785ms/step - accuracy: 0.9831 - loss: 0.0425 - val\_accuracy: 0.9909 - val\_loss: 0.0371 - learning\_rate: 1.0000e-07

Epoch 34/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 121s 786ms/step - accuracy: 0.9885 - loss: 0.0364 - val\_accuracy: 0.9909 - val\_loss: 0.0385 - learning\_rate: 1.0000e-07

Epoch 35/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 122s 793ms/step - accuracy: 0.9768 - loss: 0.0619 - val\_accuracy: 0.9892 - val\_loss: 0.0376 - learning\_rate: 1.0000e-07

Epoch 36/100

152/152 ━━━━━━━━━━━━━━━━━━━━ 123s 795ms/step - accuracy: 0.9851 - loss: 0.0349 - val\_accuracy: 0.9901 - val\_loss: 0.0347 - learning\_rate: 1.0000e-07

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')` or `keras.saving.save\_model(model, 'my\_model.keras')`.

Model kaydedildi: /Users/haticeaksu/Desktop/mask\_detector\_model.h5

48/48 ━━━━━━━━━━━━━━━━━━━━ 57s 642ms/step

Important

Figures are displayed in the Plots pane by default. To make them also appear inline in the console, you need to uncheck "Mute inline plotting" under the options menu of Plots.

**precision recall f1-score support**

**0 1.00 1.00 1.00 761**

**1 1.00 1.00 1.00 750**

**Görsel Sonuçlar:**

Örnek Çıktı:

insan yüzü, kadın, ekran görüntüsü, giyim içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Örnek Çıktı:

insan yüzü, kişi, şahıs, ekran görüntüsü, gülümsemek, gülüş içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Örnek Çıktı:

insan yüzü, ekran görüntüsü, kadın, kolaj içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Örnek Çıktı:

insan yüzü, ekran görüntüsü, adam, insan, kolaj içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.