

BİTİRME PROJESİ – RAPOR 1

FOTOĞRAFTAN DUYGU TANIMALI MÜZİK ÇALAR

20060343 Müge TETİK

19060387 Ahmet GÜNEŞ

20060405 Mehmet Batuhan TOPAL

KISIM 1: TEORİK BİLGİLER

Nöronların Farklı Öğrenme Biçimleri/ Datasetler

Nöronların neyi öğreneceklerini belirlemek için ilk önce bir dataset belirlemek gerekmektedir. Buna training set denir ve birkaç farklı türü vardır.

Unsupervised Learning: Sadece inputları içeren bir set. Network benzer inputları tanıyıp kategorize etmeye çalışıyor. Fakat pek verimli değil.

Reinforcement Learning: Set inputları ve inputların yanında ek açıklama içeriyor. Network inputa göre bir output üretiyor ve daha sonra onu ek açıklama ile kıyaslıyor ve ürettiği çıktının doğru mu yanlış mı olduğuna karar veriyor.

Supervised Learning: Inputlarla birlikte istenilen output bilgisi de veriliyor. Bu yolla network ölçülen çıktı istenilen çıktı ile aynı mı diye kontrol edebiliyor.

Bizim çalışmamız supervised learning kullanılarak yürütülecektir.

Convolutional Neural Networks

CNN, görüntüleri tanıyabilmek için kullanılan bir yaklaşımdır. Bu yaklaşım beynin Visual Cortex adı verilen bir bölgesinden esinlenerek ortaya çıkmıştır. Beynin bu bölgesinde nöronlar sadece receptive field(alıcı bölge)’dan gelen uyarılara yanıt verir. Farklı nöronların bu receptive fieldları örtüşür ve birlikte görme alanını oluştururlar.

Bu belli nöronların sadece belli özellikler tarafından aktive edilebildiği anlamına gelir. Yani görme alanında dikey bir kenar varsa farklı nöronlar, yatay bir kenar varsa farklı nöronlar aktive olacaktır.

Bunu yapay olarak gerçekleştirebilmek için de bir yılanı yılan yapan ya da bir uçağı uçak yapan ayırt edici özellikleri tespit etmeye çalışır. Fakat bunu yaparken öncelikle alt seviye özellikleri kullanır: kıvrımlar, kenarlar gibi. Daha sonra bunu daha soyut kavramlara dönüştürürler.

CNN görüntüleri işleyebilmek için birkaç farklı katman kullanır:

Convolutional Layer:

Bu katmanda görüntünün farklı özelliklerini algılamak için farklı özellikler kullanılır. Görüntü bir piksel olarak temsil edilir.

Görüntü:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filtre:

1	0	1
0	1	0
1	0	1

Filtre her seferinde önce sağa doğru 1 piksel kaydırılır. Satır sonuna gelindiğinde önce 1 piksel aşağı ve daha sonra tekrar 1 piksel sağa olacak şekilde görüntüye uygulanır. Filtre uygulanmış görüntüye Feature Map(Özellik Haritası) adı verilir. Feature Map filtrenin temsil ettiği özelliğin görüntüde nerede bulunduğunu belirtir. Bir görüntüde genelde birden fazla Feature Map bulunur.

Feature Map için bir diğer önemli özellik de stride'dır. Filtreler uygulanırken görüntünün orijinal değerini korumak önemlidir. Bu yüzden resim kenarlarına 0lardan oluşan değerleri haritaya ekler.

Non-linearity Layer:

Her convolutional katmandan sonra genelde non-linearity layer olur.

Lineerlik görüntüde bir problem oluşturur çünkü lineer çıktılarda sistem her zaman tek bir lineer işlemde birleşebilir. Bu da sistemin ayırt edici özelliklerini kaybetmesine neden olur ki bu açıkça CNN'in mantığına ters düşer.

Bu katmanda aktivasyon fonksiyonlarından biri olan Rectifier(ReLU) fonksiyonunu kullanırız. Bu fonksiyon lineerlikten kurtulmak için 0'dan küçük olan verileri 0'a eşitler.

Rectifier $\rightarrow -f(x) = \max(0, x)$

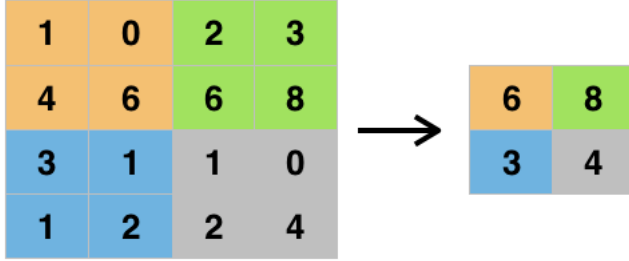
Aşağıdaki görüntüde, Feature Map'te bulunan siyah bölgeler negatif değerleri temsil eder. Rectifier fonksiyonları bunları sıfırlamıştır. Yani sonuç olarak Non-Linearity katmanı sonunda lineer değerlerden arındırılmış bir görüntüye sahip oluruz.



Pooling(Downsampling) Layer:

Gereksiz özellikleri yok saymak ve işlem gücünü azaltmak için boyutsallığı azaltmak gerekir. Boyutsallığı azaltarak aynı zamanda over-fittingi de kontrol altına almış oluruz. Farklı pooling algoritmaları bulunur ama en popülerleri max pooling'dir.

Burada filtre benzeri bir yapı oluşturulur: Pooling boyutu 2x2 dir. Pooling 4x4 görüntü üzerinde gezdirilerek en büyük değeri alır ve yeni görüntümüzde yeterli bilgiye sahip oluruz.



Flattening Layer:

Sinir Ağları veriyi tek boyutlu olarak alır. Yani diğer katmanlar boyunca oluşturulan matrislerin tek boyutlu bir diziye çevrilmesi gerekmektedir. Bu katman da diğer katmanlardan matris olarak alınan verileri tek boyutlu bir diziye sıkıştırır.

