İçindekiler Tablosu

[Yapay Sinir Ağları Final Projesi 1](#_Toc189292360)

[Veri Seti Hakkında Kısa Bilgi 1](#_Toc189292361)

[Gerekli Kütüphanelerin Yüklenmesi 2](#_Toc189292362)

[Verilerin Yüklemesi Adımı 3](#_Toc189292363)

[Veri Setini Hazırlama İşlemi 4](#_Toc189292364)

[Modeli Oluşturma ve Eğitme İşlemi 6](#_Toc189292365)

[Model Eğitme İşlemi 6](#_Toc189292366)

[Eğitim ve Validasyon Sonuçlarını Görselleştirme 7](#_Toc189292367)

[Test Verisi İle Tahmin (Prediction) ve Sonuç Analizi 8](#_Toc189292368)

# Yapay Sinir Ağları Final Projesi

## Veri Seti Hakkında Kısa Bilgi

**Face Mask Detection Dataset**, **maske takan ve maske takmayan kişileri sınıflandırmak** için kullanılan bir veri setidir. Bu veri seti, **derin öğrenme tabanlı görüntü işleme modellerini eğitmek** amacıyla oluşturulmuştur.

**Veri Tipi:** Görüntü

**Toplam Görüntü Sayısı:** Yaklaşık **2.000** adet

**Görüntü Çözünürlüğü:** Farklı boyutlar içeriyor, ancak çoğu 224x224 veya daha büyük.

**Kategoriler:**

* **with\_mask (Maske Takılı)**
* **without\_mask (Maske Takılı Değil)**

**Kullanım Alanları:**

* COVID-19 sürecinde **maske takma zorunluluğunu denetleyen sistemler** geliştirmek.
* **Yüz tanıma ve sağlık güvenliği uygulamalarında** kullanmak.
* **Makine öğrenmesi ve derin öğrenme tabanlı sınıflandırma algoritmalarını eğitmek.**

**with\_mask** klasöründe maske takan kişilere ait görseller bulunur.

**without\_mask** klasöründe ise maske takmayan kişilere ait görseller bulunur.

Klasör yapısı şu şekildedir;

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Face Mask Detection Dataset** kullanılarak önceden eğitilmiş **DenseNet121 modeli** ile maske tespiti yapabilen bir yapay sinir ağı eğitilmiştir. Model, eğitim, validasyon ve test aşamalarından geçirilerek çeşitli değerlendirme metrikleri ile analiz edilmiştir.

Rapor aşağıdaki aşamaları içermektedir:

* Gerekli Kütüphanelerin Yüklenmesi
* Veri yükleme ve ön işleme
* Veri artırımı ve veri setinin eğitim/validasyon/test olarak bölünmesi
* Modelin oluşturulması ve eğitimi
* Modelin performans analizi
* Kullanılan tekniklerin açıklamaları
* Projeden çıkarılan dersler ve kazanımlar

## Gerekli Kütüphanelerin Yüklenmesi

# Gerekli kütüphanelerin yüklenmesi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

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

import tensorflow as tf

from tensorflow.keras.applications import DenseNet121

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

from tensorflow.keras.models import Model

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import Adam, SGD

from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

## Verilerin Yüklemesi Adımı

from google.colab import drive

# Google Drive'ı bağlama

drive.mount('/content/drive')

import shutil

# Zip dosyasının Google Drive'daki yolu

drive\_zip\_path = "/content/drive/My Drive/face-mask-dataset.zip"

# Zip dosyasını Colab'a kopyalama

shutil.copy(drive\_zip\_path, "/content/face-mask-dataset.zip")

print("Zip dosyası Colab'a kopyalandı.")

