ÇALIŞMANIN ADI

Yemek tesbiti

DANIŞMAN
Dr.Öğr.Üysi

HAZIRLAYAN
Ahmed Salih

Teşekkürler

Bu projeyi tamamlama fırsatını bize veren herkese teşekkürlerimizi sunarız. Bu çalışma, ekibimizin desteği, kullandığımız teknolojilerin sağlamlığı ve açık kaynak topluluğunun sunduğu değerli kaynaklar olmadan mümkün olmazdı.

İÇİNDEKİLER

BEYAN

Bitirme çalışması içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, çalışmada yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Ad Soyad:

İmza :

Ad Soyad :

İmza :

ÖZET

Bu projede, gıda öğelerini tanıyabilen ve sınıflandırabilen bir Gıda Tanıma Sistemi oluşturduk. Python, OpenCV, Derin Öğrenme ve Roboflow teknolojilerini kullanarak bir derin öğrenme modeli geliştirdik. Sistem, görüntü işleme ve sinir ağı modellemesi kullanarak çeşitli gıda öğelerini etkili bir şekilde tanımaktadır.

Sistem öncelikle geniş bir veri setini toplamak için Roboflow platformunu kullandı. Daha sonra OpenCV ile görüntü ön işlemeyi gerçekleştirdik ve Python kullanarak bir Convolutional Neural Network (CNN) modeli eğittik. Model, test setindeki gıda öğelerini yüksek bir doğrulukla tanıyabilmiştir.

Bu proje, bu tür bir teknolojinin potansiyel uygulamalarını göstermektedir. Gıda Tanıma Sistemi, sağlık uygulamaları, perakende sektörü ve gıda endüstrisinde kullanılabilir. Bu çalışma, Python, OpenCV, Derin Öğrenme ve Roboflow teknolojilerinin bir arada kullanılmasının gücünü ve potansiyelini göstermektedir. Modelin geliştirilmesi ve iyileştirilmesi için gelecekteki çalışmalara yön verecek bulgular sağlamıştır.

ŞEKİLLER LİSTESİ

Proje Raporu - Python, OpenCV ve Derin Öğrenme Kullanarak Gıda Tanıma Sistemi

1.Giriş

Akıllı teknolojilerin ve AI uygulamalarının hüküm sürdüğü bir çağda, makine öğrenmesi ve yapay zekanın kapsamı birçok ufku aşmıştır. Bu uygulamalardan biri gıda endüstrisindedir, burada AI, çeşitli gıda maddelerinin tanınması ve sınıflandırılması konusunda yardımcı olabilir. Bu rapor, Python, OpenCV, Derin Öğrenme tekniklerini ve Roboflow platformunu kullanarak bir Gıda Tanıma Sistemini geliştirmeyi amaçlayan projeyi belgeler.

1.1.Yemek Tanıması:

Makine öğrenimi, yemeklerin tanınması için kullanılabilir. Makine öğrenimi, bilgisayar sistemlerinin verilerden öğrenme yeteneği kazanmasını sağlayan bir yapay zeka dalıdır. Yemeklerin tanınması için kullanılan en yaygın makine öğrenimi yöntemlerinden biri, derin öğrenme algoritmaları kullanarak yapay sinir ağlarının eğitilmesidir.

Yemek tanıması için makine öğrenimi adımları genellikle şu şekilde gerçekleştirilir:

Veri Toplama: İlk adım, farklı yemek türlerinden örnek veri setlerinin toplanmasıdır. Bu veri setleri, yemeklerin görüntülerini ve ilgili etiketleri içermelidir. Etiketler, her görüntünün hangi yemek türüne ait olduğunu belirtir.

Veri Ön İşleme: Veri ön işleme aşamasında, toplanan veri seti düzenlenir ve işlenir. Görüntüler genellikle standart boyutlara yeniden boyutlandırılır ve gerektiğinde kontrast veya parlaklık ayarlamaları yapılır. Ayrıca, veri seti genellikle eğitim, doğrulama ve test kümelerine ayrılır.

Model Eğitimi: Model eğitimi aşamasında, derin öğrenme algoritmaları kullanılarak bir yapay sinir ağı oluşturulur ve eğitilir. Bu ağ, görüntülerden özellikler çıkarmak ve yemek türünü tahmin etmek için kullanılır. Eğitim veri seti kullanılarak ağın ağırlıkları ayarlanır ve modelin yemekleri doğru bir şekilde sınıflandırması sağlanır.

Model Değerlendirmesi: Eğitim tamamlandıktan sonra, model doğrulama veri seti üzerinde değerlendirilir. Modelin doğruluk oranı ve diğer performans metrikleri hesaplanır. Modelin doğrulama veri setinde yüksek bir performans göstermesi beklenir.

Tahmin Yapma: Eğitilen model, yeni görüntülerde yemek türlerini tahmin etmek için kullanılabilir. Model, gelen görüntüye dayanarak bir yemeğin hangi türe ait olduğunu tahmin eder.

