**KARADENİZ TEKNİK ÜNİVERSİTESİ**

**OF TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**Görüntülerden Sahte/Gerçek Yüz Ayrımının Yapılması**

**(Fake Face Detection)**

**GÖRÜNTÜ İŞLEME VİZE PROJESİ**

**369539 - Hasan Cihad Altay**

**2022-2023 BAHAR DÖNEMİ**

# Özet

Bu çalışmada, sahte yüzlerin tespiti için üç farklı CNN modeli sunulmaktadır: MobileNetV2, DenseNet201 ve önerilen özgün CNN modeli. Bu modeller, sahte yüzlerin tespiti için yüksek doğruluk ve güvenilirlik sağlamak amacıyla tasarlanmıştır. Öncelikle, eğitim süreci için büyük çaplı bir veri seti toplanmış ve etiketlenmiştir. Bu veri seti, gerçek ve sahte yüz görüntülerini içeren geniş bir çeşitlilik sunmaktadır. Ardından, eğitim süreci için MobileNetV2 ve DenseNet201 gibi önceden eğitilmiş ağırlıklar kullanılmıştır. Bu modeller, transfer öğrenme yöntemleriyle geliştirilmiştir ve sahte yüzlerin tespiti için etkili bir performans sergilemektedir. Ayrıca, önerilen özgün CNN modeli de geliştirilmiştir. Bu model, sahte yüzlerin tespiti için özel olarak tasarlanmış ve optimize edilmiştir. Kendi CNN modelimiz, veri setindeki özellikleri daha iyi öğrenmek ve daha hassas bir tespit sağlamak amacıyla eğitim sürecinde kullanılmıştır.

**Veri Ön İşleme:**

- Veri boyutlandırma: Görüntü özgün CNN modeli için 256x256, MobilenetV2 ve Densenet201 için 224x224 piksel olarak ayarlanır.

- Veri normalizasyonu: Görüntü pikselleri 0 ile 1 arasında normalizasyon işlemi uygulanır.

**Özellik Seçimi ve Çıkarımı, Model Oluşturma ve Son İşleme**

- Özgün CNN modeli için Şekil 5’te özeti verilen model,

- MobilenetV2 modeli için Şekil 7’de özeti verilen model,

- Densenet201 modeli için Şekil 9’da özeti verilen model kullanılmıştır.

**Giriş görüntüsünün alınması:**

Modelin girişine, gerçek veya sahte bir insan yüzü görüntüsü sağlanır

**Şekil 1.** Projenin genel şeması

Eğitim tamamlandıktan sonra, modeller test veri seti üzerinde değerlendirilmiştir. Elde edilen sonuçlar, her üç modelin de sahte yüzlerin tespiti konusunda yüksek performans sergilediğini göstermiştir. Modeller, gerçek yüzler ile sahte yüzler arasındaki farklılıkları belirlemek için farklı özelliklerin öğrenilmesini sağlamaktadır. Sonuç olarak, bu çalışmada sunulan MobileNetV2, DenseNet201 ve özgün CNN modeli, sahte yüzlerin tespiti konusunda yüksek doğruluk ve güvenilirlik sağlama potansiyeline sahiptir. Bu modeller, sahte yüzlerin otomatik olarak tespit edilmesiyle kişisel güvenliği artırabilir ve kötü niyetli faaliyetleri önleme konusunda önemli bir rol oynayabilir. Gelecekteki çalışmalar, farklı veri setleri üzerinde modellerin performansının daha da değerlendirilmesini ve yeni yöntemlerin keşfedilmesini içermelidir.

# 1. Giriş

Dijital görüntü manipülasyonu teknolojilerinin hızla ilerlemesiyle birlikte, sahte yüzlerin oluşturulması ve kullanılması potansiyel riskleri artırmıştır. Bu riskler, kişisel verilerin kötüye kullanımı, itibar kaybı, dolandırıcılık ve diğer kötü niyetli eylemler gibi bir dizi sorunu beraberinde getirmektedir. Bu nedenle, sahte yüzlerin tespit edilmesi ve doğrulanması, bireylerin güvenliğini sağlamak ve yanıltıcı uygulamalara karşı korumak için büyük önem taşımaktadır.

Geleneksel sahte yüz tespiti yöntemleri, bir dizi el işareti ve görsel ipucu temelinde çalışırken, bu yöntemlerin sınırlamaları bulunmaktadır. Örneğin, manuel denetim gerektirebilir, hatalı sonuçlara neden olabilir veya yeni ve sofistike sahte yüzlerin tanımlanmasını zorlaştırabilir. Bu zorluklarla başa çıkmak için, derin öğrenme teknikleri ve özellikle Evrişimli Sinir Ağları (CNN'ler), sahte yüz tespiti için etkili bir yaklaşım sunmaktadır.

Son yıllarda, sahte yüz tespiti alanında birçok araştırma yapılmıştır. Önceki çalışmalarda, farklı CNN modelleri ve öğrenme yöntemleri kullanılarak sahte yüzlerin tespiti ve doğrulanması üzerinde çalışılmıştır. Örneğin, ResNet, VGGNet ve Inception gibi popüler CNN modelleri sahte yüz tespiti için başarıyla kullanılmıştır. Ayrıca, transfer öğrenme teknikleri de kullanılarak, önceden eğitilmiş ağırlıkların yeni sahte yüz veri setlerine uygulanmasıyla modellerin performansı artırılmıştır. Bununla birlikte, her modelin kendine özgü avantajları ve dezavantajları vardır. MobileNetV2, hafif yapısı ve düşük hesaplama maliyeti nedeniyle sahte yüz tespiti için tercih edilebilir. DenseNet201 ise derin ağ yapısı sayesinde daha karmaşık özellikleri öğrenme yeteneğiyle öne çıkmaktadır. Özgün CNN modeli ise veri setine özgü özellikleri daha iyi öğrenebilme ve hassas bir tespit yapabilme potansiyeline sahiptir. Bu çalışmada, sahte yüzlerin tespiti için geliştirilen MobileNetV2, DenseNet201 ve özgün CNN modeli, önceki çalışmalardan faydalanarak ve veri setine özgü özellikleri dikkate alarak tasarlanmıştır. Amacımız, bu modellerin sahte yüzleri tespit etme ve doğrulama konusundaki performansını değerlendirmek ve sahte yüzlerin tespiti alanında yeni bir adım atarak güvenliği sağlamaktır.

