**Proje Adı:** Create an age estimation network using the Faces dataset.

**İsim ve Soyad:** Yağmur Çağla Öztürk

**Son Teslim Tarihi:** 20 Mayıs 2024

**ÖZET**

Yaş tahmini, bir kişinin yüz görüntüsünden yaşını tahmin etmeyi amaçlayan bir bilgisayar görüşü problemidir. Bu proje, UTKFace veri seti kullanarak yaş tahmini yapabilen derin öğrenme tabanlı bir ağ geliştirmeyi hedeflemektedir. Kullanılan veri setinde 23678 adet fotoğraf bulunmaktadır. Ayrıca yaş aralığı 0-116 olarak değişmektedir. Projenin ana aşamaları veri hazırlama, model geliştirme, eğitim ve değerlendirme süreçlerinden oluşmaktadır.

**Veri Hazırlama:**

* **Veri Toplama:** Faces veri seti, yaş tahmini için uygun olan yüz görüntülerini içermektedir. Veri seti, farklı yaş gruplarından çeşitli yüz görüntüleri içermektedir.
* **Ön İşleme:** Görüntüler, modelin ihtiyaçlarına göre ölçeklendirilir ve normalleştirilir.
* **Etiketleme:** Görüntülere karşılık gelen yaş etiketleri belirlenir ve bu etiketler modelin öğrenmesi için kullanılır.

**Model Geliştirme:**

* **Model Seçimi:** Yaş tahmini için Convolutional Neural Network (CNN) tabanlı bir model seçilir. CNN'ler, görüntü işleme görevlerinde yüksek performans göstermektedir.
* **Ağ Mimarisi:** Ağ mimarisi, bir dizi konvolüsyonel katman, pooling katmanları ve tam bağlantılı katmanlardan oluşur. Bu katmanlar, yüz görüntülerinden özellik çıkarımı ve yaş tahmini yapmak için optimize edilir.

**Eğitim:**

* **Eğitim Süreci:** Model, Faces veri seti üzerinde eğitilir. Eğitim sırasında, veri seti eğitim ve doğrulama setlerine ayrılır. Bu sayede modelin genelleme kabiliyeti değerlendirilir.
* **Optimizasyon:** Eğitim sürecinde, optimizasyon algoritmaları (örneğin Adam, SGD) kullanılarak model parametreleri güncellenir. Öğrenme oranı ve diğer hiperparametreler, modelin performansını maksimize edecek şekilde ayarlanır.

**Değerlendirme:**

* **Doğrulama ve Test:** Model, doğrulama ve test veri setleri üzerinde değerlendirilir. Bu aşamada, modelin yaşı ne kadar doğru tahmin ettiği ölçülür.
* **Performans Metriği**: Modelin performansı, ortalama mutlak hata (Mean Absolute Error - MAE) gibi metriklerle değerlendirilir.
* **Görselleştirme:** Tahmin edilen yaşlar ve gerçek yaşlar arasındaki farklar görselleştirilir, böylece modelin performansı daha iyi anlaşılır.

**Teknik Sunum**

**Veri Hazırlama:**

* **os.listdir(BASE\_DIR):** Belirtilen dizindeki dosyaları listeler. Bu, Faces veri setindeki görüntü dosyalarının yolu ve adını içerir.
* **filename.split('\_'):** Dosya adını '\_' karakterine göre ayırır. Bu, dosya adında bulunan yaş bilgisini ayırır. Örneğin, "35\_1\_1\_20170104020230699.jpg.chip.jpg" dosyasında, '35' yaşını ayırmak için kullanılır.
* **pd.DataFrame():** Pandas kütüphanesi kullanılarak bir veri çerçevesi oluşturur. Bu çerçeve, görüntü dosyalarını ve yaş etiketlerini içerir.
* **load\_img():** Keras kütüphanesi aracılığıyla görüntüleri yükler. Görüntüler, daha sonra modele giriş olarak kullanılacak şekilde hazırlanır.
* **extract\_features():** Görüntülerden özelliklerin çıkarılmasını sağlar. Bu adım, görüntüleri işleyerek modele veri olarak verilecek uygun bir formata dönüştürür.