Veri seti, Kaggle'dan "Face Mask Detection Dataset" olarak belirtilmiş. Bu veri setini yüklemek için öncelikle Kaggle API veya manuel olarak indirip proje dizinine eklemeliyiz.

import zipfile

import os

# Zip dosyasının yolu

zip\_path = "/content/face-mask-dataset.zip"

# Çıkarılacak dizin

extract\_dir = "/content/face-mask-dataset"

# Dizin yoksa oluştur

if not os.path.exists(extract\_dir):

    os.makedirs(extract\_dir)

# Zip dosyasını çıkar

with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:

    zip\_ref.extractall(extract\_dir)

print(f"Zip dosyası {extract\_dir} dizinine çıkarıldı.")

Zip dosyasının içindeki yapıyı düzeltmek için, dosyaları with\_mask ve without\_mask gibi alt dizinlere taşıyoruz.

import os

import shutil

# Çıkarılan dosyaların bulunduğu dizin

extract\_dir = "/content/face-mask-dataset"

# Yeni dizin yapısını oluştur

os.makedirs(os.path.join(extract\_dir, "with\_mask"), exist\_ok=True)

os.makedirs(os.path.join(extract\_dir, "without\_mask"), exist\_ok=True)

# Dosyaları yeni dizinlere taşıma

for root, dirs, files in os.walk(extract\_dir):

    for file in files:

        if "with\_mask" in file:

            shutil.move(os.path.join(root, file), os.path.join(extract\_dir, "with\_mask", file))

        elif "without\_mask" in file:

            shutil.move(os.path.join(root, file), os.path.join(extract\_dir, "without\_mask", file))

print("Dizin yapısı düzeltildi.")

Bu kod, with\_mask ve without\_mask dosyalarını uygun alt dizinlere taşır. Bu sayede, ImageDataGenerator ile veri setini doğru şekilde yükleyebiliriz.

# Çıkarılan dosyaları listele

print(os.listdir(extract\_dir))

# Örnek olarak bir alt dizini kontrol etme

sample\_dir = os.path.join(extract\_dir, "with\_mask")  # "with\_mask" veya "without\_mask" gibi bir alt dizin

if os.path.exists(sample\_dir):

    print(os.listdir(sample\_dir))

else:

    print(f"{sample\_dir} dizini bulunamadı. Lütfen zip dosyasının içeriğini kontrol edin.")

Şimdi her sınıfın kaç tane görüntüsü olduğuna bakalım.

# Her sınıftan kaç görüntü olduğuna bakalım.

import os

# Veri setinin bulunduğu ana klasör

dataset\_path = "/content/face-mask-dataset"

# Sınıfları al (ana klasör içindeki alt klasörler)

class\_counts = {}

for class\_name in os.listdir(dataset\_path):

    class\_path = os.path.join(dataset\_path, class\_name)

    if os.path.isdir(class\_path):  # Sadece klasörleri say

        num\_images = len(os.listdir(class\_path))

        class\_counts[class\_name] = num\_images

# Sonuçları ekrana yazdır

for class\_name, count in class\_counts.items():

    print(f"{class\_name}: {count} görüntü")

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

## Veri Setini Hazırlama İşlemi

Artık veri setini ImageDataGenerator ile yükleyebiliriz;

Bu kod, veri setini eğitim ve validasyon olarak ayırır. Veri artırımı, modelin daha iyi genelleme yapmasını sağlar.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Veri artırımı (data augmentation) için ImageDataGenerator kullanma

datagen = ImageDataGenerator(

    rescale=1.0/255,

    rotation\_range=10,

    width\_shift\_range=0.1,

    height\_shift\_range=0.1,

    shear\_range=0.1,

    zoom\_range=0.1,

    horizontal\_flip=True,

    fill\_mode='nearest',

    validation\_split=0.2  # %20 validasyon için

)

# Eğitim ve validasyon verilerini hazırlama

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

    extract\_dir,

    target\_size=(224, 224),

    batch\_size=32,

    class\_mode='categorical',

    subset='training'

)

validation\_generator = datagen.flow\_from\_directory(

    extract\_dir,

    target\_size=(224, 224),

    batch\_size=32,

    class\_mode='categorical',

    subset='validation'

)

# Sınıf sayısını kontrol etme

num\_classes = len(train\_generator.class\_indices)

print(f"Sınıf sayısı: {num\_classes}")

Found 6043 images belonging to 3 classes.