# 2. Metadoloji

Önerilen yöntem, veri setinin toplanması, kullanılacak araçların ve dilin belirlenmesi, veri setindeki görüntülerin ön işlemesinin yapılması, model mimarisinin oluşturulması, modelin derlenmesi, eğitiminin ve doğrulamanın gerçekleştirilmesini içermektedir.

## 2.1. Veri seti

Bu çalışmadaki veri seti 74,184 adet gerçek-sahte görüntüye sahip insan yüzlerinden oluşmaktadır. Bu görüntülerden 60,000 tanesi fake insan yüzlerini içerirken 14.184 tanesi gerçek insan yüzlerini içermektedir. Eğitim sürecinin tekdüze olmaması ve modelin daha iyi sonuçlar vermesi için çeşitli kaynaklar taranmış ve çeşitli veri setleri ile görüntüler kullanılmıştır. Bu çeşitlilik, farklı görüntü tiplerini ve sahte yüzlerin farklı varyasyonlarını içeren geniş bir veri setinin oluşturulmasına olanak sağlamıştır. Böylece, model farklı kaynaklardan gelen veriler üzerinde eğitilerek, daha genel ve kapsayıcı bir öğrenme gerçekleştirebilmiştir. Bu sayede, modelin farklı tipteki sahte yüzleri tespit etme ve doğrulama yeteneği geliştirilmiş ve gerçek hayattaki senaryolara daha iyi uyum sağlaması hedeflenmiştir. Veri seti; Kaggle, generated.photos, Reddit, OpenSea, MyModernMet, Pinterest gibi siteler ve Google Görsellerde bulunan diğer kaynaklardan derlenmiştir. Bu çeşitli kaynaklar, geniş bir kapsamda gerçek ve sahte insan yüzlerini içermektedir. Kaggle, veri bilimi topluluğunun paylaştığı geniş bir veri seti deposu olduğu için çeşitli gerçek ve sahte insan yüzü görüntülerini içermektedir. generated.photos ise sahte yüzlerin üretildiği bir platformdur ve sahte yüz verilerini sağlamaktadır. Ayrıca, Google Görseller'de geniş bir yüz veri tabanı bulunmaktadır ve veri setinin oluşturulmasında buradaki kaynaklardan da yararlanılmıştır.

Veri seti, eğitim, doğrulama ve test için ayrılmıştır. Eğitim için veri setinin %70'i kullanılmıştır, böylece modelin temel öğrenme süreci bu büyük veri alt kümesi üzerinde gerçekleştirilmiştir. Doğrulama için veri setinin %20'si ayrılmış ve modelin performansının izlenmesi ve hiperparametre ayarlaması için kullanılmıştır. Geriye kalan %10'luk veri seti ise test için ayrılmış ve modelin nihai performansını değerlendirmek amacıyla kullanılmıştır. Bu ayrım, modelin eğitim sürecinde aşırı uyma (overfitting) durumunu kontrol etmek ve genellemesini değerlendirmek için önemli bir adımdır.

**Tablo 1.** Veri seti dağılımı

|  |  |  |  |
| --- | --- | --- | --- |
| **Sahte-Gerçek** | **Eğitim Verileri** | **Doğrulama Verileri** | **Test Verileri** |
| Gerçek | 9,929 | 2,837 | 1,418 |
| Sahte | 42,000 | 12,000 | 6,000 |
| Toplam | 51,929 | 14,837 | 7,418 |

## 2.2. Kullanılan Dil ve Araçlar

Bu projede Python programlama dili kullanılmıştır. Python, geniş bir kütüphane ekosistemi ve kolay okunabilir syntax yapısıyla makine öğrenimi ve derin öğrenme gibi görevler için popüler bir tercihtir.

Proje geliştirme aşamasında Jupyter Notebook kullanılmıştır. Jupyter Notebook, interaktif bir kod çalışma ortamı sunarak kodu adım adım çalıştırma, sonuçları görselleştirme ve belgelendirme yapma imkanı sağlamaktadır. Bu özellikleri sayesinde modelin oluşturulması ve test edilmesi süreçlerinde kullanıcı dostu bir ortam sunmaktadır.

TensorFlow kütüphanesi, derin öğrenme modellerinin oluşturulması, eğitimi ve değerlendirilmesi için kullanılmıştır. TensorFlow, yüksek performanslı hesaplama yetenekleri ve geniş bir derin öğrenme araç seti sunarak modelin verimli bir şekilde geliştirilmesine ve eğitilmesine yardımcı olmuştur. Ayrıca TensorFlow-GPU, GPU hızlandırma özelliği sayesinde eğitim sürecinde daha yüksek performans elde etmemizi sağlamıştır.

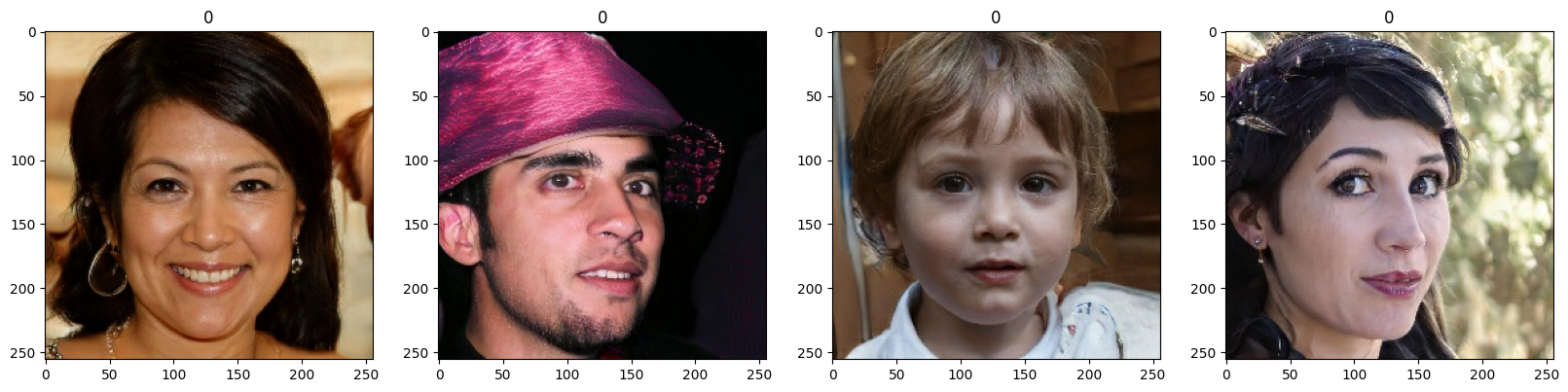
Proje için ayrıca OpenCV, matplotlib ve diğer bazı kütüphaneler de kullanılmıştır. OpenCV, görüntü işleme işlevleri için kullanılan bir kütüphanedir ve görüntülerin işlenmesi, dönüşümleri ve ön işlemeleri için kullanılmıştır. Matplotlib, veri görselleştirme için kullanılan bir kütüphanedir ve modelin performansını analiz etmek ve sonuçları görselleştirmek için kullanılmıştır. Bu dil ve araçlar, projenin başarılı bir şekilde gerçekleştirilmesi için sağlam bir temel oluşturmuştur. Veri işleme, model oluşturma ve sonuç analizi gibi adımların etkin bir şekilde gerçekleştirilmesini sağlamıştır.