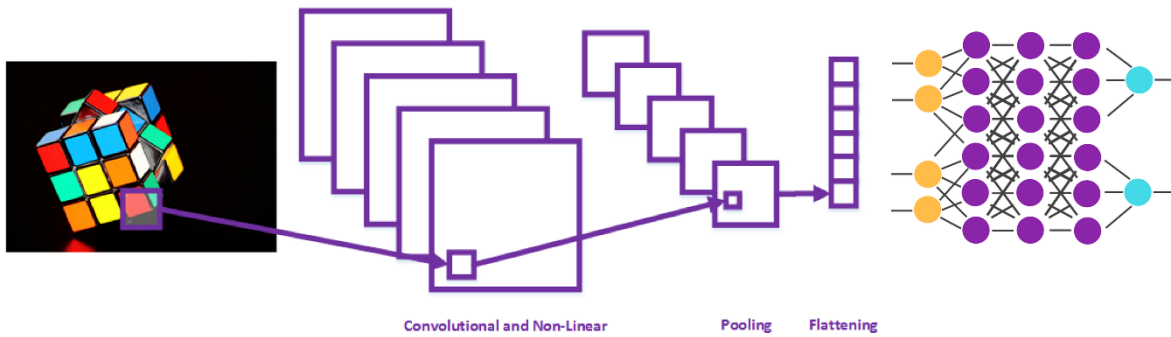
1	1	0
4	2	1
0	2	1



1
1
0
4
2
1
0
2
1

Fully-Connected Layer:

Son katman olan FC’de düzleştirilmiş girişler nöronlara bağlanır. Makine öğrenmesi klasik yapay sinir ağlarında olduğu gibi gerçekleştirilir.



Cascaded Pose Regression:

Kısaca CPR bir nesnenin başlangıç konumu verildiğinde pozunu tahmin etmeyi amaçlayan ve bunu da sürekli tekrarlayan bir şekilde ve adım adım iyileştirmeler yaparak gerçekleştiren bir tekniktir. Küçük datasetlerde bile yüksek doğruluk oranı sağlar.

CPR diğer metodlardan hem mevcut poz tahminine hem de görüntü verisine bağlı olan pose-indexed features kullanarak ayrılır. Bu özelliklerin weak invariance'ını tahmin ederek poz tahmini için bir algoritma sağlar.

Süreç kabaca bir tahmin ile başlar ve bir dizi regressor kullanılarak adım adım iyileştirilir. Şekildeki resimde θ_0 başlangıç tahmininden başlayarak θ_T 'ye kadar bir dizi regresyon yapılmıştır.

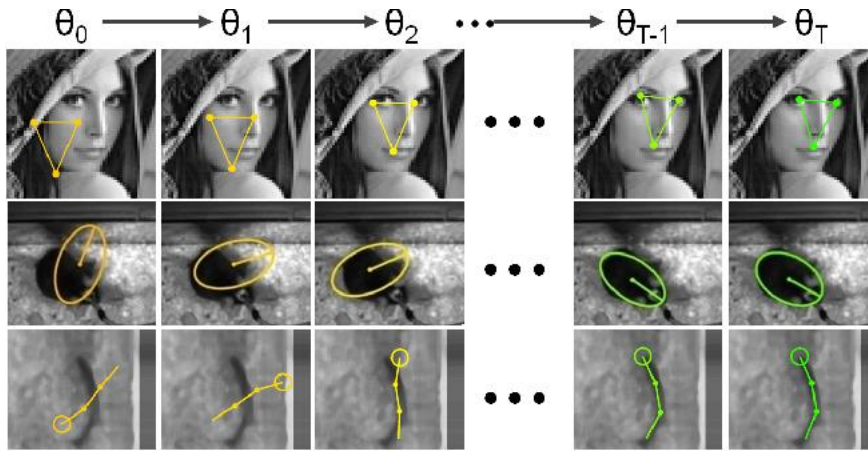


Figure 1. Object pose (green wire frame) is computed by cascaded

Konumlandırma ve ölçeklemede nesne yerini tespit etmek için kullanılan baskın yaklaşım 'sliding window' kullanmaktır. Bu teknik bir binary classification işini tekrarlar: poz parametrelerinin yeterince iyi örneklenmesi için 'x nesnesi y pozisyonunda mı?' görevi tekrarlanır.

Weak Invariance (Zayıf Değişmezlik) kavramı :

"pose-indexed features" $\rightarrow h$

poz $\rightarrow \theta$

görüntü verileri $\rightarrow I$ olmak üzere:

Olası pozlar (Θ) kümesi içindeki herhangi iki θ ve $\theta\delta$ pozu için aşağıdakiler geçerliyse, pose-indexed feature h 'nin zayıf değişmez (weak invariant) olduğu söylenir:

$$h(\theta, G(o, e)) = h(\theta\delta \circ \theta, G(o, \theta\delta))$$

Bu denklemden çıkarılacak sonuç, $h(\theta_1, G(o, \theta_2))$ çıktısının yalnızca nesneye (o) ve giriş pozu θ_1 ile gerçek poz θ_2 arasındaki göreceli poza ($\theta_1 \circ \theta_2$) bağlı olmasıdır. Bu, pozun mutlaka doğru

olmasa da tutarlı bir tahmini olduđu sürece özellik fonksiyonunun çıktısının sabit kalacağı anlamına gelir. Daha basit bir ifadeyle bu özellikler, tahmini poz tamamen doğru olmasa bile benzer pozlar için benzer çıktılar üretmek üzere tasarlanmıştır.