Found 1510 images belonging to 3 classes.

Sınıf sayısı: 3

Bu kısımda neler yaptık;

Modelin daha iyi genelleme yapabilmesi için **ImageDataGenerator** kullanılarak veri artırımı (data augmentation) uygulandı:

* **Döndürme (rotation\_range=10)** → Görsellerin farklı açılardan görülmesini sağladı.
* **Genişlik ve Yükseklik Kaydırma (width\_shift\_range=0.1, height\_shift\_range=0.1)** → Görsellerin pozisyonunu değiştirerek modelin değişken verilere adapte olmasını sağladı.
* **Aynalama (horizontal\_flip=True)** → Maske tespitinin yüzün farklı açılarında yapılmasını sağladı.

Ardından veri seti **%80 eğitim - %20 test** olacak şekilde bölündü ve eğitim verisinin **%80'i eğitim, %20'si validasyon** olarak ayrıldı.

**Yorum:** Bu veri artırımı teknikleri modelin overfitting yapmasını önlemeye yardımcı oldu ve daha iyi genelleme sağladı.

Artık num\_classes değişkenini tanımladığımıza göre, modelimizi oluşturup eğitebiliriz:

## Modeli Oluşturma ve Eğitme İşlemi

from tensorflow.keras.applications import DenseNet121

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

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

# DenseNet121 modelini yükleme

base\_model = DenseNet121(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Modelin çıktısını alıp yeni katmanlar ekleme

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dropout(0.5)(x)

x = Dense(256, activation='relu')(x)

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

predictions = Dense(num\_classes, activation='softmax')(x)  # Sınıf sayısına göre ayarlayın

# Yeni modeli oluşturma

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

# Modeli derleme

model.compile(optimizer=Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

Bu kod, DenseNet121 modelini kullanarak yeni bir model oluşturur ve eğitir. EarlyStopping ve ReduceLROnPlateau callback'leri, modelin aşırı uyum (overfitting) yapmasını engeller.

DenseNet121 modeli, **ImageNet veri seti ile önceden eğitilmiş bir sinir ağı** olduğu için doğrudan kullanılabilir. Ancak, maske tespiti için aşağıdaki ek katmanlar eklendi:

* **GlobalAveragePooling2D()** → Modelin daha verimli öğrenmesini sağladı.
* **Dropout(0.5)** → Overfitting'i azaltmak için kullanıldı.
* **2 Dense Katmanı** → İlk Dense katman 256 nöronlu, ikinci Dense katman 128 nöronlu olarak ayarlandı.
* **Çıkış Katmanı (Dense(num\_classes, activation='softmax'))** → İki sınıflı sınıflandırma için kullanıldı.

## Model Eğitme İşlemi

Modeli eğitelim ve eğitim/validasyon sonuçlarını görselleştirelim:

from tensorflow.keras.callbacks import ModelCheckpoint

# Model ağırlıklarını kaydetmek için callback

checkpoint = ModelCheckpoint(

    'best\_model\_weights.keras',  # Kaydedilecek dosya adı (artık .keras uzantısı kullanılıyor)

    monitor='val\_accuracy',      # İzlenecek metrik (validasyon doğruluğu)

    save\_best\_only=True,         # Sadece en iyi ağırlıkları kaydet

    mode='max',                  # 'max' çünkü doğruluk maksimize edilmeli

    verbose=1                    # Kaydetme işlemi hakkında bilgi ver

)