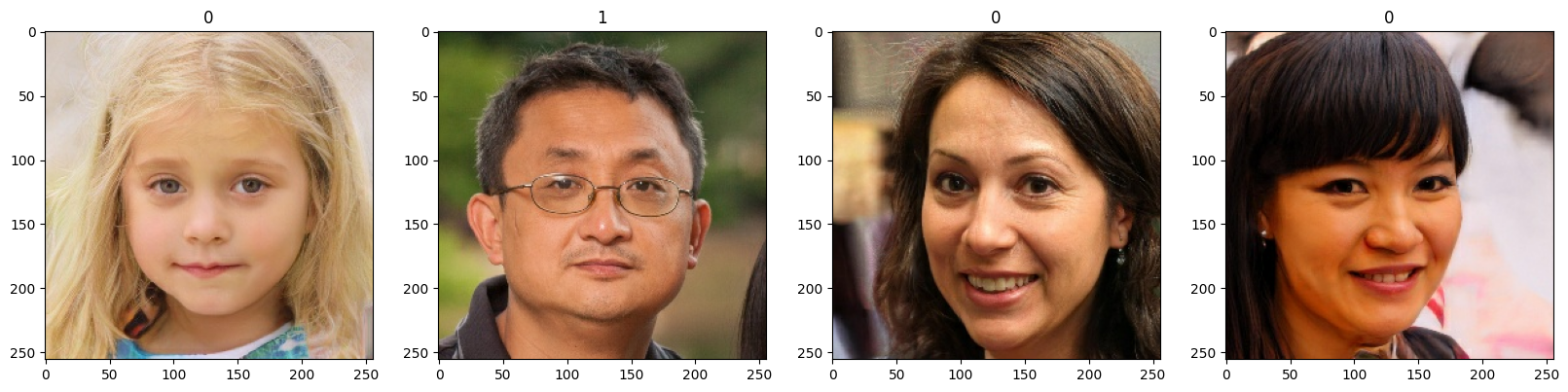
## 2.3. Görüntü Formatı

Bu projede görüntü formatı olarak JPEG (Joint Photographic Experts Group) formatı kullanılmıştır. JPEG, sıkıştırma algoritmasıyla tanınan ve fotoğrafçılıkta yaygın olarak kullanılan bir format olarak bilinir. JPEG formatı, yüksek kalitede görüntülerin daha düşük dosya boyutlarıyla saklanmasına olanak tanırken, görsel detayları korumak için kayıplı sıkıştırma yöntemini kullanır. Bu sayede, veri setindeki görüntülerin işlenmesi ve modele beslenmesi için uygun bir format seçilmiştir.

## 2.4. Ön İşleme

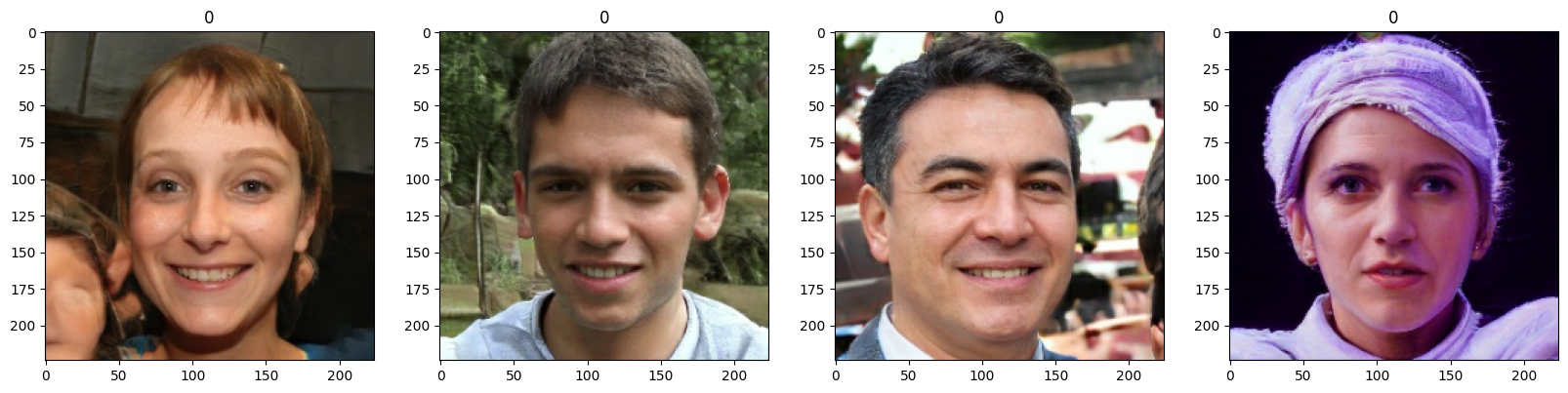
Veri ön işleme adımları kapsamında, 'fake\_real' dizininde bulunan görüntüler kullanılmıştır. Projede kullanılan MobilenetV2 ve Densenet201, 224x224 piksel boyutunda görüntüler beklemektedir. Bu nedenle bu modellerin eğitim ve test sürecinde görüntüler 224x224 olarak ayarlanmıştır. Diğer yandan özgün olarak geliştirilen CNN modelinin eğitim ve test sürecinde görüntüler 256x256 olarak ayarlanmıştır. Bu şekilde, farklı modellerin gereksinimlerini karşılayacak şekilde görüntüler ön işlenmiş ve kullanıma hazır hale getirilmiştir. Bunun yanında veri seti piksel değerlerini [0, 255] aralığından [0, 1] aralığına ölçeklendirmek için data.map fonksiyonu kullanılarak ön işleme adımı gerçekleştirilmiştir. Bu adım, görüntülerin daha iyi öğrenme performansı sağlaması ve daha istikrarlı bir eğitim süreci geçirmesi için önemlidir.