**Model Geliştirme:**

* **CNN katmanları:**
* Konvolüsyonel sinir ağı mimarisi kullanılarak özelliklerin çıkarılması ve modelin oluşturulması sağlanır.
* Konvolüsyonel katmanlar, görüntü üzerinde belirli desenlerin ve özelliklerin algılanmasını sağlar.
* **Yoğun (Dense) katmanlar:** Konvolüsyonel katmanlardan gelen özellikler, yoğun katmanlar aracılığıyla kullanılarak nihai tahmin yapılır.
* **Dropout:** Overfitting'i azaltmak için kullanılan bir tekniktir. Rastgele seçilen bir yüzde oranındaki nöronlar eğitim sırasında geçici olarak devre dışı bırakılır.
* **Aktivasyon fonksiyonları:** ReLU (Rectified Linear Activation) gibi aktivasyon fonksiyonları,

ağın öğrenme yeteneğini artırır.

* **MSE:** Modelin eğitimi sırasında, mean squared error (MSE) kaybı minimize edilmeye çalışılmıştır. MSE, tahmin edilen ve gerçek değerler arasındaki ortalama kare hata olup, büyük hatalara daha fazla ceza verir. Bu da modelin daha doğru tahminler yapmasını sağlar. Eğitim sürecinde, veri setinin %80'i eğitim, %20'si ise doğrulama için kullanılmıştır. Model, belirli sayıda epoch boyunca eğitilmiş ve her epoch sonunda modelin doğrulama seti üzerindeki performansı değerlendirilmiştir. Eğitim sırasında, modelin kaybı izlenmiş ve minimize edilmiştir.

**Eğitim:**

* **model.compile():** Modelin derlenmesini sağlar. Kayıp fonksiyonu ve optimizasyon algoritması belirlenir.
* **model.fit():** Modelin eğitimini gerçekleştirir. Görüntüler ve yaş etiketleri kullanılarak modelin parametreleri güncellenir. Eğitim sırasında, modelin doğruluğu ve performansı değerlendirilir.

**Değerlendirme:**

* **plot\_model():** Modelin mimarisini görselleştirir. Bu, modelin katmanlarını ve bağlantılarını gösterir.
* **model.predict():** Modelin tahmin yapmasını sağlar. Model, görüntü verileri üzerinde uygulanır ve tahmin edilen yaş değerleri elde edilir.