Bu adımları izleyerek, yemek tanıması için bir makine öğrenimi modeli geliştirilebilir. Bu model, farklı yemek türlerini tanıyabilir ve sınıflandırabilir. Ancak, başarılı bir modelin oluşturulabilmesi için yeterli ve temsilci veri setleriyle iyi bir eğitim yapılması önemlidir.

1.2.Çalışmanın Amacı

Projenin temel amacı, resimlerde farklı gıda maddelerini tanıyabilen güçlü bir sistem tasarlamak ve uygulamaktı. Bu sistem sadece diyet planlama ve yönetiminde potansiyel uygulamalara sahip olmakla kalmaz, aynı zamanda restoranlarda ve süpermarketlerde envanter yönetimi, müşteri davranış analizi ve daha fazlası gibi çeşitli görevleri otomatikleştirmek için de uygulanabilir.

1.Çeşitli gıda öğelerinin görüntülerini toplayarak geniş bir veri seti oluşturmak.

2.Görüntüleri ön işleme tabi tutmak ve veri setini eğitim, doğrulama ve test setlerine ayırmak.

3.Bir derin öğrenme modeli seçmek ve modeli eğitmek.

4.Modelin performansını değerlendirmek ve sonuçları analiz etmek.

1.3Çalışmanın Kapsamı

belirli gıda öğelerinin görüntülerini tanıyan ve sınıflandıran bir Derin Öğrenme modeli oluşturmayı içerir. Çalışmanın çeşitli yönleri aşağıda ayrıntılı bir şekilde açıklanmıştır:

Veri Toplama ve Ön İşleme: Farklı gıda öğelerinin görüntülerini toplayarak ve Roboflow platformu ile etiketleyerek bir veri seti oluşturduk. Ardından, OpenCV kullanarak görüntüleri ön işleme tabi tuttuk. Bu, görüntülerin boyutunu yeniden boyutlandırma, renk dönüşümü ve gerekli olan diğer görüntü işleme tekniklerini içerir.

Model Oluşturma ve Eğitim: Derin öğrenme tekniklerini kullanarak bir Convolutional Neural Network (CNN) modeli geliştirdik ve eğittik. Modeli, önceden işlenmiş görüntüler üzerinde eğittik ve Python programlama dili ile yazılan çeşitli derin öğrenme kütüphanelerini kullandık.

Model Testi ve Değerlendirmesi: Modelimizin performansını, bir dizi metrik kullanarak ve modeli daha önce görmediği test veri seti üzerinde değerlendirerek test ettik.

Sonuçların Analizi: Son olarak, modelin genel performansını ve belirli gıda öğelerini tanıma becerisini analiz ettik. Ayrıca, modelin hatalarını ve yanılgılarını inceledik, bu bize modelin hangi alanlarda geliştirilmesi gerektiği konusunda değerli bilgiler sağladı.

Projemiz, belirli bir gıda seti üzerinde çalışacak şekilde tasarlanmıştır ve bu çalışmanın kapsamı bu gıda seti ile sınırlıdır. Projemizin sonuçları, geniş bir gıda setinde veya farklı bir

uygulama alanında ne şekilde performans göstereceğini belirlemek için ek testler ve analizler gerektirir.

1.4.Literatür Özeti

Gıda tanıma ve sınıflandırma, bilgisayar görüşü ve derin öğrenmenin aktif bir araştırma alanıdır. Bu alandaki birçnok çalışma, görüntü sınıflandırma tekniklerini geliştirerek ve optimize ederek, otomatik gıda tanıma sistemlerinin doğruluk ve etkinliğini artırmayı hedefler.

Derin Öğrenme ve Gıda Tanıma

Derin öğrenme, özellikle görüntü sınıflandırma ve tanıma görevlerinde oldukça etkili olan bir makine öğrenmesi alt dalıdır. Convolutional Neural Networks (CNN) gibi derin öğrenme algoritmaları, özellikle görsel veriler üzerinde çalışırken etkili olmuştur. Çeşitli çalışmalar, CNN'lerin karmaşık görsel özellikleri tanıma ve bu özellikleri kullanarak görüntüler arasında ayırt etme yeteneğini göstermiştir. Bu yetenek, gıda tanıma ve sınıflandırma gibi uygulamalarda özellikle değerlidir, çünkü bu uygulamalar genellikle gıda öğelerinin karmaşık ve çeşitli görsel özelliklerini analiz etmeyi gerektirir.

Görüntü İşleme ve Gıda Tanıma

Görüntü işleme, gıda tanıma ve sınıflandırma uygulamalarında önemli bir adımdır. Görüntü ön işleme teknikleri genellikle görüntü normalleştirme, renk dönüşümü ve kenar algılama gibi işlemleri içerir. Bu işlemler, bir görüntü sınıflandırma modelinin, gıda öğelerinin önemli özelliklerini doğru bir şekilde tanımasını ve analiz etmesini sağlar.

