

**KARADENİZ TEKNİK ÜNİVERSİTESİ**

**YAZILIM MÜHENDİSLİĞİ**

**Görüntü İşleme**

**Arasınav Projesi**

**Hazırlayanlar**

**397242 – Duygu ARDA**

**397234 – Gizem DEMİRBİLEK**

**397239 – Edanur KOCAKOÇ**

**İÇİNDEKİLER**

1. Giriş…………………………………………………………………………………………

2. Kullanılan Metaryeller……………………………………………………………………...

2.1 Kaggle……………………………………………………………………………………………

2.2 CNN (Convolutional Neural Networks)…………………………………………………………

2.3 Visual Studio Code………………………………………………………………………………

3. Gerçekleştirim……………………………………………………………………………...

3.1. Veri Sınıflandırma ve Veri Bölümleme…………………………………………………………

3.2. Sabit Değerleri Belirleme……………………………………………………………………….

3.3 CNN Oluşturulması…………………………………………………………………………..….

3.4. Modelin Kullanılması…………………………………………………………………….……..

4. Test…………………………………………………………………………………...…….

5. Sonuçlar…………………………………………………………………………………….

5.1 Proje Çıktısı

5.2 Sınıflandırma başarımları ve Loss grafikleri…………………………………………….,…….

5.3 Acc………………………………………………………………………………….. ……….…

5.4 Karmaşıklık matrisleri……………………………………………………………… ………..…

6. Kaynakça………………………………………………………………………………..….

**1. Giriş**

Fake face detection projesinin amacı, sahte yüzlerin otomatik olarak tespit edilmesini sağlamaktır. Günümüzde, görüntü işleme ve yapay zeka teknikleriyle üretilen sahte yüzlerin sayısı giderek artmaktadır. Bu sahte yüzler, sosyal medyada veya diğer platformlarda kullanıcıları yanıltmak, dolandırıcılık amacıyla kullanılmak veya kişilik hırsızlığı gibi kötü niyetli amaçlar için kullanılmaktadır.

Bu proje, sahte yüzlerin tespit edilmesinde derin öğrenme algoritmalarından biri olan Evrişimli Sinir Ağları (CNN) kullanarak bir çözüm sunmaktadır. CNN, görüntülerdeki karmaşık örüntüleri öğrenmek ve analiz etmek için kullanılan güçlü bir derin öğrenme modelidir.

Proje, Kaggle gibi bir kaynakta bulunan hazır bir veri seti kullanılarak geliştirilmiştir. Bu veri seti, sahte yüzlerle gerçek yüzlerin bir arada bulunduğu etiketlenmiş görüntüler içermektedir. CNN modeli, bu veri setini eğiterek sahte yüzlerle gerçek yüzler arasındaki farklılıkları öğrenir ve sahte yüzleri doğru bir şekilde tanımlayabilmeyi amaçlar.

Projenin amacı, güvenilirlik ve güvenlik açısından önemli olan sahte yüzlerin tespit edilmesine katkıda bulunmaktır. Bu, sosyal medya platformları, güvenlik sistemleri ve diğer uygulamalarda kullanıcıları sahte yüzlerin potansiyel risklerine karşı koruma sağlamak için kullanılabilir. Ayrıca, ileriye dönük olarak bu alanda yapılacak çalışmalara ve sahte yüzlerin gelişen tekniklerine karşı daha güçlü tespit yöntemleri geliştirmeye bir adım olarak da değerlendirilebilir.

Fake face detection projesi, görüntü işleme, derin öğrenme ve yapay zekâ alanlarının birleşimini kullanarak sahte yüzlerin tespit edilmesi konusunda önemli bir katkı sağlar.

**2. Kullanılan Mataryeller**

**2.1. Kaggle**

Kaggle, veri bilimciler ve makine öğrenimi ile ilgilenenler için çevrimiçi bir platformdur. Veri bilimcileri ve makine öğrenimine ilgi duyan kişileri bir araya getiren büyük bir veri kümesi paylaşımı bulunan topluluktur.