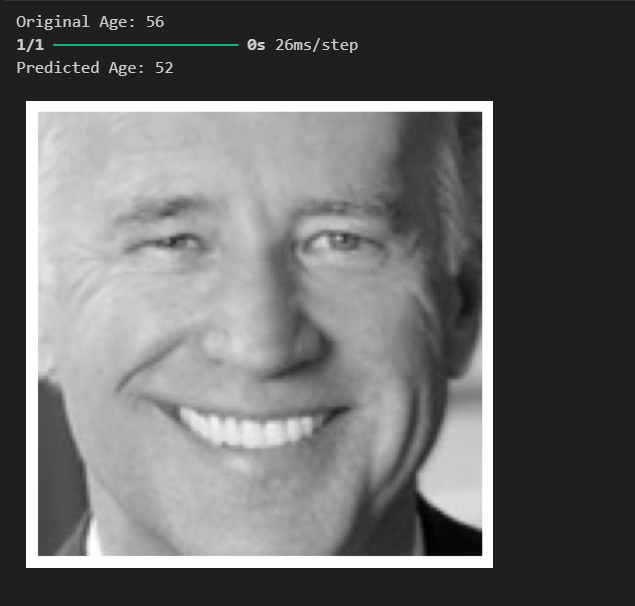
**Yaş Tahmini:**

* **CNN kullanımı:** Konvolüsyonel sinir ağı (CNN), görüntü işleme problemleri için özellikle etkilidir. Yüz tanıma gibi karmaşık görevlerde kullanılır.
* **Konvolüsyonel işlemi:** Görüntüler üzerinde konvolüsyonel filtrelerinin (kernels) kaydırılmasıyla, belirli desenlerin algılanması sağlanır.
* **Pooling işlemi:** Konvolüsyonel işlemi sonrası elde edilen özellik haritalarının boyutunu küçültür.
* **Aktivasyon fonksiyonları:** Giriş verisi üzerindeki doğrusal olmayan özellikleri çıkarmak için kullanılır.
* **Kayıp fonksiyonu:** Modelin tahminlerinin gerçek yaş etiketlerinden ne kadar farklı olduğunu ölçer ve modelin performansını değerlendirir.

**Bulguların Tartışılması**

**Çıktıların Tartışılması**

1. **Eğitim Verisi Üzerindeki Tahminler:**

* Kod, eğitim verisi üzerindeki bir örneği kullanarak yaş tahmini yapıyor. Bu örnekte, modelin tahmin ettiği yaş değeri ve gerçek yaş değeri karşılaştırılıyor. Örneğin, 7454. indeks numaralı bir görüntü için:
* **Orijinal Yaş:** 56
* **Tahmin Edilen Yaş:** 52
* Bu sonuçlar, modelin eğitim verisi üzerinde makul bir doğrulukta tahminler yapabildiğini gösterir. Ancak, modelin gerçek performansını değerlendirmek için doğrulama veya test verisi üzerinde değerlendirme yapmak daha iyi olacaktır.
* 

1. **Yeni Bir Görüntü Üzerindeki Tahminler:**

* Kod, eğitim verisinde bulunmayan yeni bir görüntü üzerinde yaş tahmini yapıyor. Örneğin, yeni bir görüntü için tahmin edilen yaş değeri:
  + **Orijinal Yaş:**23
  + **Tahmin Edilen Yaş:** 27
  + ****
* Bu tür tahminler, modelin genel tahmin kabiliyetini test etmek için önemlidir. Eğer model yeni veriler üzerinde de makul tahminler yapabiliyorsa, bu modelin iyi bir genel tahmin kapasitesine sahip olduğunu gösterir.

**Epoch Sayısının Artırılması**

Epoch sayısının artırılması, modelin tüm veri kümesi üzerinde daha fazla eğitim yapması anlamına gelir. Bu durumun birkaç etkisi olabilir:

1. **Pozitif Etkileri**

* **Daha İyi Öğrenme:** Model daha fazla eğitim verisi üzerinde çalışarak daha karmaşık desenleri ve özellikleri öğrenebilir.
* **Daha Düşük Eğitim Hatası:** Modelin eğitim hatası (loss) daha düşük seviyelere inebilir, bu da modelin eğitim verisi üzerinde daha iyi performans göstermesini sağlar.

1. **Negatif Etkiler:**

* **Aşırı Uydurma (Overfitting):** Epoch sayısının fazla olması, modelin eğitim verisine aşırı uyum sağlamasına ve doğrulama veya test verisi üzerinde kötü performans göstermesine neden olabilir. Bu durum, modelin genel tahmin kabiliyetini düşürebilir.
* **Daha Uzun Eğitim Süresi:** Epoch sayısının artması, modelin eğitim süresini uzatır, bu da daha fazla hesaplama kaynağı ve zaman gerektirir.

**Veri Sayısının Artırılması**

Veri sayısının artırılması, modelin performansını genellikle olumlu yönde etkiler:

**1.Daha İyi Tahmin:**

* **Daha Fazla Çeşitlilik:** Daha büyük veri setleri, modelin farklı örnekler üzerinde eğitim almasını sağlar. Bu da modelin daha genel geçer özellikler öğrenmesine yardımcı olur.
* **Daha Az Aşırı Uydurma:** Daha fazla veri, modelin aşırı uydurma riskini azaltır. Model, veri setinin daha geniş bir temsilini görerek daha sağlam ve genel geçer tahminler yapar.