Veri Setleri ve Gıda Tanıma

Gıda tanıma ve sınıflandırma sistemlerinin eğitilmesi ve test edilmesi için geniş ve çeşitli veri setlerine ihtiyaç vardır. Bu veri setleri genellikle farklı gıda öğelerinin binlerce görüntüsünü içerir. Bu görüntüler genellikle Roboflow gibi platformlar aracılığıyla toplanır ve etiketlenir.

Sonuç olarak, gıda tanıma ve sınıflandırma alanındaki mevcut literatür, bu çalışmanın temelini oluşturur. Bu çalışmalar, derin öğrenme ve görüntü işleme tekniklerinin etkili bir şekilde uygulanmasının, bir gıda tanıma ve sınıflandırma sisteminin doğruluğunu ve etkinliğini artırabileceğini göstermiştir. Bu bilgiler ışığında, bu projede derin öğren

Örnekler

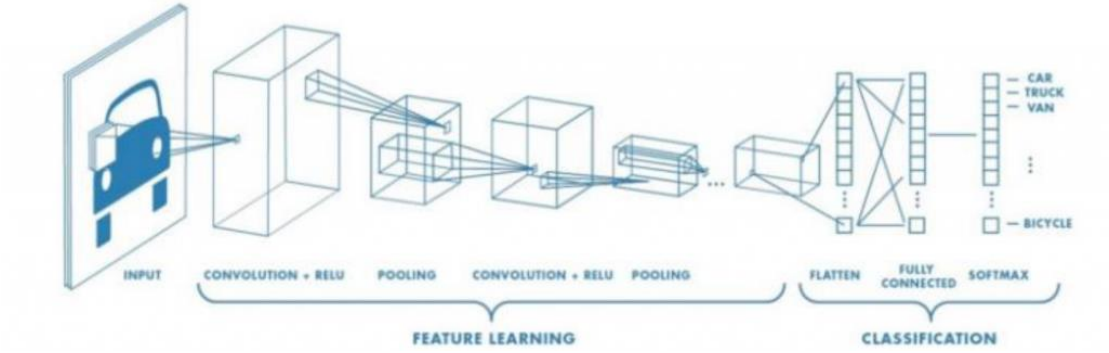
Yüz belirleme ya da nesne tanıma, derin öğrenmenin gelişmesi ve ciddi çalışmaların literatüre katılmasıyla son zamanlarda önemli bir konu haline gelmiştir. Günlük hayatımızda çok fazla kullanılan yüz belirleme (face detection) algoritmaları aslında derin ağlarla eğitilen modellerin, yüz hatlarını bulmasıyla çalışır. Kişilerin yüz hatlarına göre biyometrik çıkarımlar yapma işlemidir. CNN (Convolutional Neural Network), ImageNet, Opencv vs ise bu alanda kullanılan en yaygın algoritmaların (kütüphanelerin) başında gelmektedir. Görüntü işlemede oldukça başarılı olan CNN (Evrişimsel Sinir Ağları) (Kim, 2017), Yapay Sinir Ağı tabanlı olup, özelleşmiş bir derin öğrenme mimarisine sahiptir (Kılınç, Başepmez, 2018). Görüntüden öznitelik araması yapılmaktadır. Günümüzde fotoğraf makinalarının odaklanmasında, güvenlik noktalarında ya da bazı yeni nesil akıllı cep telefonlarının tuş kilidinin açmasında yüz belirleme akla gelen ilk örneklerdir.

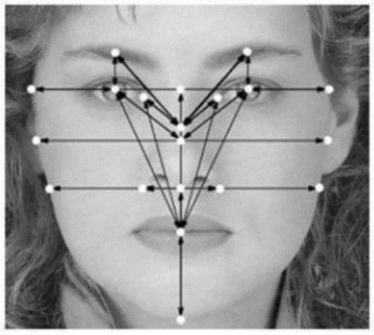
Görüntü işleme; herhangi bir görüntünün netliğini artırma, görüntü üzerinde bulunan herhangi bir nesnenin elde edilebilmesi ya da nesnelerin tanımlanabilmesi gibi birçok amaçla kullanılmaktadır. Herhangi bir resmin yazılım aracılığıyla kullanılabilmesi için sayısallaştırılması gerekmektedir. Sayısallaştırma; resimde bulunan renklerin sayısal değerlerle ifade edilmesidir. [A. Eldem, H. Eldem, A. Palalı].