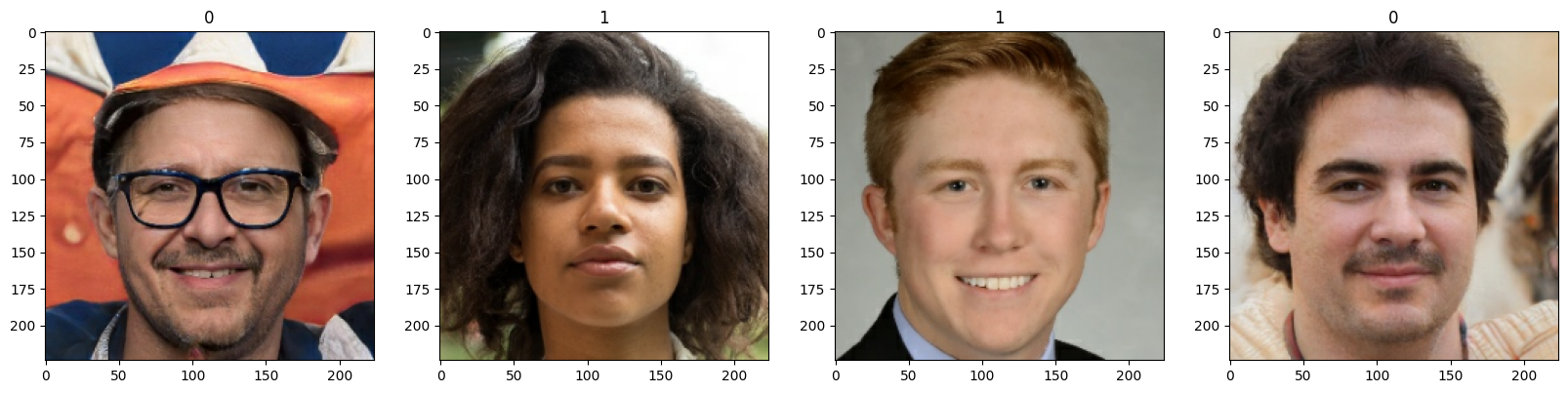


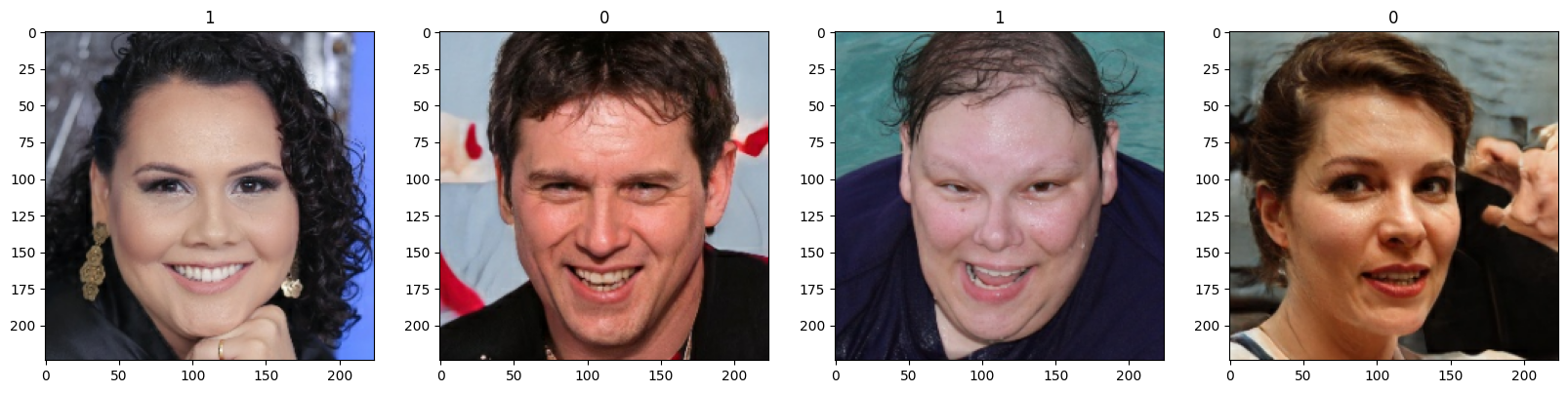




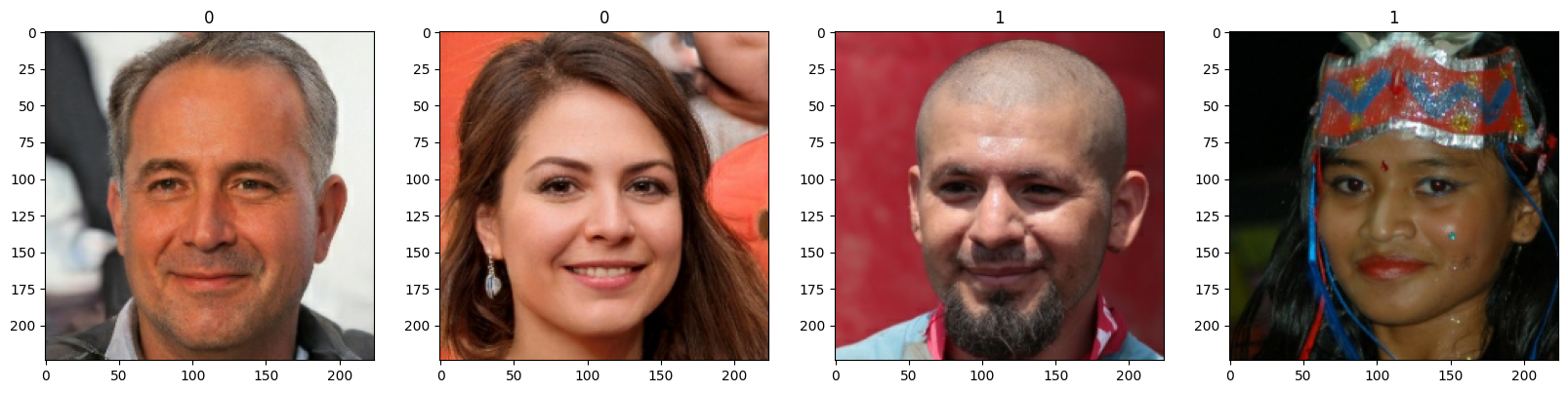
**Şekil 2.** Özgün CNN modeli için kullanılan görüntü örnekleri