**2.Pozitif Etkiler:**

* Daha Yüksek Doğruluk: Daha fazla veri, modelin tahmin doğruluğunu artırabilir. Model, daha fazla örnek üzerinden öğrenme yaparak daha isabetli tahminler üretebilir.
* Daha İyi Performans: Daha büyük veri setleri, modelin daha çeşitli örnekleri görmesine olanak tanır, bu da genel performansı artırır.

**Sonuç**

Modelin epoch sayısını artırmak, dikkatli bir şekilde yönetilmezse aşırı uydurma riskini artırabilir. Ancak, modelin veri seti üzerindeki daha karmaşık desenleri öğrenmesine de yardımcı olabilir. Veri sayısının artırılması ise genellikle modelin genel performansını ve doğruluğunu artırır. Daha fazla veri, modelin daha sağlam ve genel geçer tahminler yapmasını sağlar.

Bu nedenle, modelin performansını artırmak için epoch sayısını ve veri seti boyutunu dikkatli bir şekilde dengelemek gerekir. Doğrulama seti üzerinde modelin performansını izlemek, aşırı uydurma riskini azaltmak için erken durdurma (early stopping) gibi teknikleri kullanmak önemlidir. Ayrıca, veri setini artırmak için veri genişletme (data augmentation) gibi yöntemler de kullanılabilir.

**EK**

**Kodlar**

import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

from tqdm.notebook import tqdm  # tqdm, bir döngü veya işlem ilerlemesini görselleştirmek için kullanılan bir kütüphanedir.

warnings.filterwarnings('ignore')  # Uyarıları engellemek için kullanılır.

%matplotlib inline  # Jupyter defterinde çizimlerin doğrudan hücrede görüntülenmesini sağlar.

import tensorflow as tf

from keras.preprocessing.image import load\_img

from keras.models import Sequential, Model

from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D, Input

BASE\_DIR = '/kaggle/input/utkface-new/UTKFace'  # Görüntülerin bulunduğu dizin.

image\_paths = []  # Görüntü dosyalarının yollarını depolamak için boş bir liste.

age\_labels = []  # Yaş etiketlerini depolamak için boş bir liste.

# BASE\_DIR içindeki her bir dosya için:

for filename in tqdm(os.listdir(BASE\_DIR)):  # tqdm, döngünün ilerlemesini görselleştirmek için kullanılır.

    image\_path = os.path.join(BASE\_DIR, filename)  # Dosyanın tam yolunu oluştur.

    temp = filename.split('\_')  # Dosya adını '\_' karakterine göre böler.

    age = int(temp[0])  # Yaş bilgisini al.

    image\_paths.append(image\_path)  # Görüntü yolunu listeye ekle.

    age\_labels.append(age)  # Yaş etiketini listeye ekle.

# Görüntü yollarını ve yaş etiketlerini içeren bir DataFrame oluştur.

df = pd.DataFrame()

df['image'], df['age'] = image\_paths, age\_labels

df.head()

# İlk dört görüntüyü görselleştirme

from PIL import Image

img = Image.open(df['image'][3])

plt.axis('off')  # Eksenleri kapat

plt.imshow(img)

# Yaş dağılımını görselleştirme

sns.displot(df['age'])

# İlk 25 görüntüyü görselleştirme

plt.figure(figsize=(25, 25))

files = df.iloc[18:25]  # İlk 25 görüntüyü seç

for index, file, age in files.itertuples():

    plt.subplot(5, 5, index+1)

    img = load\_img(file)  # Görüntüyü yükle

    img = np.array(img)

    plt.imshow(img)

    plt.title(f"Age: {age}")

    plt.axis('off')

def extract\_features(images):

    features = []

    for image in tqdm(images):

        img = load\_img(image, color\_mode='grayscale')  # Görüntüyü gri tonlama ile yükle

        img = img.resize((128, 128), Image.ANTIALIAS)  # Görüntüyü yeniden boyutlandır

        img = np.array(img)

        features.append(img)