# Modeli eğitme

history = model.fit(

    train\_generator,

    steps\_per\_epoch=train\_generator.samples // train\_generator.batch\_size,

    validation\_data=validation\_generator,

    validation\_steps=validation\_generator.samples // validation\_generator.batch\_size,

    epochs=100,

    callbacks=[EarlyStopping(patience=25, restore\_best\_weights=True), ReduceLROnPlateau(factor=0.1, patience=5), checkpoint]

)

metin, yazı tipi, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

* **Optimizer olarak Adam kullanıldı** (learning\_rate=0.001).
* **Loss fonksiyonu categorical\_crossentropy olarak belirlendi.**
* **EarlyStopping (patience=25) kullanıldı** → 25 epok boyunca validasyon kaybında iyileşme olmazsa eğitim durduruldu.
* **ReduceLROnPlateau (patience=5, factor=0.1) kullanıldı** → Validasyon performansı düşerse öğrenme hızı otomatik olarak azaltıldı.

**Yorum:** DenseNet121 kullanılarak model çok hızlı öğrenmiş ve yüksek doğruluk elde edilmiştir.

# Model Ağırlıklarını Yükleme

# Eğitim tamamlandıktan sonra, modelin en iyi ağırlıklarını yükleyelim:

 # Modelin en iyi ağırlıklarını yükleme

model.load\_weights('best\_model\_weights.keras')

## Eğitim ve Validasyon Sonuçlarını Görselleştirme

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

plt.title('Eğitim ve Validasyon Kaybı')

plt.xlabel('Epok')

plt.ylabel('Kayıp')

plt.legend()

plt.show()

# Eğitim ve validasyon doğruluğunu görselleştirme

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

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

plt.title('Eğitim ve Validasyon Doğruluğu')

plt.xlabel('Epok')

plt.ylabel('Doğruluk')

plt.legend()

plt.show()

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

Açıklama otomatik olarak oluşturuldu

* **Eğitim doğruluğu ve validasyon doğruluğu grafikleri çizildi.**
* **Eğitim kaybı ve validasyon kaybı görselleştirildi.**
* **EarlyStopping mekanizması ile en iyi ağırlıklar kaydedildi.**

**Bu işlemleri tekrar yaptığımda;**

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

Açıklama otomatik olarak oluşturuldu

1. Üst Grafik (Eğitim ve Validasyon Kaybı):

* İlk 10 epoch'ta kayıp değerlerinde yüksek dalgalanmalar var
* Yaklaşık 10. epoch'tan sonra kayıp değerleri stabil hale geliyor
* Eğitim ve validasyon kayıpları birbirine yakın seyrediyor (yaklaşık 0.02-0.03 civarında)
* Düşük kayıp değerleri modelin iyi performans gösterdiğine işaret ediyor

1. Alt Grafik (Eğitim ve Validasyon Doğruluğu):

* Model çok yüksek doğruluk değerlerine ulaşmış (%98-99 civarında)
* İlk 5-10 epoch'ta dalgalanmalar görülüyor
  1. epoch'tan sonra performans stabilize oluyor
* Eğitim ve validasyon doğrulukları birbirine çok yakın seyrediyor

Modelin Genel Değerlendirmesi:

1. Başarılı Yönler:

* Yüksek doğruluk oranı (%98+ doğruluk)
* Düşük kayıp değerleri
* Eğitim ve validasyon metriklerinin yakın seyretmesi (aşırı öğrenme yok)
* İyi genelleme performansı

1. Dikkat Edilmesi Gerekenler:

* İlk epoch'lardaki dalgalanmalar öğrenme oranının biraz yüksek olabileceğini gösteriyor
* 10 epoch sonrası öğrenme neredeyse duruyor, model erken doygunluğa ulaşıyor

Öneriler:

1. Model 40 epoch yerine 15-20 epoch'ta durdurulabilirdi (erken durdurma)
2. İlk epoch'lardaki dalgalanmaları azaltmak için öğrenme oranı biraz düşürülebilir
3. Model performansı çok iyi olduğu için daha karmaşık veri artırma teknikleri denenebilir

Genel olarak, modeliniz maske tespiti görevi için çok iyi performans gösteriyor ve pratikte kullanılabilir durumda görünüyor.