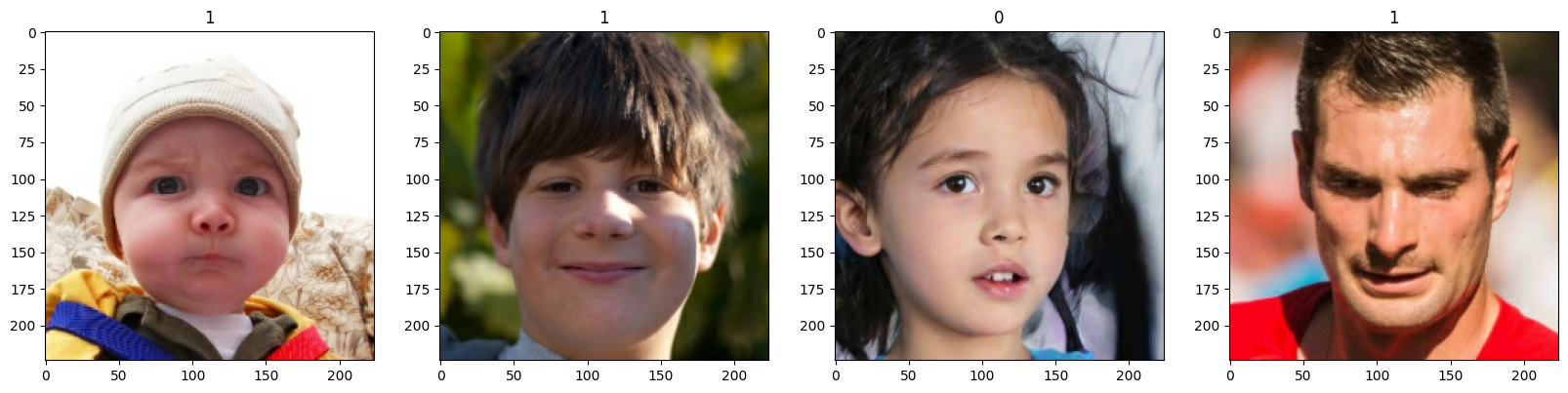
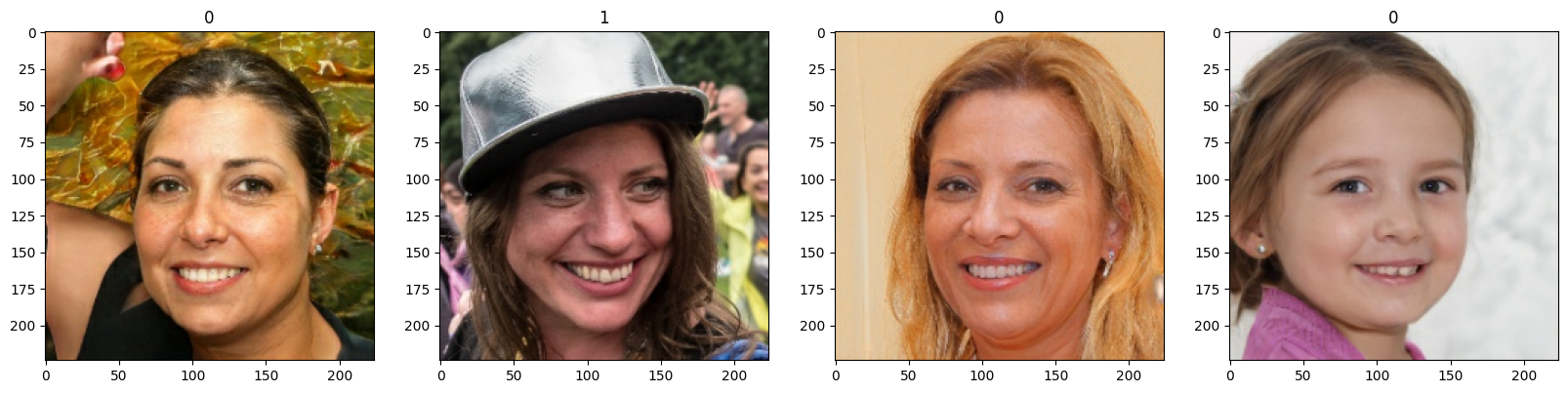






**Şekil 3.** MobilenetV2 için kullanılan görüntü örnekleri



**Şekil 4.** DenseNet201 için kullanılan görüntü örnekleri

## 2.5. Model Ağı

Daha önce de bahsettiğimiz gibi, proje için kullanılan ‘fake\_real’ veri seti, oluşturduğumuz özgün bir CNN modeli ve ön-eğitimli 2 model (MobilenetV2 ve Densenet201) kullanılarak eğitilmiştir.

## 2.6. Modellerin Eğitimi ve Doğrulanması

### 2.6.1. Önerilen Model

Önerilen özgün model, sahte yüzlerin tespiti için bir evrişimli sinir ağı (Convolutional Neural Network - CNN) kullanmaktadır. Bu model, görüntülerin boyutu, evrişim katmanları, dropout katmanları ve tam bağlantılı katmanlar gibi çeşitli bileşenlerden oluşmaktadır.

2.6.1.1. Giriş Katmanı

Görüntülerin boyutu (256, 256, 3) olarak kabul edilir, yani 256 piksel genişlik, 256 piksel yükseklik ve 3 renk kanalı (RGB) bulunur. Bu katman, modele görüntülerin girişini sağlar.

2.6.1.2. Evrişim Katmanları

İlk evrişim katmanı, 16 adet (3, 3) boyutunda filtre kullanır ve ReLU aktivasyon fonksiyonu ile aktive edilir. Ardından maksimum havuzlama (MaxPooling) katmanı gelir, boyutu varsayılan olarak belirlenir. İkinci evrişim katmanı, 32 adet (3, 3) boyutunda filtre kullanır ve yine ReLU aktivasyon fonksiyonu ile aktive edilir.

2.6.1.3. Dropout Katmanları

İlk Dropout katmanı, %40 oranında nöronları rastgele atarak aşırı uyumu önlemek için kullanılır. İkinci Dropout katmanı da aynı şekilde %40 oranında nöronları rastgele atar.

2.6.1.4. Düzleştirme Katmanı

Düzleştirme (Flatten) katmanı, evrişim katmanlarından elde edilen özellik haritalarını düz bir vektöre dönüştürür. Bu, tam bağlantılı (Dense) katmanlarda kullanılmak üzere verilerin düzenlenmesini sağlar.