Ayrıca, bu zayıf değişmez özellikler standart işlemler kullanılarak birleştirildiğinde veya oluşturulduğunda, ortaya çıkan bileşik özellikler aynı zamanda zayıf değişmezlik özelliğini de sergiler. Bu, zayıf şekilde değişmez olan bu özelliklerin birleştirilmesiyle oluşturulan herhangi bir özelliğin, zayıf şekilde değişmez olma özelliğini koruyacağı ve böylece farklı pozlara göre davranışlarında tutarlılık sağlanacağı anlamına gelir.

OpenCV ve Haar Cascade Face Detection

Görsel algılama ve analiz alanındaki önemli bir oyuncu olan OpenCV kütüphanesine odaklanmaktadır. OpenCV, bilgisayarlı görüş ve görüntü işleme projelerinde kapsamlı bir şekilde kullanılan, açık kaynaklı bir kütüphanedir. Ayrıca, bu rapor özellikle yüz tespiti konusunda oldukça etkili olan Haar Cascade sınıflandırıcılarının önemine vurgu yapmaktadır.

OpenCV'nin esnek yapısı, geniş bir kullanıcı tabanına sahip olmasını sağlar ve geliştiricilere zengin bir set araç ve algoritma sunar. Bu kütüphane, bilgisayarlı görüş uygulamalarının geliştirilmesinde güçlü bir temel oluşturur.

Haar Cascade sınıflandırıcıları ise, önceden eğitilmiş modeller aracılığıyla nesne tanıma görevlerinde üst düzey performans sağlar. Özellikle yüz tespiti için kullanılan bu sınıflandırıcılar, görüntüdeki belirli özellikleri analiz ederek nesneleri algılamada etkin bir yol sunar. Bu, özellikle real-time uygulamalarda ve video analizlerinde önemli bir avantaj sağlar.

Sonuç olarak, OpenCV ve Haar Cascade sınıflandırıcıları, bilgisayarlı görüş projelerinde güvenilir ve etkili bir şekilde kullanılan araçlardır. Bu rapor, bu teknolojilerin temel özelliklerine odaklanarak, geliştiricilere bu alanda daha derinlemesine keşif yapma ve projelerine bu kuvvetli temeller üzerine inşa etme fırsatı sunmaktadır.

1. OpenCV Nedir?

OpenCV (Open Source Computer Vision), bilgisayarlı görüş ve görüntü işleme projelerinin vazgeçilmez bir kaynağıdır. Bu açık kaynaklı kütüphane, çok sayıda programlama dilini destekleyerek, geliştiricilere geniş bir dil yelpazesi üzerinde çalışma esnekliği sağlar. Python, C++, Java gibi popüler dillerde kullanılabilmesi, projelerin farklı ekosistemlere sorunsuz entegrasyonunu mümkün kılar.

OpenCV'nin çekirdek yetenekleri, gelişmiş görüntü işleme algoritmalarından kamera kalibrasyonuna, nesne tespitinden yüz tanımaya kadar geniş bir yelpazeyi içerir. Bu özellikler, hem akademik araştırmalarda hem de endüstriyel uygulamalarda çeşitli alanlarda başarıyla kullanılabilmesini sağlar. Gelişmiş algoritma seti, kullanıcılara çeşitli problemlere uygun çözümler üretme esnekliği sunar.

OpenCV'nin sürekli gelişimi ve büyük topluluğunun sağladığı destek, kullanıcıların güncel teknoloji ve en iyi uygulamalardan faydalanmalarına yardımcı olur. Bu kütüphane, bilgisayarlı görüş uygulamalarını geliştirmek isteyen herkes için güçlü bir araç ve kaynaktır.

2. Haar Cascade Nedir?

Haar Cascade, nesne tanıma alanında etkili bir kullanım sunan bir tür nesne tespit algoritmasıdır. Özellikle yüz tespiti için tasarlanmış olan önceden eğitilmiş Haar Cascade sınıflandırıcıları, belirli görsel özelliklere dayanarak nesneleri tanıma yeteneğiyle öne çıkar.

Bu algoritmanın temel ilkesi, nesnelerin özel özelliklerini (Haar özellikleri olarak adlandırılır) analiz ederek tanımadır. Özellikle yüz tespiti için kullanıldığında, bu özellikler genellikle farklı tonlamalardaki pikseller arasındaki yoğunluk değişikliklerini temsil eder. Yüzlerin genel formu ve özellikleri, bu özelliklerin kombinasyonlarıyla temsil edilerek algılanır.

Haar Cascade sınıflandırıcıları, öğrenilmiş modelleri kullanarak nesne tespiti gerçekleştirir. Eğitim aşamasında, birçok pozitif ve negatif örneğin kullanılmasıyla algoritma, belirli nesne sınıflarını başarıyla tanıyan bir model oluşturur. Bu sayede, gerçek zamanlı uygulamalarda hızlı ve etkili nesne tespiti sağlar.

Yüz tespiti için özellikle kullanıldığında, Haar Cascade sınıflandırıcıları yüzlerin genel konturlarını ve özelliklerini yakalamada yüksek başarı elde eder. Bu, video analizi, güvenlik sistemleri, otomatik odaklama gibi birçok uygulama alanında yaygın olarak kullanılmalarını sağlar.

Sonuç olarak, Haar Cascade algoritmaları, belirli nesneleri tanıma görevlerinde etkili bir çözüm sunarak, bilgisayarlı görüş projelerine büyük katkı sağlar.

3.Yüz Tespiti Kullanımı

OpenCV'nin yüz tespiti özelliklerini kullanmak için, temelde `cv2.CascadeClassifier` sınıfını entegre etmek gerekmektedir. Bu sınıf, önceden eğitilmiş Haar Cascade sınıflandırıcı dosyalarını kullanarak nesne tespiti yeteneklerini projenize entegre etmenizi sağlar.