    features = np.array(features)

    features = features.reshape(len(features), 128, 128, 1)  # Görüntüleri uygun şekle dönüştür

    return features

X = extract\_features(df['image'])  # Görüntülerden özelliklerin çıkarılması

X.shape  # Özelliklerin şeklini görüntüleme

# Görüntüleri normalize etme

X = X/255

# Yaş verisini numpy dizisine dönüştürme

y\_age = np.array(df['age'])

input\_shape = (128, 128, 1)  # Giriş şekli tanımlama

inputs = Input((input\_shape))  # Giriş katmanı oluşturma

# Konvolüsyonel katmanlar oluşturuluyor. Bu katmanlar, özellikleri çıkarmak için görüntünün farklı ölçeklerindeki desenlere tepki verir.

# Her bir Conv2D katmanı, birbirini takip eden önceki katmanın çıktısını alır.

# Conv2D katmanlarının çıktılarını azaltmak için MaxPooling2D katmanları kullanılır.

# Bu, modelin öğrenmesini hızlandırır ve öğrenilen özelliklerin ölçeklenmesini sağlar.

conv\_1 = Conv2D(32, kernel\_size=(3, 3), activation='relu')(inputs)  # 32 filtre ile 3x3 boyutunda Konvolüsyonel katman

maxp\_1 = MaxPooling2D(pool\_size=(2, 2))(conv\_1)  # 2x2 boyutunda maksimum pooling katmanı

conv\_2 = Conv2D(64, kernel\_size=(3, 3), activation='relu')(maxp\_1)  # 64 filtre ile 3x3 boyutunda Konvolüsyonel katman

maxp\_2 = MaxPooling2D(pool\_size=(2, 2))(conv\_2)  # 2x2 boyutunda maksimum pooling katmanı

conv\_3 = Conv2D(128, kernel\_size=(3, 3), activation='relu')(maxp\_2)  # 128 filtre ile 3x3 boyutunda Konvolüsyonel katman

maxp\_3 = MaxPooling2D(pool\_size=(2, 2))(conv\_3)  # 2x2 boyutunda maksimum pooling katmanı

conv\_4 = Conv2D(256, kernel\_size=(3, 3), activation='relu')(maxp\_3)  # 256 filtre ile 3x3 boyutunda Konvolüsyonel katman

maxp\_4 = MaxPooling2D(pool\_size=(2, 2))(conv\_4)  # 2x2 boyutunda maksimum pooling katmanı

# Düzleştirme katmanı, Konvolüsyonel ve katmanlarının çıktılarını birbirine bağlar ve ağı tam bağlantılı katmanlar için uygun hale getirir.

flatten = Flatten()(maxp\_4)

# Tam bağlantılı katmanlar, ağı Konvolüsyonel katmanlardan gelen özelliklerle besler ve çıkış katmanına yönlendirir.

dense\_1 = Dense(256, activation='relu')(flatten)  # 256 nöron içeren tam bağlantılı bir gizli katman

dropout\_1 = Dropout(0.4)(dense\_1)  # Aşırı uydurmaya karşı rastgele nöron atma

dense\_2 = Dense(256, activation='relu')(dropout\_1)  # 256 nöron içeren ikinci tam bağlantılı gizli katman

dropout\_2 = Dropout(0.4)(dense\_2)  # Aşırı uydurmaya karşı rastgele nöron atma

output = Dense(1, activation='relu', name='age\_out')(dropout\_2)  # Yaş tahmini için çıkış katmanı

# Model, giriş ve çıkış katmanlarını belirterek fonksiyonel API ile oluşturuluyor.

model = Model(inputs=[inputs], outputs=[output])

# Model, ortalama karesel hata (MSE) kaybını ve Adam optimizasyonunu kullanarak derleniyor.

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['accuracy', 'mae'])

from tensorflow.keras.utils import plot\_model

# Modeli çizme

plot\_model(model, to\_file='model.png', show\_shapes=True, show\_layer\_names=True, dpi=150)