2.6.1.5. Tam Bağlantılı Katmanlar

İlk tam bağlantılı katman, 256 nöron içerir ve ReLU aktivasyon fonksiyonu ile aktive edilir. Son katman, ikili sınıflandırma yapılacağı için 1 nöron içerir ve sigmoid aktivasyon fonksiyonu kullanır.

2.6.1.6. Derleme

Model, 'adam' optimizer ile derlenir. 'adam' optimizer, adaptif momentum tahmini kullanarak ağırlıkları günceller. Kayıp fonksiyonu olarak Binary Crossentropy kullanılır, çünkü bu bir ikili sınıflandırma problemini çözmektedir. Doğruluk (accuracy) metriği, modelin sınıflandırma performansını ölçmek için kullanılır.

|  |
| --- |
| Algoritma 1. Özgün CNN modelinin kodları |
| # *Kütüphanelerin import edilmesi* |
| from tensorflow.keras.models import Sequential |
| from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout |
|  |
| *# Modelin oluşturulması* |
| model = Sequential() |
| from keras import regularizers |
| model.add(Conv2D(16, (3,3), 1, activation='relu', input\_shape=(256,256,3))) |
| model.add(MaxPooling2D()) |
| model.add(Conv2D(32, (3,3), 1, activation='relu')) |
| model.add(MaxPooling2D()) |
| model.add(Dropout(0.4)) |
| model.add(Conv2D(16, (3,3), 1, activation='relu')) |
| model.add(MaxPooling2D()) |
| model.add(Dropout(0.4)) |
| model.add(Flatten()) |
| model.add(Dense(256, activation='relu')) |
| model.add(Dense(1, activation='sigmoid')) |
|  |
| # *Modelin derlenmesi* |
| model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy']) |

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 254, 254, 16) 448

max\_pooling2d (MaxPooling2D (None, 127, 127, 16) 0

)

conv2d\_1 (Conv2D) (None, 125, 125, 32) 4640

max\_pooling2d\_1 (MaxPooling (None, 62, 62, 32) 0

2D)

dropout (Dropout) (None, 62, 62, 32) 0

conv2d\_2 (Conv2D) (None, 60, 60, 16) 4624

max\_pooling2d\_2 (MaxPooling (None, 30, 30, 16) 0

2D)

dropout\_1 (Dropout) (None, 30, 30, 16) 0

flatten (Flatten) (None, 14400) 0

dense (Dense) (None, 256) 3686656

dense\_1 (Dense) (None, 1) 257

=================================================================

Total params: 3,696,625

Trainable params: 3,696,625

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Şekil 5.** Özgün modelin özeti

2.6.1.7. Özgün modelin doğruluk ve kayıp verileri

|  |  |
| --- | --- |
| C:\Users\user\fake_face_env\Scripts\epoch20 acc.png | C:\Users\user\fake_face_env\Scripts\epoch20-loss.png |
| a) Modelin eğitim ve doğrulama doğruluk eğrileri | b) Modelin eğitim ve doğrulama kayıp eğrileri |

**Şekil 6.** Özgün modelin eğitim ve doğruluk eğrileri

Şekil 6, eğitim ve doğrulama sırasında ağın öğrenme eğrilerini göstermektedir. Şekil 6.(a)’da gözlemlendiği üzere genel olarak doğrulama doğruluğu ve eğitim doğruluğu düzenli bir artış göstermiş ve %98.74 eğitim doğruluğu ve %93.22 doğrulama doğruluğuna ulaşmıştır. Şekil 6.(b)’de ise ağın eğitim kaybınınn düzenli olarak düştüğü görülmektedir. Doğrulama kaybında ise zaman zaman küçük artış eğilimleri olsa da genele baktığımız zaman büyük bir düşüş söz konusudur.

### 2.6.1.8. Özgün modelin sınıflandırma başarımları ve karmaşıklık matrisi

Bu bölümde, modelin sınıflandırma performansını değerlendirmek için kullanılan karmaşıklık matrisi ve elde edilen sınıflandırma başarımları sunulmuştur. Karmaşıklık matrisi, modelin gerçek ve tahmin edilen sınıflar arasındaki ilişkiyi gösterirken, sınıflandırma başarımları ise kesinlik, hassasiyet ve doğruluk gibi metriklerle modelin performansını değerlendirir.

Doğruluk (accuracy): Toplam örneklerin ne kadarının doğru sınıflandırıldığını gösterir. Yani doğru tahmin edilen örneklerin toplam örneklere oranını verir. Bu metrik’deki gibi hesaplanmıştır :

Kesinlik (precision): Pozitif olarak tahmin edilen örneklerin gerçekten pozitif olanların oranını gösterir. Yanlış pozitifleri minimize etmek için kullanılır.

Hassasiyet (recall): Gerçekten pozitif olan örneklerin ne kadarının doğru bir şekilde pozitif olarak tahmin edildiğini gösterir. Yanlış negatifleri minimize etmek için kullanılır.

F1 skoru: Kesinlik ve hassasiyet metriklerinin harmonik ortalamasını alarak hesaplanır. Bu, F1 skorunun düşük olan metrikler arasında denge kurarak modelin performansını daha kapsamlı bir şekilde değerlendirmesini sağlar.

Bu metrikler aşağıdaki denklemlerdeki gibi hesaplanmıştır (Doğru Pozitif (True Positive, TP), Yanlış Pozitif (False Positive, FP), Yanlış Negatif (False Negative, FN), Doğru Negatif (True Negative, TN)):

*Doğruluk (Accuracy) = (TP + TN) / (TP + TN + FP + FN)*  Denklem (1)

*Hassasiyet (Precision) = TP / (TP + FN)*  Denklem (2)

*Kesinlik (Recall) = TP / (TP + FP)*  Denklem (3)

*F1 Skoru = 2 \* (Kesinlik \* Hassasiyet) / (Kesinlik + Hassasiyet)* Denklem(4)

**Tablo 2.** Özgün CNN modelinin karmaşıklık matrisi

|  |  |  |
| --- | --- | --- |
|  | **Doğru** | **Yanlış** |
| **Pozitif** | 5972 | 22 |
| **Negatif** | 1281 | 125 |

**Tablo 3.** Özgün CNN modelinin sınıflandırma başarımları

|  |  |  |  |
| --- | --- | --- | --- |
| **Kesinlik (%)** | **Hassasiyet (%)** | **Doğruluk (%)** | **F1 Skoru (%)** |
| 99.63 | 97.94 | 98.01 | 98.77 |