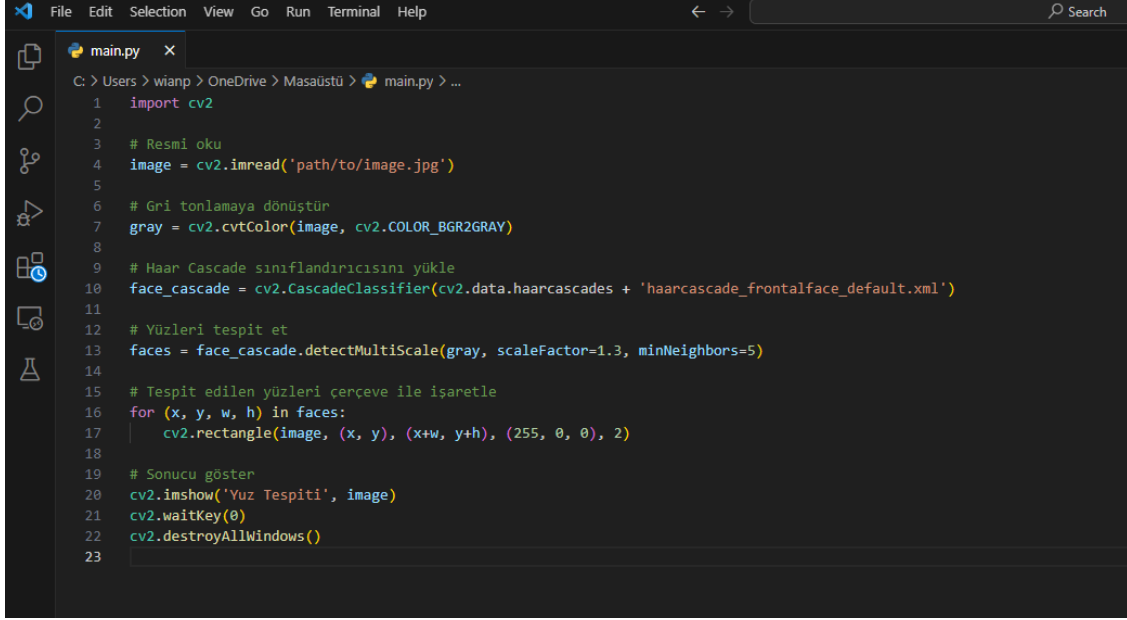
İlk aşama, Haar Cascade sınıflandırıcı dosyasının `cv2.CascadeClassifier` sınıfına yüklenmesi ve yüz tespiti için hazır hale getirilmesidir. Bu dosya, genellikle OpenCV'nin içinde bulunan önceden eğitilmiş bir modeli temsil eder ve yüz tespiti konusunda özel olarak optimize edilmiştir.

Ardından, `detectMultiScale` fonksiyonu, görüntüdeki potansiyel yüzleri tespit etmek için kullanılır. Bu fonksiyon, belirli bir nesnenin (örneğin yüz) konumunu ve boyutunu bulmak üzere özel olarak tasarlanmıştır. Parametreler arasında `scaleFactor` ve `minNeighbors` bulunur ve bu değerler, tespitin hassasiyeti ve doğruluğu üzerinde kontrol sağlar.

Son adımda, tespit edilen yüzler, dikdörtgen çerçeveler içine alınarak görselleştirilir. Bu, kullanıcıya tespit edilen yüzleri net bir şekilde görmesini sağlar ve aynı zamanda bu tespitin gerçekleştiği konumu daha iyi anlamasına yardımcı olur.

Bu basit adımlar, OpenCV'nin güçlü yüz tespiti yeteneklerini kullanarak görüntü üzerindeki yüzleri hızlı ve etkili bir şekilde belirlemek için izlenir. Bu süreç, güvenlik sistemleri, otomatik odaklama, duygu analizi ve birçok diğer uygulama alanında başarıyla kullanılabilen bir güç aracıdır. OpenCV'nin esnekliği ve Haar Cascade sınıflandırıcılarının etkinliği, bilgisayarlı görüş projelerine sağlam bir temel oluşturur.

4. Kod örneği



```
File Edit Selection View Go Run Terminal Help
main.py x
C: > Users > wianp > OneDrive > Masaüstü > main.py > ...
1 import cv2
2
3 # Resmi oku
4 image = cv2.imread('path/to/image.jpg')
5
6 # Gri tonlamaya dönüştür
7 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9 # Haar Cascade sınıflandırıcısını yükle
10 face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
11
12 # Yüzleri tespit et
13 faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
14
15 # Tespit edilen yüzleri çerçeve ile işaretle
16 for (x, y, w, h) in faces:
17     cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
18
19 # Sonucu göster
20 cv2.imshow('Yüz Tespiti', image)
21 cv2.waitKey(0)
22 cv2.destroyAllWindows()
23
```

5. Avantajlar:

Haar Cascade sınıflandırıcıları, bilhassa nesne tespiti görevlerinde sağladığı hız ve etkinlik ile ön plana çıkar. Bu algoritmalar, önceden eğitilmiş modelleri kullanarak görüntülerde belirli nesneleri tanımlama yeteneğiyle bilgisayarlı görüş projelerinde kritik bir rol oynar. Hızlı işleme kapasitesi, özellikle gerçek zamanlı uygulamalarda ve video analizlerinde önemli bir avantaj sunar.

OpenCV'nin geniş topluluğu ve sağladığı belgeler, kullanıcıların bu teknolojiyi hızla benimsemesine ve uygulamalarına entegre etmesine yardımcı olur. Bu destek, geliştiricilere başlangıçtan ileri düzeye kadar geniş bir yelpazede bilgi ve çözüm sunar. Kapsamlı belgeler, kullanıcıların Haar Cascade sınıflandırıcılarını etkili bir şekilde kullanmalarına ve özel ihtiyaçlarına uygun çözümler geliştirmelerine olanak tanır.

Sonuç olarak, Haar Cascade sınıflandırıcıları, hız ve etkinlikleriyle beraber OpenCV'nin geniş topluluğu ve kaynakları ile bir araya geldiğinde güçlü bir kombinasyon oluşturur. Bu birleşim, nesne tespiti alanındaki çeşitli uygulamalarda başarılı sonuçlar elde etmek isteyen geliştiricilere sağlam bir temel sunar.