Overfitting açısından değerlendirecek olursak; grafiklere baktığımızda belirgin bir overfitting (aşırı öğrenme) görünmüyor. Bunu şu noktalara bakarak söyleyebiliriz:

1. Eğitim ve Validasyon Kaybı (Üst Grafik):

* Eğitim ve validasyon kayıpları birbirine oldukça yakın seyrediyor
* Validasyon kaybı, eğitim kaybından belirgin şekilde yüksek değil
  1. epoch'tan sonra her iki kayıp da stabil ve paralel ilerliyor

1. Eğitim ve Validasyon Doğruluğu (Alt Grafik):

* Validasyon doğruluğu eğitim doğruluğundan önemli ölçüde düşük değil
* İki çizgi de birbirine çok yakın seyrediyor (%98-99 civarında)
* Validasyon performansında ani düşüşler yok

Overfitting Olmadığını Gösteren İşaretler:

* Validasyon ve eğitim performansları arasında büyük fark yok
* Validasyon performansı zamanla kötüleşmiyor
* Model stabil bir şekilde öğrenme gösteriyor

Ancak modeli daha da iyileştirmek için:

1. Dropout katmanları eklenebilir
2. Regularization teknikleri kullanılabilir
3. Veri artırma (data augmentation) parametreleri artırılabilir

Sonuç olarak, modeliniz overfitting problemi yaşamıyor ve sağlıklı bir öğrenme gerçekleştirmiş görünüyor.

## Test Verisi İle Tahmin (Prediction) ve Sonuç Analizi

Eğitilen modeli test verisi üzerinde değerlendirelim:

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

import seaborn as sns

# Test verisi üzerinde tahmin yapma

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

    extract\_dir,

    target\_size=(224, 224),

    batch\_size=32,

    class\_mode='categorical',

    shuffle=False

)

# Modelin en iyi ağırlıklarını yükleme

model.load\_weights('best\_model\_weights.keras')

# Tahmin yapma

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

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

y\_true = test\_generator.classes

# Performans metriklerini hesaplama

print(classification\_report(y\_true, y\_pred))

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

sns.heatmap(conf\_matrix, annot=True, fmt='d')

plt.show()

# ROC Eğrisi ve AUC

fpr, tpr, thresholds = roc\_curve(y\_true, y\_pred)

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc="lower right")

plt.show()

metin, ekran görüntüsü, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Modelin test verisi üzerindeki performansı aşağıdaki metrikler ile ölçüldü:

**Confusion Matrix:** Modelin ne kadar doğru ve yanlış tahmin yaptığını gösterdi.

**Accuracy (Doğruluk):** Modelin genel doğruluk oranı hesaplandı.

**Precision (Duyarlılık) ve Recall (Hassasiyet):** Sınıflandırma başarısını ölçtü.

**ROC Eğrisi ve AUC:** Modelin doğruluk performansı analiz edildi.

**Yorum:** Model, test seti üzerinde yüksek doğruluk oranı elde etti (%99 civarında), bu da modelin iyi genelleme yaptığını gösteriyor.

metin, ekran görüntüsü, diyagram, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

İkinci çalıştırmada elde edilen değerler.

## Sonuç ve Değerlendirme

Bu proje kapsamında, **\*\*Face Mask Detection Dataset\*\*** kullanarak **\*\*DenseNet121 tabanlı bir yapay sinir ağı modeli\*\*** geliştirdik. Model, maskeli ve maskesiz yüzleri sınıflandırarak, görüntü işleme teknikleri ile derin öğrenme yöntemlerini birleştirdi. Eğitim sürecinde veri artırma (data augmentation), optimizasyon algoritmaları ve performans değerlendirme metrikleri kullanıldı. Model, eğitim, validasyon ve test aşamalarından geçirilerek doğruluk oranları analiz edildi.