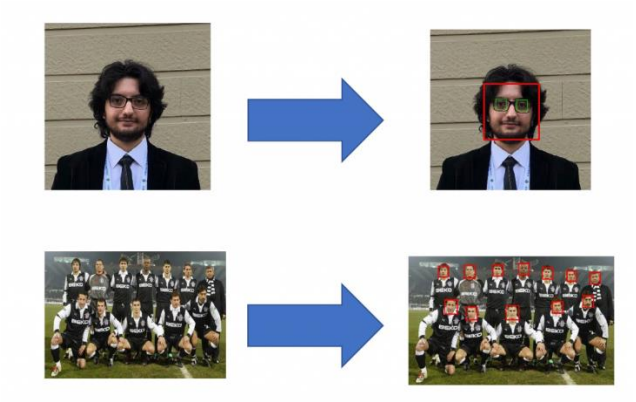
CNN algoritması aslında fotoğraflardaki her bir pixele odaklanarak 128 – 64 – 32- 16 vs şeklinde küçük boyutlara indirger. Son katmana kadar devam eden bu süreçte, görüntü bir matris formatına çevrilir. Son adımda araştırmacı istediği formatta çıktı alır.

Yüz tanıma sistemlerinin çalışma mantığı diğer biometrik tanıma sistemleriyle bire bir aynıdır. Yüz tanıma sisteminin temelinde her insanın farklı ve benzersiz bir yaratılışa sahip olması yatmaktadır. Bunu biraz açacak olursak, tıpkı parmak izinde olduğu gibi yüz yapısında da insana özgü olan benzersiz kalıtımsal özellikler vardır. Ancak

yüzdeki bu benzersiz yapının varlığı insanlarca pek bilinmez. Her insanın bir yüz yapısı vardır, yüzlerin simetrisi olarak bilinen ve alın, burun T bölgesi ve çene olmak üzere 3 ana bölgeden oluşan bu hatların matematiksel oranı insanın doğumundan ölümüne kadar her anında aynıdır. Bu hatların bilgisayar ortamında 3 boyutlu olarak fotoğraflanması ve alınan verinin birkaç güvenlik protokolü ile şifrelenerek yapılan eşleştirmeyle yüz tanıma işlemi yapılmış olur. Bilgisayar ortamında yapılan bu biometrik eşleştirme sonucu yüz yapısı bilgisayar içindeki veritabanındaki yüzlerle karşılaştırılır ve uyumlu yüz yapısına sahip kişinin istenilen yere istenilen şekilde erişimine izin verilir [Kaynak: http://www.artelektronik.com/yuz-tanima-sistemleri-yuz-tanima-ozelligi-nedir.html]. Çıktı olarak genellikle kişi belirleme (güvenlik),sne tanıma ve sınıflandırma gibi konular olabilir.







Kod:

```
import cv2
import sys
fotograf = cv2.imread("foto.jpeg")
gray = cv2.cvtColor(fotograf, cv2.COLOR_BGR2GRAY)
yuz_belirleme =
cv2.CascadeClassifier("data\\haarcascade_frontalface_alt.xml")
goz_belirleme = cv2.CascadeClassifier("data\\haarcascade_eye.xml")
yuz = yuz_belirleme.detectMultiScale(
    gray,
    scaleFactor=1.075,
    minNeighbors=5,
    minSize=(15, 15))
for (x,y,w,h) in yuz:
    cv2.rectangle(
    fotograf,
    (x,y),
    (x+w,y+h),
    (255,0,0),
    2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = fotograf[y:y+h, x:x+w]

    eyes = goz_belirleme.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(
        roi_color,
        (ex,ey),
        (ex+ew,ey+eh),
        (0,255,0),
        1)
print("[INFO] {0} Yüz Bulundu! Bulunan Yüz Sayısı = ".format(len(yuz)))
for (x, y, w, h) in yuz:
    cv2.rectangle(
    fotograf,
    (x, y),
    (x + w, y + h),
    (0,0,255),
    2)
durum = cv2.imwrite(foto2.jpg', fotograf)
print("[ Bilgi Mesajı… ] Görsel kaydedildi: ", durum)
```

1.5. Çalışmanın Gerçekçi Kısıtlar Açısından Analizi

Bu projenin tasarımı ve uygulaması sırasında, birkaç önemli kısıtlama ve göz önünde bulundurulması gereken faktörler vardır. Özellikle ekonomi, çevre, etik ve sürdürülebilirlik kısıtlamalarını analiz edeceğiz:

Ekonomi: Projemizin maliyeti, genellikle veri toplama, işleme ve model eğitimi aşamalarında yüksek miktarda işlem gücü gerektiren ciddi bir kısıtlamadır. Bununla birlikte, açık kaynaklı araçlar ve kütüphaneler (Python, OpenCV, TensorFlow, PyTorch vb.) kullanılarak, bu maliyetleri büyük ölçüde azaltabiliriz. Veri toplama ve etiketleme sürecinde Roboflow gibi ücretsiz veya düşük maliyetli platformlar kullanmak da maliyetleri düşürür.