**2.2. CNN (Convolutional Neural Networks)**

Zamanlar gelişim hızını daha da artırarak büyüyen yapay zekâ sayesinde makinelerin işlevleri daha da artmakta. NLP (Natural Language Processing) ile insan dilini, Speech Recognition (Konuşma tanıma) ile sesleri tanımlayabilen makineler bilgisayarlı görme sayesinde insanlar gibi görerek nesneleri, yüzleri, hareketleri algılayabilmektedir. Resimleri yüksek doğruluk oranları ile algılayabilen, sınıflandırabilen, yeniden yapılandırabilen derin öğrenmeli bilgisayarlı görme modellerinin vazgeçilmezi olan algoritma, CNN (Convolutional Neural Networks), bir diğer adıyla evrişimli sinir ağıdır. gibi görsel verilerle çalışırken etkili bir şekilde kullanılır.

CNN'ler, diğer yapay sinir ağı türlerinden farklı olarak özellik çıkarımını otomatik olarak gerçekleştirme yeteneğine sahiptir. CNN'lerin temel amacı, giriş olarak aldıkları verileri (genellikle görüntüler) analiz ederek çeşitli özellikleri tanımlamak ve sınıflandırmaktır. Bu özellikler, filtreler veya çekirdekler olarak bilinen küçük boyutlu matrislerle uygulanan evrişim işlemleriyle çıkarılır. Ardından, elde edilen özellik haritaları, havuzlama (pooling) adı verilen bir işlemle boyutları küçültülerek özetlenir.

**2.3. Visual Studio Code**

Projenin gerçekleştirilmesi için kullanılan, Microsoft tarafından geliştirilen kod düzenleyicisi kullanıldı. Dil olarak Python ve kütüphaneleri kullanılmıştır.

**3. Gerçekleştirim**

**3.1. Veri Sınıflandırma ve Veri Bölümleme**

Kaggle üzerinden fake face ve reel face veri seti indirildi. Bu veri seti 1081 fake, 960 reel face içeriyor. Bu veriler “fake” ve “reel” sınıf olarak data dizisine atandı. Bu sınıflar reel = 1, fake = 0 olacak şekilde label dizisine atanarak binary sınıflandırma yapıldı.

transforms.Resize((128, 128)) ile (128, 128) boyutları verilmiştir, yani görüntüler 128x128 piksel boyutuna getirilir. Bu, görüntünün boyutunu modele uygun hale getirmek için kullanılır.

transforms.ToTensor() ile görüntüyü PyTorch tensor formatına dönüştürür. PyTorch'ta modeller, veri olarak tensorları kullanır. Bu dönüşüm, görüntüyü kanallara (RGB) ayrıştırır ve piksel değerlerini 0 ile 1 arasında normalize eder.

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]), bu dönüşüm, görüntünün piksel değerlerini normalize eder. Normalleştirme, görüntünün ortalamasını ve standart sapmasını kullanarak her pikselin değerini belirli bir dağılım ile değiştirir. Bu, modele daha iyi öğrenme sağlayabilir. Bu özel normalize değerleri (mean ve std), ImageNet veri kümesi üzerinde eğitilen popüler bir görüntü sınıflandırma modeli olan ResNet'in normalleştirme parametreleridir.

Veri bölümleme ile mevcut veri setini iki veya daha fazla alt kümeye ayırarak bir eğitim seti ve bir doğrulama (veya test) seti oluşturmayı sağlar. Eğitim seti, modelin eğitiminde kullanılan veri kümesidir ve modelin parametrelerini öğrenmek için kullanılır. Doğrulama (veya test) seti ise, eğitim süreci boyunca modelin performansını değerlendirmek ve genellemesini kontrol etmek için kullanılır. Böylelikle %80 eğitim, %20 test veri kümesi şeklinde veriler ayrıldı.

insan yüzü, kolaj, kadın, fotomontaj içeren bir resim

Açıklama otomatik olarak oluşturuldu

Eğitimde kullanılan fake face verileri

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

Açıklama otomatik olarak oluşturuldu

Eğitimde kullanılan real face verileri

**3.2. Sabit Değerleri Belirleme**

batch\_size = 64, veri yükleyicilerinde kullanılan mini toplu boyutudur. Eğitim ve doğrulama için kullanılan train\_dataLoader ve valid\_loader veri yükleyicileri için 64 olarak belirlenmiştir.

Epoch sayısı 15 olarak belirlendi. Veri setinin boyutu, epoch sayısının belirlenmesinde etkili olduğundan küçük veri setleri için daha az epoch yeterlidir. Bu yüzden 15 epoch, veri seti boyutuna uygun bir değer olarak seçildi.