### 2.6.2. MobilenetV2 modeli

MobilenetV2 modeli, önceden eğitilmiş ağırlıklara sahip MobileNetV2 taban modelini kullanarak oluşturulmuştur. Taban model, 224x224 boyutundaki giriş görüntülerini alır ve üst katmanları çıkartarak daha küçük boyutlu özellik haritaları oluşturur. Bu sayede daha hafif bir model elde edilir. MobilenetV2 taban modeli, başka bir görev için önceden eğitildiği için, bu ağırlıkları dondurarak ve eğitilebilirliklerini devre dışı bırakarak transfer öğrenme yaklaşımını benimseriz. Bunu yaparak, mevcut öğrenilmiş özelliklere dayanarak kendi veri setimize özgü sınıflandırma katmanını ekleyebiliriz. Global Average Pooling katmanı, çıktı özellik haritalarını vektörel bir temsil haline getirir ve yoğunluk (dense) katmanı, sınıflandırma için gerekli çıktıyı sağlar. Modeli derledikten sonra, belirli bir öğrenme hızıyla eğitim yaparız. Bu çalışmada, 'adam' optimize edici ve ikili çapraz entropi kaybı kullanarak eğitim gerçekleştirilmiştir. Sonuç olarak, MobilenetV2 modeli, yüksek doğruluk ve verimlilik sağlayan önceden eğitilmiş bir modeli transfer öğrenme ile kullanarak sınıflandırma görevini gerçekleştirmektedir.

Algoritma 2’de MobilenetV2 modelinin kodları verilmiştir.

|  |
| --- |
| Algoritma 2. MobilenetV2 modelinin kodları |
| # *Kütüphanelerin import edilmesi* |
| from tensorflow.keras.applications import MobileNetV2 |
| from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout |
|  |
| *# Modelin oluşturulması* |
| base\_model = tf.keras.applications.MobileNetV2(input\_shape=(224,224,3), |
| include\_top=False, |
| weights='imagenet') |
| base\_model.trainable = False |
| global\_average\_layer = tf.keras.layers.GlobalAveragePooling2D() |
| prediction\_layer = tf.keras.layers.Dense(1) |
|  |
| mobilenetv2 = tf.keras.Sequential([ |
| base\_model, |
| global\_average\_layer, |
| prediction\_layer |
| ]) |
|  |
| base\_learning\_rate = 0.0001 |
| mobilenetv2.compile(optimizer=tf.keras.optimizers.RMSprop(learning\_rate=base\_learning\_rate), |
| loss=tf.keras.losses.BinaryCrossentropy(from\_logits=True), |
| metrics=['accuracy']) |
|  |
| # *Modelin derlenmesi* |
| mobilenetv2.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy']) |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

mobilenetv2\_1.00\_224 (Funct (None, 7, 7, 1280) 2257984

ional)

global\_average\_pooling2d\_2 (None, 1280) 0

(GlobalAveragePooling2D)

dense (Dense) (None, 1) 1281

=================================================================

Total params: 2,259,265

Trainable params: 1,281

Non-trainable params: 2,257,984

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Şekil 7.** MobilenetV2 modelinin özeti

### 2.6.2.1. MobilenetV2 modelinin doğruluk ve kayıp eğrileri

|  |  |
| --- | --- |
| C:\Users\user\fake_face_env\Scripts\mobilenetv2 acc.png | C:\Users\user\fake_face_env\Scripts\mobilenetv2 loss.png |
| a) Modelin eğitim ve doğrulama doğruluk eğrileri | b) Modelin eğitim ve doğrulama kayıp eğrileri |

**Şekil 8.** MobilenetV2 modelinin eğitim ve doğruluk eğrileri

### 2.6.2.2. MobilenetV2 modelinin sınıflandırma başarımları ve karmaşıklık matrisi

Bu bölümde, modelin sınıflandırma performansını değerlendirmek için kullanılan karmaşıklık matrisi ve elde edilen sınıflandırma başarımları sunulmuştur. Sınıflandırma başarımlarını değerlendirmek için Denklem (1), Denklem (2), Denklem (3) ve Denklem (4) kullanılmıştır.

**Tablo 4.** MobilenetV2 modelinin karmaşıklık matrisi

|  |  |  |
| --- | --- | --- |
|  | **Doğru** | **Yanlış** |
| **Pozitif** | 5753 | 219 |
| **Negatif** | 737 | 691 |

**Tablo 5.** MobilenetV2 modelinin sınıflandırma başarımları

|  |  |  |  |
| --- | --- | --- | --- |
| **Kesinlik (%)** | **Hassasiyet (%)** | **Doğruluk (%)** | **F1 Skoru (%)** |
| 96.33 | 89.27 | 87.70 | 94.18 |

### 2.6.3. Densenet201 modeli

Densenet201, görüntü sınıflandırma görevleri için kullanılan bir derin öğrenme modelidir. Bu model, TensorFlow Keras kütüphanesinin DenseNet201 sınıfı kullanılarak oluşturulmuştur. Model, 224x224 boyutundaki giriş görüntülerini kabul eder. Önceden eğitilmiş imagenet ağırlıkları kullanılarak inşa edilmiştir. Modelin temel bileşenleri, evrişimli sinir ağı (CNN) taban modeli, global average pooling katmanı ve yoğunluk (dense) katmanıdır.

Densenet201 modelinin özelliği, yoğun bağlantıları içeren bir yapıya sahip olmasıdır. Bu bağlantılar, her katmanın kendisinden önceki tüm katmanlardan doğrudan bilgi almasını sağlar. Bu sayede modelin bilgi akışı daha etkili ve gradyanların daha iyi aktarılması sağlanır.

Modelin önceden eğitilmiş ağırlıkları kullanılarak, transfer öğrenme yaklaşımı benimsenmiştir. Taban modelin eğitilebilirliği devre dışı bırakılmıştır, yani bu ağırlıklar güncellenmez. Sadece global average pooling katmanı ve yoğunluk katmanı eğitim sırasında öğrenilir.

Modelin eğitim süreci için RMSprop optimize edici kullanılmıştır. Kayıp fonksiyonu olarak ikili çapraz entropi kullanılmış ve doğruluk metriği değerlendirilmiştir. Bu yapılandırmalar, modelin iyi bir sınıflandırma performansı sergilemesini sağlar. Densenet201 modeli, toplamda 18,323,905 parametreye sahiptir. Bununla birlikte, eğitilebilir parametre sayısı sadece 1,921'dir. Bu durum, önceden eğitilmiş ağırlıkların büyük bir kısmının kullanıldığını ve sadece son katmanların eğitim sırasında öğrenildiğini gösterir. Bu şekilde, Densenet201 modeli kullanılarak yüksek performanslı görüntü sınıflandırma görevleri gerçekleştirilebilir.