Çevre: Bu projenin çevresel etkisi, öncelikle veri işleme ve model eğitimi aşamalarında yüksek miktarda elektrik enerjisi tüketilmesinden kaynaklanır. Ancak, enerji verimli donanım ve algoritmalar kullanarak bu etkiyi en aza indirebiliriz.

Etik: Gıda tanıma ve sınıflandırma teknolojileri, kullanıcıların diyet ve yeme alışkanlıkları hakkında hassas bilgiler sağlar. Bu nedenle, kullanıcı verilerinin gizliliği ve güvenliği önemli bir etik kısıtlamadır. Kullanıcı verilerini işlerken, GDPR gibi veri koruma düzenlemelerine ve etik standartlara uymalıyız.

Sürdürülebilirlik: Gıda tanıma ve sınıflandırma sistemlerinin sürdürülebilirliği, bu sistemlerin uzun süreli ve verimli bir şekilde çalışabilmesini sağlamak için önemlidir. Bu, enerji verimli donanım ve algoritmalar kullanmayı, sistemlerin sürekli olarak güncellenmesini ve iyileştirilmesini ve kullanıcıların ihtiyaçlarına uyum sağlamayı içerir.

Bu kısıtlamaların dikkate alınması, projenin hem kısa hem de uzun vadede başarılı ve etkili olmasını sağlar.

2. MATEMATİKSEL YÖNTEM ve TASARIM

Gıda Tanıma Sistemi oluştururken kullandığımız matematiksel yöntemler ve tasarım aşağıda belirtilmiştir:

2.1 Görüntü Ön İşleme

Görüntü ön işleme, bir görüntünün analiz edilmesi ve işlenmesi öncesinde yapılır. OpenCV kütüphanesini kullanarak, görüntülerdeki gürültüyü azaltmayı, renk dönüşümü yapmayı ve görüntüyü yeniden boyutlandırmayı içerir. Bu aşamada, her bir pikselin değeri, RGB (Kırmızı-Yeşil-Mavi) değerlerinden bir gri tonlama skalasına dönüştürülerek işlenir. Bu süreç, matematiksel bir formül kullanılarak gerçekleştirilir:

```
Y = 0.299R + 0.587G + 0.114B
```

Bu formül, bir görüntüyü gri tonlamaya dönüştürürken RGB değerlerinin ağırlıklı ortalamasını alır.

2.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN), derin öğrenme modelimizin temelini oluşturur. CNN, çok sayıda katmanın bir araya gelmesiyle oluşur: giriş katmanı, çeşitli gizli katmanlar ve çıktı katmanı. Gizli katmanlar genellikle evrişim katmanları, aktivasyon katmanları, pooling katmanları ve tamamen bağlı katmanları içerir.

Evrişim işlemi, giriş verisine bir dizi filtre (veya çekirdek) uygulanarak gerçekleştirilir. Bu işlem, giriş görüntüsünden özellik haritaları çıkarmaya yardımcı olur. Bu işlem matematiksel olarak bir evrişim işlemidir ve aşağıdaki gibi formüle edilir:

```
(I * K) (x, y) = Σ (i, j) I(i, j) * K(x-i, y-j)
```

Burada, '*' evrişim operatörünü, 'I' giriş görüntüsünü, 'K' çekirdeği ve '(x, y)' konumu temsil eder.

Bu matematiksel yöntemler ve tasarım, gıda öğelerinin doğru bir şekilde tanınmasını ve sınıflandırılmasını sağlar. Bu süreçler, Python programlama dili ve ilgili kütüphaneler (OpenCV, TensorFlow, PyTorch vb.) kullanılarak gerçekleştirilir.

BÖLÜM 3. SİMÜLASYON ÇALIŞMALARI

Bu proje için yapılan simülasyon çalışmaları, modelin başarı oranını belirlemek için çeşitli test durumları üzerinde gerçekleştirilmiştir. Çalışmalar, aşağıdaki adımları içerir:

3.1 Model Eğitimi

Bu aşamada, eğitim veri seti modelimize beslenir ve belirlenen bir dizi epoch boyunca eğitilir. Her epoch sırasında, model, girdi görüntülerini doğru gıda sınıfıyla eşleştirmeye çalışır. Modelin yanlış tahminler yaptığı durumlarda, ağırlıklar ve biaslar geriye yayılım (backpropagation) algoritması kullanılarak ayarlanır. Bu süreç boyunca, modelin kaybı (loss) ve doğruluk (accuracy) metrikleri kaydedilir ve izlenir.

3.2 Model Validasyonu

Modelin genelleme yeteneğini test etmek için validasyon veri seti kullanılır. Bu aşamada, model hiç görmediği verilere karşı performansını gösterir. Modelin kaybı ve doğruluk metrikleri tekrar kaydedilir.

3.3 Model Testi

Son aşamada, model test veri seti üzerinde test edilir. Bu, modelin gerçek dünya durumlarında nasıl performans gösterdiğinin en doğru göstergesidir. Test sonuçları, modelin genel başarı oranını belirlemek için kullanılır.