Convolutional Layer = 4 adettir, her biri 3 kanal giriş, 16 kanal çıkış, 3x3 kernel boyutu içerir.

Linear Layer = 2 adettir, giriş özellikleri olarak 32x6x6 boyutunu alır ve 128 çıkış özellikleri üretir.

lr = Optimizasyon algoritması olan Adam optimizasyonunun öğrenme oranı 0.001 olarak belirlenmiştir.

weight\_decay = Adam optimizasyonunda kullanılan ağırlık düzenlemesi (weight decay) değeri 0.0001 olarak belirlenmiştir. Bu değer, ağırlıkların büyüklüklerini azaltarak aşırı uymayı (overfitting) önlemeye yardımcı olur.

**3.3. CNN Oluşturulması**

Net() sınıfı kullanılarak bir yapay sinir ağı modeli oluşturulur. Net, özelleştirilmiş bir model sınıfının örneğini temsil eder. Bu sınıf, genellikle PyTorch'ün nn.Module sınıfından türetilir ve modelin yapısını ve ileri yayılım işlemini tanımlar.

CNN modelini oluşturmak için gerekli convolution katmanları eklendi. Genellikle, evrişimli (convolutional) katmanlar, aktivasyon fonksiyonları, düşürme (dropout) katmanları ve tam bağlantı (fully connected) katmanları gibi farklı katman türlerini kullanabilir.

CNN, özellik çıkarım süreci için evrişimli (convolutional) katmanları kullanır. Evrişimli katmanlar, girdi görüntü üzerindeki özellikleri yakalamak için kullanılan filtrelerden oluşur. Bu filtreler, çeşitli özellikleri algılamak için belirli özellikler üzerinde yoğunlaşır.

Evrişimli katmanlar, her biri belirli bir boyutta ve derinlikte filtreler içeren matrislerdir. Bu filtreler, girdi görüntünün farklı bölgelerinde gezinerek, her konumda bir evrişim işlemi gerçekleştirir. Evrişim işlemi, filtre matrisinin, görüntünün ilgili bölgesiyle eleman bazlı çarpımını alıp sonuçları toplamlayarak yeni bir çıktı matrisi oluşturur.

Python içindeki cnn kütüphanesini kullanarak aşağıdaki filtreler ile otomatik olarak özellik çıkarımı yapılmış olundu.

1. Kenar Algılayıcı Filtreler: Kenarlar, yüzün farklı bölgelerinin sınırlarını belirleyen önemli özelliklerdir. Kenar algılayıcı filtreler, görüntünün farklı yönlerindeki kenarları yakalamak için kullanılır. Örneğin, Sobel veya Prewitt filtreleri, dikey veya yatay kenarları tespit etmek için yaygın olarak kullanılır.
2. Gabor Filtreler: Gabor filtreleri, yüzdeki farklı frekans ve yönlere sahip özellikleri algılamak için kullanılır. Bu filtreler, yüzdeki dokusal ve yapısal özellikleri vurgular. Gabor filtreleri, özellikle gözlerin veya ağzın gibi yüzdeki belirli bölgelerin tanınmasında etkilidir.
3. HOG (Histogram of Oriented Gradients) Filtreleri: HOG filtreleri, görüntüdeki kenarların yoğunluk ve yönelim bilgisini kullanarak özellik çıkarır. Bu filtreler, yüzdeki belirli bölgelerin şeklini ve konturunu yakalamak için kullanılabilir. Yüzdeki gözler veya ağız gibi özelliklerin tespitinde sıklıkla kullanılırlar.
4. Haar-like Filtreler: Haar-like filtreler, görüntüdeki yüz bölgelerindeki parlaklık değişimlerini kullanarak özellikleri çıkarır. Bu filtreler, yüzün belirli bölgelerindeki yatay, dikey veya çapraz desenlerin algılanmasında etkilidir. Örneğin, gözlerin veya ağzın olduğu bölgelerdeki parlaklık değişimlerini vurgulamak için kullanılabilirler.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Cnn modeli

**3.4. Modelin Kullanılması**

Adam, forward, train ve test fonksiyonları oluşturularak Cnn modeli kullanılmaya amaçlandı.