6. Sonuç:

OpenCV ve Haar Cascade sınıflandırıcıları, bilgisayarlı görüş ve nesne tanıma projelerinde kullanılan son derece güçlü araçlardır. Bu teknolojiler, özellikle yüz tespiti gibi karmaşık nesne tanıma görevlerinde üstün performans sergiler.

OpenCV, geniş yelpazesıyla bilgisayarlı görüş uygulamaları için endüstri standardı haline gelmiş bir kütüphanedir. Haar Cascade sınıflandırıcıları ise, önceden eğitilmiş modeller aracılığıyla nesne tanıma konusunda etkileyici bir hassasiyet ve hız sunar.

Bu teknolojiler, güvenlik sistemlerinden otomatik odaklamaya, duygu analizinden nesne sayımına kadar birçok uygulama alanında başarıyla kullanılmaktadır. Geliştiricilere, projelerinde hızlı ve etkili nesne tanıma yetenekleri sağlayarak, karmaşık görsel verilerle çalışma konusunda önemli bir avantaj sunar.

Sonuç olarak, OpenCV ve Haar Cascade sınıflandırıcıları, bilgisayarlı görüş projelerinin temel taşlarıdır ve bu alanlarda çalışan geliştiricilere güçlü, güvenilir ve kullanıcı dostu araçlar sunarlar.

Bu rapor, OpenCV ve Haar Cascade ile yüz tespiti konusunda temel bilgiler sağlamayı amaçlamaktadır.

Kısım 2: Uygulama: Mobil, React Native

React Native, Facebook tarafından geliştirilen açık kaynaklı bir mobil uygulama çerçevesidir. Bu raporda, React Native'in temel amaçları, yaklaşımı, öne çıkan özellikleri, avantajları ve dezavantajları detaylı bir şekilde incelenecektir.

1. Temel Amaç ve Yaklaşım

React Native, JavaScript ve React kullanarak hem iOS hem de Android platformları için uygulama geliştirmeyi sağlamaktadır. JSX adı verilen bir sözdizimi ile geliştiricilere JavaScript içinde XML benzeri bir dil ile kullanıcı arayüzü bileşenlerini tanımlama esnekliği sunar. Bu, geliştirme sürecini daha anlaşılır ve etkili kılar.

2. Öne Çıkan Özellikler

Hızlı Geliştirme Döngüsü: "Hot Reloading" özelliği ile kod değişiklikleri uygulama çalışırken hemen görülebilir, bu da geliştirme sürecini hızlandırır.

Geniş Bileşen Kitaplığı: Sorunsuz çalışan geniş bir bileşen kitaplığı, geliştiricilere zengin ve kullanışlı arayüzler oluşturmak için güçlü araçlar sunar.

3. Avantajlar ve Dezavantajlar

Avantajlar:

- ❖ Çok platformlu geliştirme

- ❖ Kullanıcı dostu arayüz bileşenleri
- ❖ Platforma özgü işlevsellik

Dezavantajlar:

- ❖ Öğrenme eğrisi
- ❖ Performans
- ❖ Departman sayısı

4. Sonuç

React Native, çok yönlü bir mobil uygulama geliştirme çözümü olarak öne çıkıyor. Ancak, hızlı geliştirme döngüsü, geniş bileşen kitaplığı ve platforma özgü işlevsellik gibi avantajları, öğrenme eğrisi ve performans gibi dezavantajlarla birlikte değerlendirilmelidir. Projenin gereksinimlerine ve geliştirici tercihlerine bağlı olarak uygun bir seçenek olabilir.

Projede Kullanılacak Feature ve Kütüphaneler:

Bu React Native uygulaması, kullanıcıların duygu durumunu analiz ederek onlara uygun müzik önerileri sunmayı hedeflemektedir. Aşağıda belirtilen kütüphaneler ve özellikler, projenin başarıyla geliştirilmesi için kullanılacaktır.

- Kamera Entegrasyonu:

Projenin temel işlevselliği arasında yer alan kamera entegrasyonu için, react-native-camera kütüphanesi tercih edilecektir. Bu güçlü kütüphane, kullanıcıların uygulama içinde fotoğraf çekmelerini ve bu fotoğrafları yüklemelerini mümkün kılacaktır. react-native-camera, kamera işlemlerini basit ve etkili bir şekilde yönetebilmemiz için gelişmiş özelliklere sahip olup, kullanıcıların uygulama deneyimini zenginleştirmemize olanak tanıyacaktır.

- Görüntü İşleme:

Proje kapsamında duygu durumu analizi için yüksek performanslı görüntü işleme kütüphaneleri olan OpenCV veya TensorFlow gibi güçlü çözümler değerlendirilecektir. Bu kütüphaneler, kullanıcıların yüzlerini tanımlama ve duygu durumlarını kesin bir şekilde belirleme konusunda önemli yeteneklere sahiptir. Projenin amacına uygun olarak seçilecek olan bu kütüphaneler, kullanıcı deneyimini derinleştirmek ve duygu durumu analizi sonuçlarını daha hassas bir şekilde sunmak için etkili bir araç seti sağlayacaktır.

- Model Entegrasyonu:

Proje bünyesindeki duygu durumu modeli, TensorFlow ve TensorFlow Lite gibi kapsamlı kütüphaneler kullanılarak React Native projesine başarılı bir şekilde entegre edilecektir. Bu entegrasyon, kullanıcının duygu durumunu daha derinlemesine anlamamıza ve buna uygun müzik önerileri sunmamıza olanak sağlayacaktır. Seçilen TensorFlow tabanlı kütüphaneler, gelişmiş öğrenme algoritmalarıyla duygu durumu analizini optimize ederek, kullanıcının deneyimini kişiselleştirmek ve duygusal bağ kurmak adına kritik bir rol oynayacaktır.