Algoritma 3’te Densenet201 modelinin kodları verilmiştir.

|  |
| --- |
| Algoritma 3. Densenet201 modelinin kodları |
| *#Kütüphanelerin import edilmesi* |
| from tensorflow.keras.applications import DenseNet201 |
| from tensorflow.keras.layers.experimental.preprocessing import Rescaling |
| from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout |
|  |
| *#Modelin oluşturulması* |
| base\_model = tf.keras.applications.DenseNet201(input\_shape=(224,224,3), |
| include\_top=False, |
| weights='imagenet') |
| base\_model.trainable = False |
| global\_average\_layer = tf.keras.layers.GlobalAveragePooling2D() |
| prediction\_layer = tf.keras.layers.Dense(1) |
|  |
|  |
| densenet201 = tf.keras.Sequential([ |
| base\_model, |
| global\_average\_layer, |
| prediction\_layer |
| ]) |
|  |
| base\_learning\_rate = 0.0001 |
| densenet201.compile(optimizer=tf.keras.optimizers.RMSprop(learning\_rate=base\_learning\_rate), |
| loss=tf.keras.losses.BinaryCrossentropy(from\_logits=True), |
| metrics=['accuracy']) |
|  |
| *#Modelin derlenmesi* |
| model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy']) |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

densenet201 (Functional) (None, 7, 7, 1920) 18321984

global\_average\_pooling2d (G (None, 1920) 0

lobalAveragePooling2D)

dense (Dense) (None, 1) 1921

=================================================================

Total params: 18,323,905

Trainable params: 1,921

Non-trainable params: 18,321,984

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Şekil 9.** Densenet201 modelinin özeti

### 2.6.3.1. Densenet201 modelinin doğruluk ve kayıp eğrileri

|  |  |
| --- | --- |
| C:\Users\user\fake_face_env\Scripts\densenet acc.png | C:\Users\user\fake_face_env\Scripts\densenet loss.png |
| a) Modelin eğitim ve doğrulama doğruluk eğrileri | b) Modelin eğitim ve doğrulama kayıp eğrileri |

**Şekil 10.** Densenet201 modelinin eğitim ve doğruluk eğrileri

Şekil 10, eğitim ve doğrulama sırasında ağın öğrenme eğrilerini göstermektedir. Şekil 10.(a)’da gözlemlendiği üzere genel olarak doğrulama doğruluğu ve eğitim doğruluğu düzenli bir artış göstermiş ve %87.13 eğitim doğruluğu ve %86.73 doğrulama doğruluğuna ulaşmıştır. Şekil 10.(b)’de ise ağın eğitim ve doğrulama kaybının düzenli olarak düştüğü görülmektedir.

### 2.6.3.2. Densenet201 modelinin sınıflandırma başarımları ve karmaşıklık matrisleri

Bu bölümde, modelin sınıflandırma performansını değerlendirmek için kullanılan karmaşıklık matrisi ve elde edilen sınıflandırma başarımları sunulmuştur. Sınıflandırma başarımlarını değerlendirmek için Denklem (1), Denklem (2), Denklem (3) ve Denklem (4) kullanılmıştır.

**Tablo 6.** Densenet201 modelinin karmaşıklık matrisi

|  |  |  |
| --- | --- | --- |
|  | **Doğru** | **Yanlış** |
| **Pozitif** | 5766 | 206 |
| **Negatif** | 741 | 687 |

**Tablo 7.** Densenet201 modelinin sınıflandırma başarımları

|  |  |  |  |
| --- | --- | --- | --- |
| **Kesinlik (%)** | **Hassasiyet (%)** | **Doğruluk (%)** | **F1 Skoru** |
| 96.5 | 88.6 | 87.2 | 92.3 |

# 3. Sonuç

Bu çalışmanın sonuçlarına genel olarak bakıldığında, kullanılan veri seti ve eğitim yöntemleri ile oluşturulan modellerin başarılı bir performans sergilediği söylenebilir. Çalışmada kullanılan özgün CNN modeli, MobilenetV2 ve Densenet201 modelleri, veri seti üzerinde eğitilmiştir. Bu modellerin sınıflandırma performansları doğrulama ve kayıp eğrileri, karmaşıklık matrisleri ve sınıflandırma başarımları üzerinden değerlendirilmiştir. Sonuçlara dayanarak, özgün CNN modelinin diğer modellere göre daha yüksek bir performans sergilediği söylenebilir. Ancak, MobilenetV2 ve Densenet201 modellerinin de genel olarak iyi bir sınıflandırma performansı gösterdiği görülmektedir.

Bu çalışmanın sonuçları, sahte ve gerçek insan yüzlerini tespit etmek için kullanılan modelin etkili bir şekilde çalışabileceğini ve gerçek hayattaki senaryolara uyum sağlayabileceğini göstermektedir. Bu tür çalışmalar, sahte görüntüleri algılama ve doğrulama konularında ilerlemeye yönelik önemli adımlar oluşturabilir ve güvenlik, dolandırıcılık tespiti ve benzeri alanlarda kullanışlı olabilir. Ancak, daha fazla veri seti ve farklı model denemeleriyle yapılan ileri çalışmaların yapılması da önemlidir.

# 4. Kaynakça

1. <https://www.geeksforgeeks.org/python-data-augmentation/>

2. <https://keras.io/api/applications/mobilenet/>

3. <https://medium.com/nerd-for-tech/image-classification-using-transfer-learning-vgg-16-2dc2221be34c>

4. <https://www.analyticsvidhya.com/blog/2020/09/overfitting-in-cnn-show-to-treat-overfitting-in-convolutional-neural-networks/>

5. <https://philarchive.org/archive/SALCOR-3>

6. <https://www.youtube.com/watch?v=jztwpsIzEGc&t=389s>

7. <https://www.kaggle.com/datasets/ciplab/real-and-fake-face-detection/code?resource=download>

8. <https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/>

9. <https://blog.roboflow.com/how-to-train-mobilenetv2-on-a-custom-dataset/>

10. <https://www.yesilscience.com/transfer-learning-densenet201/>

11. <https://keras.io/guides/preprocessing_layers/#image-data-augmentation>

12. <https://www.datacamp.com/>

13. [https://www.tensorflow.org/tutorials/](https://www.tensorflow.org/tutorials/images/data_augmentation?hl=tr#apply_augmentation_to_a_dataset)