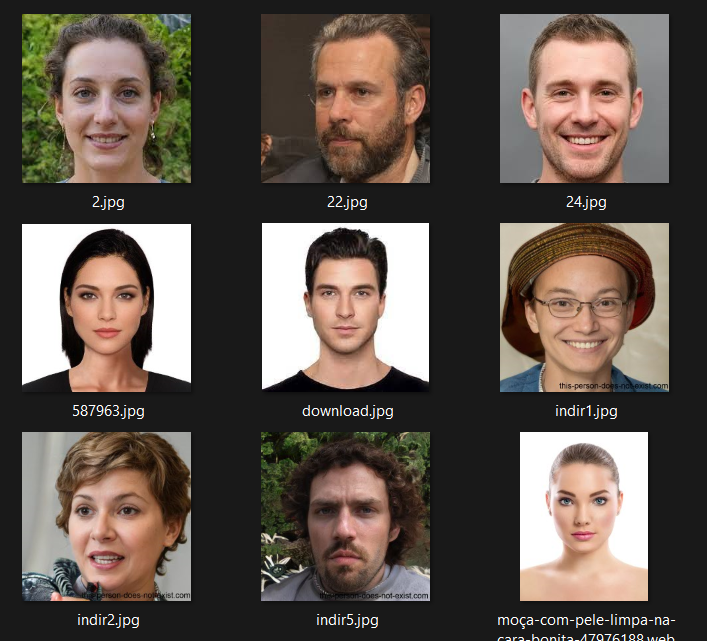
* Adam, gradyan tabanlı bir optimizasyon algoritmasıdır ve modelin ağırlıklarını güncellemek için kullanıldı.
* Forward fonksiyonu ile modelin ileri yayımlı eğitim sürecini gerçekleştirir. Her bir döngüde, modelin parametreleri güncellenir ve eğitim kaybı ile doğrulama kaybı ve doğruluk değerleri hesaplanır. Bu değerlerin izlenmesi, modelin performansını değerlendirmek ve aşırı uyma (overfitting) durumunu kontrol etmek için kullanıldı.
* Train, eğitim süreci için gerekli olan fonksiyon döngüsüdür.
* Test, eğitimde kullanılmayan görseller üzerinden sonucun öğrenilmesi için kullanılacak fonksiyondur.
* Torc fonksiyonu modelin test veri kümesi üzerindeki performansını değerlendirmek için kullanılır. Kaybı ve doğruluğu hesaplar, tahminleri ve gerçek etiketleri saklar ve bu değerleri raporlamak için kullanıldı.
* Plot fonksiyonu ile training loss, training acc, validation loss, validation acc değerleri grafik olarak çıktı almak için kullanıldı.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**4. Test**

Oluşturulan Test fonksiyonu ile model test edilerek istenen grafiklerin çıkışı için plot kütüphanesinden yararlanılarak çizdirme kodları yazılır. Test verilerinde önceden kullanılmayan veriler kullanıldı.