- Spotify API Entegrasyonu:

Kullanıcının duygu durumuna özel müzik önerileri sunmak adına, proje içinde react-native-spotify-sdk ve Axios gibi etkili HTTP istek yönetim kütüphaneleri kullanılacaktır. Bu entegrasyon, Spotify API ile etkileşim kurarak, kullanıcının anlık duygu durumuna uygun bir müzik listesi oluşturmayı amaçlamaktadır. react-native-spotify-sdk kullanılarak Spotify platformu ile güçlü bir bağlantı kurulacak ve Axios ile yapılan HTTP istekleriyle kullanıcının tercihlerine uygun şarkılar anında çekilebilecektir. Bu sayede, uygulama kullanıcı deneyimini zenginleştirecek, duygusal bağ kurmayı güçlendirecek ve müzik keyfini kişiselleştirecektir.

- State Management:

Uygulama içindeki durumu etkin bir şekilde yönetmek amacıyla, Redux Toolkit kullanılacaktır. Kullanıcı oturumu, duygu durumu ve diğer kritik veriler, Redux Toolkit'in sunduğu merkezi yönetim yapısı kullanılarak kolayca kontrol edilebilecektir. Bu çözüm, uygulama genelinde tutarlı bir durum yönetimi sağlayarak, verilerin güvenilir bir biçimde paylaşılmasını ve güncellenmesini mümkün kılacaktır. Redux Toolkit, karmaşık durumların yönetimini basitleştirerek, uygulama içindeki veri akışını optimize edecektir.

- Navigasyon:

Uygulama içinde sezgisel ve akıcı bir gezinme deneyimi sunabilmek için, react-navigation kütüphanesi tercih edilecektir. Bu kütüphane, farklı sayfalar arasında rahat bir geçiş sağlamak için gelişmiş navigasyon özellikleri sunar. react-navigation, uygulamanın karmaşıklığını azaltarak kullanıcılara berrak bir gezinme yapısı sunar ve uygulama içindeki her sayfa arasındaki etkileşimleri optimize eder. Bu sayede, kullanıcılar uygulamanızda gezinirken beklenen ve akıcı bir deneyim yaşarlar.

- Stil:

Uygulamanın görsel tasarımını etkili bir şekilde yönetmek için, React Native'in kendi StyleSheet özelliği kullanılacaktır. Bu özellik, stil tanımlarını düzenli bir şekilde tutarak, uygulamanın genel görünümünü tutarlı kılacaktır. Ayrıca, tasarım sürecini daha da kolaylaştırmak ve esnekliği artırmak adına, styled-components veya styled-system gibi CSS-in-JS çözümleri de değerlendirilecektir. Bu çözümler, bileşen tabanlı bir yaklaşım benimseyerek, uygulama içindeki stillemenin daha modüler ve sürdürülebilir olmasını sağlayacaktır. Bu sayede, tasarım değişiklikleri hızlı bir şekilde uygulanabilir ve uygulamanın genel estetiği kolayca güncellenebilir.

- Asenkron İşlemler:

Uygulama içindeki asenkron işlemleri başarılı bir şekilde yönetmek amacıyla, modern JavaScript'in async/await yapısı kullanılacaktır. Bu yapının kullanımı, kodun okunabilirliğini artırırken, asenkron işlemlerin yönetimini de daha şeffaf hale getirecektir.

Ayrıca, ihtiyaç duyulması durumunda, daha karmaşık asenkron senaryoları ele almak ve yönetmek için redux-thunk veya redux-saga gibi çözümler entegre edilecektir. Bu Redux ortamlarının kullanımı, uygulama içindeki durumu güncellemek veya dış kaynaklardan veri çekmek gibi asenkron işlemleri daha etkili bir biçimde gerçekleştirebilmemizi sağlayacaktır. Bu modüller

özmler, asenkron işlemlerin kontrolünü ve takibini optimize ederek, uygulama performansını artıracak ve daha güçlü bir asenkron işlemler yönetimi sunacaktır.

- UI Kütüphaneleri:

Kullanıcı arayüzünü oluşturmak ve estetik bir görünüm elde etmek için, popüler ve güvenilir UI kütüphaneleri tercih edilecektir. Bu kapsamda, react-native-elements, react-native-paper, veya native-base gibi öncü kütüphaneler, uygulamanın görsel tasarımını zenginleştirmek ve kullanıcılar arasında tutarlı bir deneyim sunmak adına kullanılacaktır.

Bu kütüphaneler, önceden tasarlanmış bileşenler, stil temaları ve kullanışlı UI öğeleri sunarak, geliştiricilere hızlı ve etkili bir şekilde kullanıcı arayüzü oluşturma imkanı tanır. Ayrıca, bu kütüphanelerin modüler yapısı, uygulamanın büyümesi ve değişmesi sürecinde esnekliği artırarak, tasarımın kolayca güncellenmesine olanak sağlar. Bu seçilen kütüphaneler, kullanıcı arayüzünün özelleştirilebilir, etkileşimli ve modern bir şekilde tasarlanmasını destekleyerek, uygulamanın genel estetiğini güçlendirecektir.

- Test ve Hata İzleme:

Uygulama geliştirme sürecinde güvenilirliği sağlamak ve sorunsuz bir deneyim sunmak adına, Jest veya Detox gibi kapsamlı test kütüphaneleri kullanılarak kapsamlı bir test süreci uygulanacaktır. Bu test kütüphaneleri, uygulamanın farklı bileşenlerini ve senaryolarını test etme konusunda gelişmiş yeteneklere sahiptir, böylece uygulama performansı ve istikrarı güvence altına alınacaktır.