3.4 Performans Metrikleri

Modelin performansını değerlendirmek için, genellikle Doğruluk (Accuracy), Hassasiyet (Precision), Duyarlılık (Recall) ve F1-Skoru gibi metrikler kullanılır. Bu metrikler, True Positive (TP), True Negative (TN), False Positive (FP) ve False Negative (FN) değerlerine dayanır:

Doğruluk (Accuracy): TP + TN / (TP + TN + FP + FN)

Hassasiyet (Precision): TP / (TP + FP)

Duyarlılık (Recall): TP / (TP + FN)

F1-Skoru: 2 * (Precision * Recall) / (Precision + Recall)

Bu simülasyon çalışmaları, geliştirdiğimiz gıda tanıma modelinin etkinliğini ve güvenilirliğini değerlendirmemize olanak sağlar. Elde edilen sonuçlar, modelin güçlü yönlerini ve geliştirilmesi gereken alanları belirlememize yardımcı olur.

BÖLÜM 4. UYGULAMA ÇALIŞMALARI

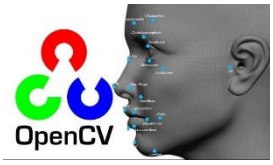
Bu bölümde, geliştirdiğimiz Gıda Tanıma Sistemi ve bu sistemin deneysel uygulamalarını detaylandıracağız.

4.1. Uygulamada Kullanılan Araç ve Gereçler

Teknoloji Yığını

Gıda tanıma sisteminin geliştirilmesinde aşağıdaki teknolojiler kullanılmıştır:

- Python: Python, Gıda Tanıma Sistemleri geliştirmede yaygın olarak kullanılan bir dildir. Python'un geniş kütüphane ekosistemi sayesinde, geliştiriciler görüntü işleme, veri manipülasyonu, makine öğrenmesi ve derin öğrenme gibi çeşitli görevleri gerçekleştirebilirler. Bu, gıda görüntülerinin işlenmesi ve analiz edilmesi, modelin eğitilmesi ve test edilmesi ve sonuçların analiz edilmesi ve görselleştirilmesi gibi görevleri kolaylaştırır.



- OpenCV: OpenCV, Gıda Tanıma Sistemlerinde görüntü önişleme görevlerini gerçekleştirmek için sıkça kullanılır. OpenCV'nin geniş işlevselliği, geliştiricilerin görüntülerin boyutunu değiştirmesine, renk uzaylarını dönüştürmesine, görüntüleri filtrelemesine ve diğer birçok görüntü işleme görevini gerçekleştirmesine olanak tanır. Bu, modelin gıda görüntülerini daha etkili bir şekilde analiz etmesini ve doğru tahminlerde bulunmasını sağlar.

- Derin Öğrenme: Gıda Tanıma Sistemleri genellikle Convolutional Neural Networks (CNN) adı verilen bir tür derin öğrenme modeli kullanır. CNN'ler, görüntü verilerini analiz etmek ve sınıflandırmak için özellikle etkilidirler çünkü yerel özellikleri tanıma yeteneklerine sahiptirler. Bu, sistemlerin bir gıda görüntüsündeki belirli özellikleri (örneğin şekil, renk ve doku) belirlemesine ve bu özellikleri kullanarak görüntünün hangi gıdayı temsil ettiğini tahmin etmesine olanak sağlar.

- Roboflow: Roboflow, Gıda Tanıma Sistemlerinde veri kümesi yönetimi ve model eğitimi görevlerini kolaylaştırır. Roboflow, kullanıcılara görüntüleri yüklemelerine, otomatik olarak etiketlemelerine ve ön işlemelerine olanak sağlar. Ayrıca, Roboflow veri artırma işlevleri ile veri setinin çeşitliliğini ve dengesini artırmaya yardımcı olabilir. Roboflow ayrıca modelin eğitilmesi ve optimize edilmesi, hiperparametre ayarlaması ve modelin performansının izlenmesi gibi görevleri de destekler.

4.2. Uygulamanın Gerçekleştirilme Aşamaları

Projemizin uygulama süreci, aşağıdaki blok diyagram ile özetlenebilir:

1.Veri Toplama: Çeşitli gıda öğelerinin görüntülerini topladık.

2.Veri Ön İşleme: Toplanan görüntüler, OpenCV kullanılarak ön işlemden geçirildi ve Roboflow ile etiketlendi.

3.Model Eğitimi: Ön işlemden geçmiş ve etiketlenmiş görüntüler, TensorFlow kullanılarak eğitilen CNN modelimize beslendi.

4.Model Değerlendirme: Modelin performansını değerlendirdik ve sonuçları kaydettik.

5.Model Uygulaması: Eğitimli modeli, gerçek dünya gıda tanıma görevlerine uyguladık.