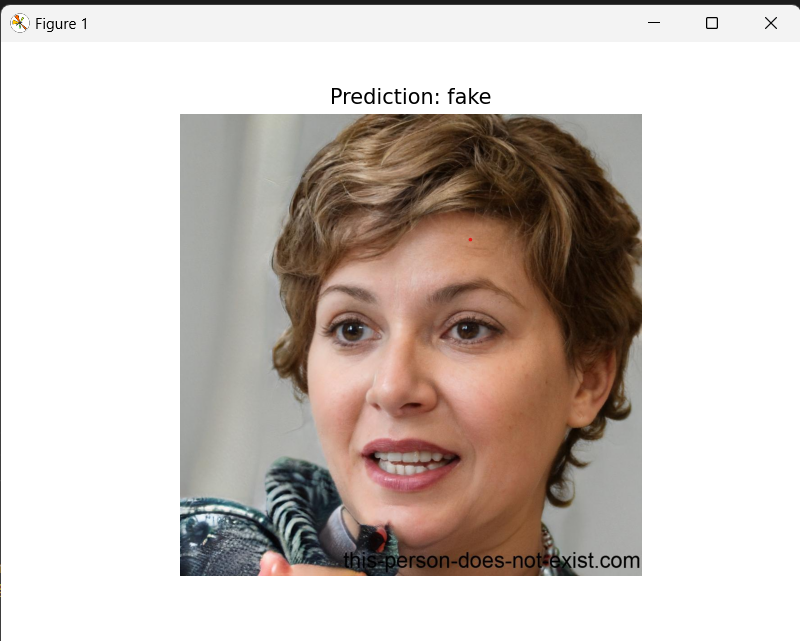


Test Verileri

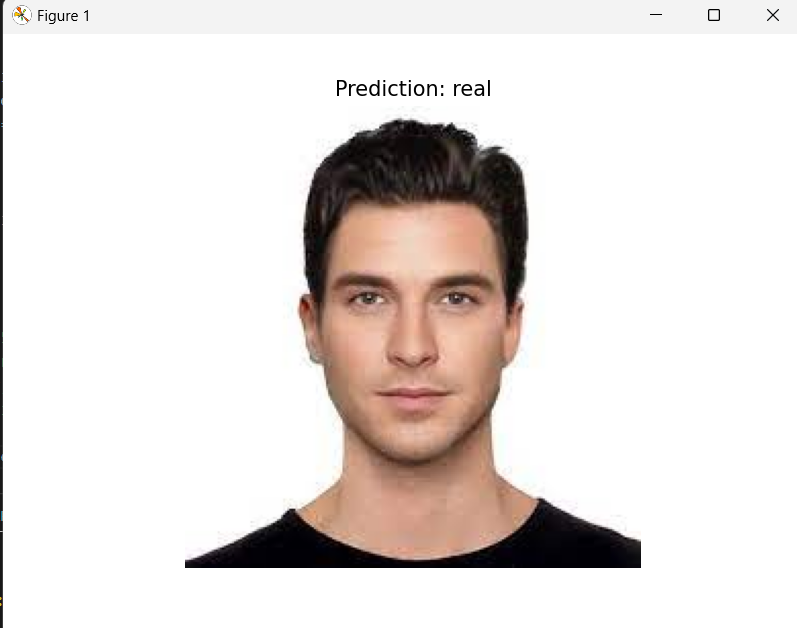
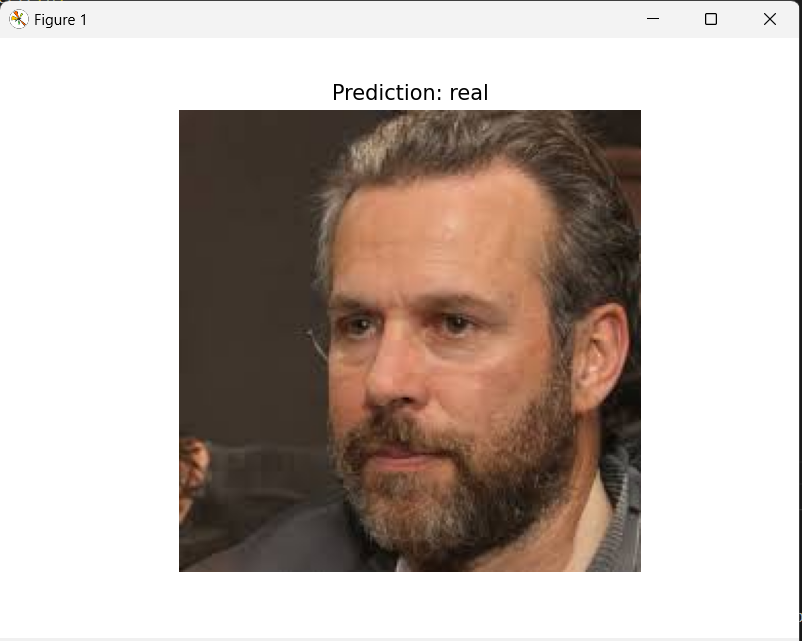
**5. Sonuç**

**5.1. Proje Çıktısı**

Yüzde 60 oranında bir başarı oranına sahip olundu. Bazı test verileri ile sonuçlar aşağıdaki gibidir.