- Animasyonlar:

Kullanıcı deneyimini daha etkileyici ve akıcı hale getirmek adına, uygulama içinde çeşitli animasyonları başarıyla entegre etmek için react-native-reanimated veya react-navigation-shared-element gibi güçlü kütüphaneler kullanılacaktır. Bu kütüphaneler, uygulama içindeki geçişleri ve etkileşimleri daha canlı ve doğal hale getirerek, kullanıcılara görsel olarak iyi bir deneyim sunacaktır.

react-native-reanimated, hareketli bileşenlerin daha pürüzsüz ve hızlı bir şekilde çalışmasını sağlamak için gelişmiş animasyon yeteneklerine odaklanırken, react-navigation-shared-element ise farklı sayfalar arasındaki öğeler arasında animasyonlu geçişlerin kolayca yönetilmesine imkan tanır. Bu kütüphanelerin kullanımı, uygulama içindeki animasyonların sadece estetik değil, aynı zamanda kullanıcı etkileşimini artıran önemli bir bileşen haline gelmesine olanak sağlayacaktır.

- Geliştirme ve Hata Ayıklama Araçları:

Uygulama geliştirme sürecini optimize etmek ve hata ayıklama işlemlerini daha etkili kılmak amacıyla, çeşitli güçlü araçlar kullanılacaktır. Bu araçlar arasında, React DevTools, Flipper, ve Reactotron gibi önemli geliştirme ve hata ayıklama araçları yer alacaktır.

React DevTools: Bu araç, React uygulamalarının geliştirilmesini kolaylaştıran bir Chrome eklentisidir. Component ağacını inceleme, state ve props değerlerini anlık olarak gözlemleme gibi özellikleri içerir, bu da geliştiricilere hızlı bir geri bildirim sağlar.

Flipper: Geliştiricilerin uygulama performansını, network trafiğini, ve uygulama durumunu anlık olarak gözlemlemelerini sağlayan bir hata ayıklama ve inceleme platformudur. Çeşitli eklentileri sayesinde özelleştirilebilir ve genişletilebilir bir yapıya sahiptir.

Reactotron: React uygulamalarını hızlı bir şekilde hata ayıklamak için tasarlanmış bir masaüstü uygulamasıdır. State, action ve network istekleri gibi önemli bilgileri anlık olarak gözlemleyebilme özelliğine sahiptir. Aynı zamanda, Redux ve MobX gibi state yönetim kütüphaneleri ile uyumlu olarak çalışır.

Bu araçlar, geliştiricilere hızlı geliştirme, derinlemesine inceleme ve hata ayıklama süreçlerinde önemli avantajlar sunarak, uygulamanın güvenilir ve yüksek kaliteli bir şekilde geliştirilmesine olanak sağlar.

Bu kütüphaneler ve özellikler, projenin geliştirilmesini ve kullanıcıya en iyi deneyimi sunmayı amaçlayan bir çerçeve oluşturacaktır.

Kısım 3: Spotify API, Model Oluşturma

Projede gerçeklemeye çalışılacak uygulamalarda kullanılan bir yöntem olarak API aracılığıyla bağlantı kurulacak. Hazırlanacak projede son aşama olarak son kullanıcıya

istenilen önerileri yapabilmek için Spotify gibi bir uygulamadan müzikleri kullanıcıya önerebilmek için API kullanılması gerekmekte.

Adım 1: Spotify API'lerine Erişim

Spotify for Developers'a üye olarak aktif API'lar araştırıldı. Burada bir hesap içeren API erişimi için gerekli olan kimliğin alınması sağlandı.

API Belgelerini İncele: Spotify'ın sunduğu API sertifikaları incelendi. Bu belgeler, kullanılabilen uç nokta'ları, istek sırasında kullanılacak şifreleri ve API'nin nasıl oynanacağına dair ayrıntılı bilgiler içerir.

Adım 2: Duygu Durumu Modeli Oluşturma

Veri Toplama ve Etiketleme: Şarkı şarkıları ve Toplam duygu durumlarını içeren bir veri kümesi oluşturulması gerekecek. Bu verileri damgalamak için duygu analizi yapabilirsin. Örneğin bir şarkının neşeli, hüzünlü, enerjik veya sakin olduğunu belirleyeceksin.

Model Eğitimi: Veri kümeniyle bir duygu analizi modeli oluşturulması. Bu model, şarkıların duygu durumlarını tahmin etmek için kullanılacak.

Adım 3: Spotify Verilerini Kullanmak

Duygu Analizi: Spotify API'ları aracılığıyla şarkı şarkıları ve özellikleri (tempo, danslık, enerji vb.) al. Daha sonra bu veriler oluşturulan duygu analizi modeline gönderilerek şarkıların duygu durumlarını tahmin etmekte kullanılacaktır.

Öneri Algoritması: Duygu durumlarına göre şarkılarını önermek için analiz sonuçları kullanılır. Örneğin neşeli bir moddaysa neşeli şarkılar, hüzünlüyse hüzünlü şarkılar gibi önerilerde bulunabilirsin.

Adım 4: Kullanıcı Arayüzü Geliştirme

Arayüz Tasarımı: Kullanıcılar için bir tasarım oluşturur. Bu sürümlerde, kullanıcıların duygu durumlarını seçmelerine ve önerilen parçaları görmelerine izin verilir.

API Entegrasyonu: Spotify API'ını bu tarife entegre etmekte kullanılıyor. Kullanıcıların ayrıntıları duygu değişimlerine göre API'dan şarkı önerilerini alma ve göstermeye yarar..

Spotify API Kullanımı:

1-) Yetkilendirme ve Kimlik Doğrulama:

- API'yi kullanmak için yetkilendirme işlemi gereklidir. Yetkilendirme için OAuth 2.0 protokolünü kullanarak kullanıcı kimlik doğrulaması yapılabilir.
- Kullanıcı kimlik ücreti sonucu elde edilen erişim belirteci (erişim belirteci), API bolluğu yetkilendirmek için kullanılır.