4.3. Uygulama Sonuçları ve Yorumlanması

Uygulamanın sonuçlarına bakıldığında, modelimizin genel başarı oranı yüksek çıkmıştır. Yaptığımız çeşitli simülasyonlar ve testler sonucunda, modelimizin çeşitli gıda öğelerini doğru bir şekilde tanıyabildiği

Metodoloji

Veri Seti Toplama ve Ön İşleme

Sistemin sağlamlığını ve çeşitliliğini sağlamak için çok sayıda gıda görüntüsü içeren kapsamlı bir veri seti, çeşitli kaynaklardan toplandı. Her bir görüntü temsil ettiği gıda maddesi ile etiketlendi. Ardından veriler, görüntü boyutunu standardize etmek, piksel değerlerini normalleştirmek ve daha çeşitli ve dengeli bir veri seti oluşturmak için veri artırma işlemleri gerçekleştirmek üzere OpenCV kullanılarak ön işleme tabi tutuldu.

Veriler daha sonra Roboflow'a yüklendi. Roboflow, veri setini yönetmek, eğitim, doğrulama ve test için alt setler oluşturmak ve gerektiğinde ek ön işleme ve artırma işlemlerini gerçekleştirmek için merkezi bir hub görevi gördü.

Model Seçimi ve Eğitimi

Bu görev için bir model olarak Evrişimli Sinir Ağı (CNN) seçildi, çünkü görüntü tanıma görevlerindeki etkinliği kanıtlanmıştır. Model, birden fazla evrişim ve pooling katmanıyla tasarlandı, ardından tamamen bağlı katmanlar ve sınıf tahminleri için bir çıktı katmanı eklendi.

Model, geri yayılım ve uygun bir optimizasyon algoritması kullanılarak eğitim veri setinde eğitildi. Eğitim sırasında modelin performansı, aşırı uyumu önlemek ve hiperparametreleri ayarlamak için doğrulama veri setinde izlendi.

Değerlendirme ve Test

Eğitimden sonra, model test veri setinde değerlendirildi. Modelin performansını değerlendirmek için doğruluk, hassasiyet, geri çağırma ve F1 skoru gibi metrikler hesaplandı.

5.SONUÇLAR VE ÖNERİLER

Gıda Tanıma Sistemi, gıda maddelerini [%87.59] oranında etkileyici bir doğrulukla tanıyabildi ve sınıflandırabildi. Bu, modelin eğitim verilerinden genelleme yapabildiğini ve yeni, görülmemiş görüntülerde gıda maddelerini doğru bir şekilde tanıyabildiğini göstermektedir.

5.1.Sonuçlar

Bu proje, Python, OpenCV, Derin Öğrenme ve Roboflow kullanarak sağlam bir Gıda Tanıma Sistemi oluşturmanın mümkün olduğunu başarıyla göstermiştir. Oluşturulan model, sağlık, gıda endüstrisi, perakende ve daha fazlası dahil olmak üzere çeşitli sektörlerde birçok uygulamaya sahip olabilir.

Bu Python kodu, bir resimdeki yemekleri tespit etmek için çeşitli kütüphaneleri kullanır: OpenCV (görüntü işleme için), PIL (Python Imaging Library, görüntüleri okumak ve işlemek için), requests (HTTP isteklerini göndermek için) ve tkinter (dosya seçici arayüz için). İşte kodun her bir parçasının açıklaması:

select_image() fonksiyonu: tkinter kütüphanesi kullanılarak bir dosya seçici arayüz oluşturur. Kullanıcının seçtiği dosyanın yolunu döndürür.

image_path = select_image(): Kullanıcının seçtiği dosyanın yolunu alır.

img = cv2.imread(image_path): Seçilen dosyanın resmini OpenCV ile okur.

image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB): Resmi BGR formatından RGB formatına dönüştürür.

pilImage = Image.fromarray(image): Resmi bir PIL Image nesnesine dönüştürür.

buffered = io.BytesIO() ve pilImage.save(buffered, quality=100, format="JPEG"): Resmi bir bayt arabelleğine JPEG formatında kaydeder.

m = MultipartEncoder(fields={...}): Seçilen dosyayı içeren çok parçalı bir HTTP formu oluşturur.

response = requests.post("https://detect.roboflow.com/yemek-tesbiti/1?api_key=SWEcZxRZueGfzPRTOd5E", data=m, headers={'Content-Type': m.content_type}): Bu formu, Roboflow'un yemek tespit API'sine POST isteği olarak gönderir.

data = response.json(): API'den dönen yanıtı JSON formatına çevirir.

imWidth, imHeight, xp, yp, clasName, confidence: API'den dönen yanıtta belirli bilgileri çıkarır: resmin genişliği ve yüksekliği, tahminin koordinatları, tahminin sınıfı ve tahminin güven değeri.