Çıktı 1



Çıktı 2

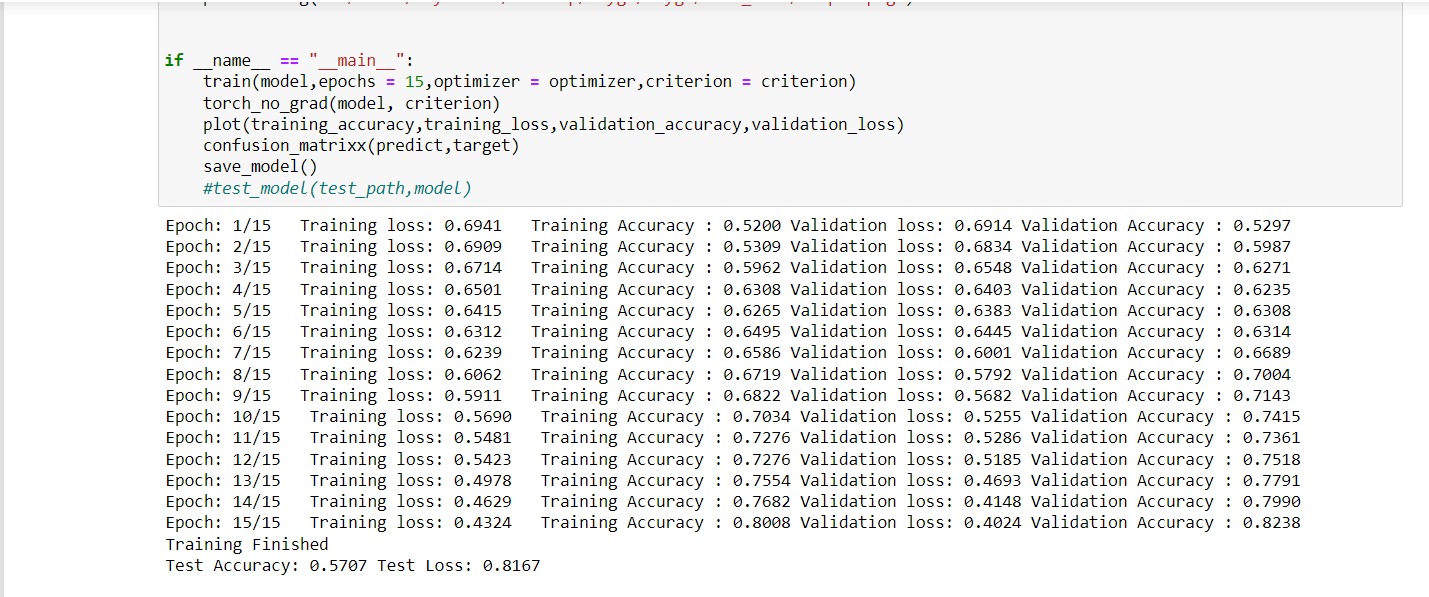
**5.2. Sınıflandırma başarımları ve Loss grafikleri**

**metin, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**5.3. Acc**

Her 9 resimde 6 resim doğru sonuçlanmaktadır. Başarı yüzdesi %60 ve üstüdür.



**5.4. Karmaşıklık matrisleri**

Confusion matrix (karmaşıklık matrisi), bir sınıflandırma modelinin performansını değerlendirmek için kullanılan bir matrisdir. Sınıflandırma problemlerinde, modelin tahminleri ve gerçek sınıflar arasındaki ilişkiyi görselleştirmek için kullanılır.

Real sınıfını tahmin etmek için kullanılan modelin doğru tahmin oranı 0.59 ise, gerçek verilerin %59'unun doğru bir şekilde sınıflandırıldığını söylenir. Aynı şekilde, fake sınıfını tahmin etmek için kullanılan modelin doğru tahmin oranı 0.63 ise, fake verilerin %63'ünün doğru bir şekilde sınıflandırıldığı söylenir.

Diğer değerler ise yanlış tahminlerin oranlarını temsil eder. Fake verileri real olarak tahmin etme oranı 0.37 ise, fake verilerin %37'sinin yanlış bir şekilde real olarak sınıflandırıldığını söylenebilir. Bu projede real verileri fake olarak tahmin etme oranı 0.41 ise, real verilerin %41'inin yanlış bir şekilde fake olarak sınıflandırıldığı söylenebilir.

**ekran görüntüsü, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**6. Kaynakça**

[**https://www.kaggle.com/**](https://www.kaggle.com/)

[**https://www.kaggle.com/datasets/ciplab/real-and-fake-face-detection**](https://www.kaggle.com/datasets/ciplab/real-and-fake-face-detection)

[**https://atcold.github.io/pytorch-Deep-Learning/tr/week11/11-1/**](https://atcold.github.io/pytorch-Deep-Learning/tr/week11/11-1/)