2-) Endpoint'ler ve Veri Erişimi:

- Spotify API, çeşitli uç noktalar aracılığıyla farklı veri türlerine erişim sağlar.

Örneğin:

Şarkılar: /tracks

Sanatçılar: /artists

Albümler: /albümler

Oynatma listeleri: /playlists

- Her endpoint, istek yapılan türüne göre farklı etki ve filtreler kabul edilir

3-) Örnek İstekler:

- Örneğin, belirli bir şarkının şarkıları almak için şu gibi bir GET isteği yapılabilir:

https://api.spotify.com/v1/tracks/{track_id}

- Veri çekme oranlarında genellikle etkisi mevcuttur. Örneğin belirli bir sanatçının albümlerini listelemek için:

https://api.spotify.com/v1/artists/{artist_id}/albums

4-) Özellikler ve Veri Formatı:

- Her Şarkı, sanatçısı, albümü vb. için çeşitli özellikler ve bilgiler mevcuttur. Örneğin şarkı özellikleri arasında danslık, enerji, tempo gibi değerler bulunabilir.
- Veri genellikle JSON biçiminde döner. Bu verilerin alınması ve çalışırken JSON formatına dikkat edilmesi önemlidir.

5-) Rate Limit ve Hata Durumları:

- Spotify API, sorgu başına belirli bir hızda istek yapmanıza izin verir. Hız limitlerini aşmamak önemlidir.
- API isteklerinde olası hata durumlarına karşı uygun hata işleme modeli mevcuttur. Örneğin, 429 Too Many request, bilgisayarın çalışmasının beklediğini veya hata mesajını gösterdiği gibi.

6-) Geliştirme ve Test Süreci:

- Geliştirme süresi boyunca API rutinlerini test etmek için örnek veriler ve postacı gibi API test araçları mevcuttur.
- Hata ayıklama ve API yanıtlarını analiz etmek için loglama ve izleme yapılabilir.

Spotify API'sini kullanarak veri çekme işlemlerini yaparken, belirli uç noktalar ve isteklerin nasıl gerçekleştirilmesi önemlidir. API dosyalarının eklenmesi, örnek istekleri yapmak ve geri dönüşleri analiz etmek, bu sürecin başarılı bir şekilde ilerlemesine yardımcı olacaktır.

Modellerin Araştırılması

Duygu Sınıflandırma Modelleri:

- Doğrusal Modeller:

Lojistik Regresyon: Basit ve hızlı bir modeldir. Özelliklerle duygu durumları arasındaki ilişkiyi öğrenir.

- Destek Vektör Makineleri (SVM):

Çekirdekli SVM: Özellikler arasındaki karmaşık ilişkiler için kullanılabilir.

- Karar Ağaçları ve Ormanlar:

Karar Ağaçları: Özelliklerin korunan bir yapıda sınıflandırılmasını sağlar.

Rastgele Ormanlar: Birden fazla karar ağacının bir araya getirilmesiyle daha güçlü bir model oluşturulması bir yöntem.

- Derin Öğrenme Modelleri:

Yapay Sinir Ağları (ANN): Özellikle çok katmanlı yapılarla duygu analizi için kullanılabilir.

Rekürrent Sinir Ağları (RNN): Zaman serileri gibi sıralı verilerde duygu durumu analizi için uygun olabilir.

Duyarlılık Doldurmalı Sıralı Bellek (LSTM) ve Uzun Kısa Süreli Bellek (GRU): Metin gibi uzun bağlamasal bağımlılıkları olan verilerde daha iyi performans gücü.

- Eğitim Teknikleri:

Veri Toplama ve Temizleme: Spotify API'sı tamamen üzerinden şarkı parçalarından ayrılır ve bu verilerin duygu durumlarına göre etiketlenebilir. Bu adım, doğru ve düzenli bir veri seti oluşturmak için önemlidir.

Öznitelik Mühendisliği: Şarkı özelliklerini (tempo, danslık, enstrümantallık vb.) kullanılabilir bir formata dönüştürülerek modellenilecek bir yapıya kavuşturmak.

Model Seçimi ve Eğitimi: Belirlenen model tiplerini kullanarak veri setini eğitir. Bu süreçte veri setini eğitim, süreklilik ve test kitapları olarak ayırarak modelinin değerlendirilmesi.

Modelinizi Ayarlayın ve Hiperparametreleri Optimizasyonu: Modelinizi eğitirken, hiperparametreleri (örneğin, öğrenme oranı, dönem sayıları, derinlik vb.) ayarlayarak en iyi performansı elde etmeye öğretin.

Model Değerlendirmesi: Eğitim sürecinin başlamasından sonra, test veri seti üzerinde modelin doğruluğu, dayanıklılığı, geri çağırılmasını ve F1 skorunu gibi metriklerle değerlendirir.

Model Dağıtımı: Modelinizi kullanıma hazır hale getirin ve Spotify API'sı ile entegre bir yapıya sahip olun.

Sonuç:

Genel Özet: Spotify API kullanılarak duygu durumu özeti yapılacak ve elde edilen analizlerle kullanıcılara özel şarkı ve çalma listesi önerileri sunulacak. Bu süreçte veri toplama, model eğitimi, API kullanımı ve kullanıcı arayüzü geliştirme adımlarının önemi aktarılır.

Kaynakça:

<https://rubikscore.net/2018/02/26/introduction-to-convolutional-neural-networks/>

<https://rubikscore.net/2018/02/19/artificial-neural-networks-series/>

<https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad>

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>

<https://pdollar.github.io/files/papers/DollarCVPR10pose.pdf>

https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalcatface.xml

<https://docs.opencv.org/4.x/>

<https://reactnative.dev/docs/getting-started>

<https://medium.com/@lorelablaka/extract-data-using-spotify-api-889222835bf4>

<https://engineering.atspotify.com/2015/03/understanding-spotify-web-api/>

Tarih: 1/12/2023