# Kaydedilen modeli görselleştirme

img = plt.imread('model.png')

plt.figure(figsize=(10, 10))

plt.imshow(img)

plt.axis('off')

plt.show()

# Modeli eğitme işlemi başlıyor.

# x=X: Modelin giriş verisi, yani görüntülerden oluşan özellikler matrisi.

# y=[y\_age]: Modelin çıkış verisi, yani yaş etiketleri dizisi. Çünkü çıkış katmanı sadece tek bir çıkışa sahip.

# batch\_size=32: Her eğitim adımında kullanılacak örnek sayısı. Model, ağırlık güncellemelerini bu örnekler üzerinde yapar.

# epochs=10: Tüm veri kümesinin kaç kez model tarafından kullanılacağını belirtir. Yani, modelin 10 tam eğitim dönemi boyunca veriler üzerinde eğitilmesini sağlar.

# validation\_split=0.2: Eğitim sırasında doğrulama için ayrılacak veri oranı. Bu oran, eğitim verisinin yüzde kaçının doğrulama için kullanılacağını belirler.

# Bu kod satırı modelin eğitimini başlatır ve eğitim süreci boyunca kaydedilen eğitim ve doğrulama kayıplarını içeren bir geçmiş nesnesi döndürür.

history = model.fit(x=X, y=[y\_age], batch\_size=32, epochs=10, validation\_split=0.2)

image\_index = 128  # Tahmin yapılacak görüntünün indeksi

print("Original Age:", y\_age[image\_index])  # Orijinal yaş değerini yazdırma

# Modelden yaş tahmini yapma

pred = model.predict(X[image\_index].reshape(1, 128, 128, 1))

pred\_age = round(pred[0][0])  # Tahmin edilen yaş değerini yuvarlama

print("Predicted Age:", pred\_age)  # Tahmin edilen yaş değerini yazdırma

plt.axis('off')  # Eksenleri kapatma

plt.imshow(X[image\_index].reshape(128, 128), cmap='gray')  # Gri tonlamalı görüntüyü görselleştirme

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

# Veri setinin dosya yolu

dataset\_path = '/kaggle/input/testfoto'

# Veri setini yükleyip işleyeceğimiz özel bir yöntem belirleme (ör. dosya adlarını listeleyerek veya başka bir yöntem)

# Burada, dosya adlarını listeleyerek bir yöntem kullanıyoruz

# Dosya yollarını bir liste olarak alabilir ve bu yolları kullanarak görüntüleri yükleyebilirsiniz

# Bu, veri setinin formatına ve dosya yapısına bağlı olarak değişebilir

# Örneğin, dataset\_path'teki tüm görüntü dosyalarını yükleme

# Ardından, belirli bir görüntü indeksi seçip yaş tahmini yapacağız

# Öncelikle, dataset\_path'teki tüm dosyaların listesini alalım

import os

image\_files = os.listdir(dataset\_path)

# Görüntü dosyalarından birini seçelim (örneğin, ilk dosyayı seçelim)

image\_path = os.path.join(dataset\_path, image\_files[0])

# Seçili görüntüyü yükleme ve boyutlandırma

image = load\_img(image\_path, target\_size=(128, 128), color\_mode='grayscale')  # Gri tonlamalı resim kullanılıyor

image = img\_to\_array(image)

# Görüntüyü ölçeklendirme

image = image / 255.0

# Modelin giriş şekline uygun hale getirme

image = np.expand\_dims(image, axis=0)

# Modelle yaş tahmini yapma

pred = model.predict(image)

# Tahmin edilen yaş değerini yuvarlama

pred\_age = round(pred[0][0])

print("Predicted Age:", pred\_age)

# Tahmin edilen yaş değerini görselleştirme

plt.axis('off')  # Eksenleri kapatma

plt.imshow(image[0], cmap='gray')  # Gri tonlamalı resmi görselleştirme

plt.title(f"Predicted Age: {pred\_age}")  # Tahmin edilen yaş değerini başlık olarak eklemek

plt.show()