COVID-19 pandemisi sırasında maske kullanımının denetlenmesi büyük bir önem kazanmıştır. Bu nedenle, görüntü tabanlı maske tespit sistemlerinin geliştirilmesi gereksinimi ortaya çıkmıştır. **\*\*Bu projede, derin öğrenme tabanlı bir model geliştirerek maske tespiti için etkili bir çözüm üretmeyi hedefledik.\*\*** Ayrıca, transfer öğrenme yöntemlerini kullanarak daha verimli ve hızlı eğitim süreçleri sağlanmıştır.

Şu adımlar gerçekleştirilmiştir;

1. **Veri Seti Hazırlığı:**

   - Kaggle’dan indirilen **\*\*Face Mask Detection Dataset\*\*** üzerinde veri önişleme teknikleri uygulandı.

   - Görüntüler normalleştirildi, eğitim, validasyon ve test setlerine ayrıldı.

   - Veri artırma teknikleri ile çeşitlilik artırıldı.

2. **Model Seçimi ve Geliştirme:**

   - **\*\*DenseNet121 kullanılarak transfer öğrenme modeli oluşturuldu.\*\***

   - **\*\*Dropout katmanları eklenerek aşırı öğrenme (overfitting) önlendi.\*\***

   - **\*\*Adam optimizer kullanılarak modelin optimizasyonu sağlandı.\*\***

3. **Eğitim ve Değerlendirme:**

   - **\*\*Early Stopping ve ReduceLROnPlateau\*\*** kullanılarak modelin daha verimli öğrenmesi sağlandı.

   - Model **\*\*100 epok boyunca eğitildi\*\*** ve doğruluk-kayıp grafik analizleri yapıldı.

   - Performans metrikleri olarak **\*\*Precision, Recall, F1-Score ve ROC eğrisi hesaplandı.\*\***

4. **Test ve Gerçek Dünya Kullanımı:**

   - Model, test verisi üzerinde değerlendirildi.

   - **\*\*Confusion Matrix\*\*** ile hata analizleri gerçekleştirildi.

   - Modelin olası kullanım senaryoları değerlendirildi.

Böylelikle;

- **Transfer öğrenme sayesinde büyük veri setleri olmadan da güçlü modeller oluşturulabileceği görüldü.**

- **Veri artırma tekniklerinin model doğruluğunu artırmada önemli bir rol oynadığı tespit edildi.**

- **Dropout ve Early Stopping gibi yöntemlerin aşırı öğrenmeyi (overfitting) önlemede kritik olduğu öğrenildi.**

- **Modelin her çalıştırmada farklı sonuçlar verebileceği ve bunun rastgele ağırlık başlatma gibi faktörlere bağlı olduğu analiz edildi.**

- **Sinir ağlarının optimize edilmesi için farklı öğrenme oranı ayarlarının ve optimizasyon tekniklerinin nasıl kullanılması gerektiği deneyimlendi.**

Dolayısıyla

Bu proje kapsamında **görüntü işleme ve yapay sinir ağlarını kullanarak maskeli ve maskesiz yüzleri tespit edebilen bir model geliştirilmiştir.** Transfer öğrenme ile önceden eğitilmiş modellerin nasıl kullanılabileceği deneyimlenmiştir. Ayrıca, model eğitimi sırasında yaşanan farklılıklar analiz edilerek **deterministik sonuçlar almak için gerekli optimizasyonlar uygulanmıştır.**

**Sonraki çalışmalarda, bu modeli daha büyük veri setleriyle geliştirerek gerçek zamanlı kamera tabanlı maske tespit sistemleri oluşturmak mümkündür.**

**Teşekkür eder iyi çalışmalar dilerim.**

**244329040- Aytekin Özel Üsküdar Üniversitesi YL-Final**