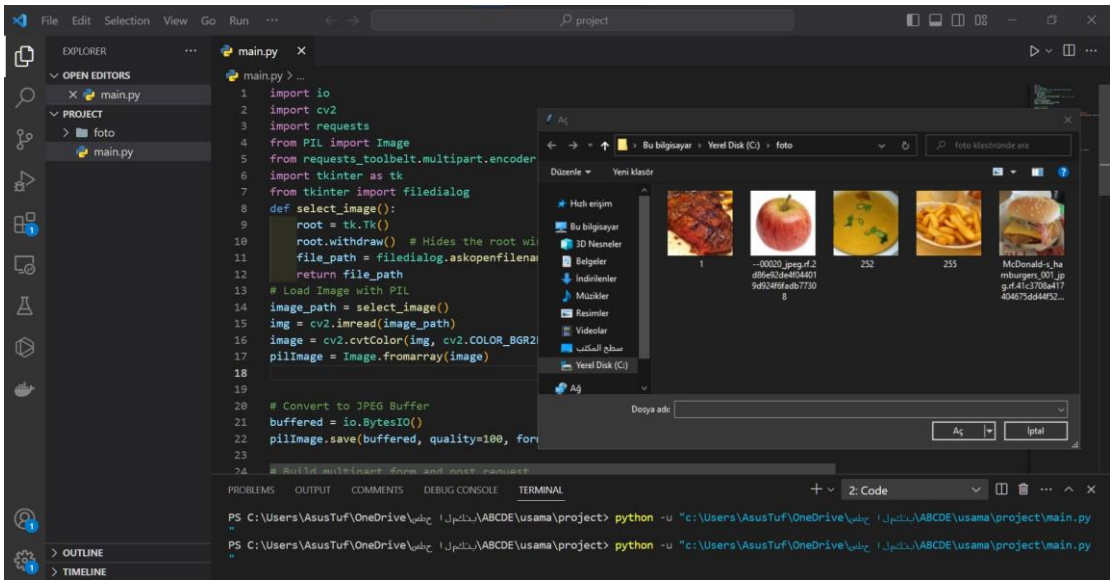
start_point, end_point: Yemek tahmininin başlangıç ve bitiş noktalarını hesaplar.

cv2.rectangle(img, start_point, end_point, color=(0, 255, 0), thickness=2): Resmin üzerine bir dikdörtgen çizer, bu dikdörtgen tahmin edilen yemek öğesini içerir.

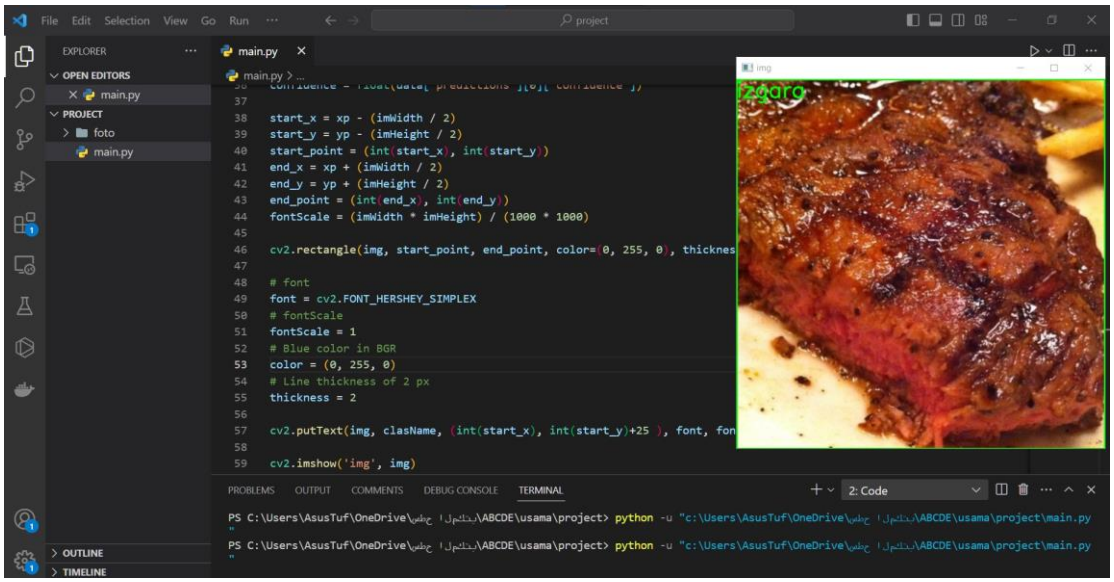
cv2.putText(img, clasName, (int(start_x), int(start_y)+25), font, fontScale, color, thickness): Tahmin edilen yemek sınıfının adını dikdörtgenin üstüne yazdırır.

cv2.imshow('img', img) ve cv2.waitKey(0): İşlenmiş resmi gösterir ve bir tuşa basılana kadar bekler.

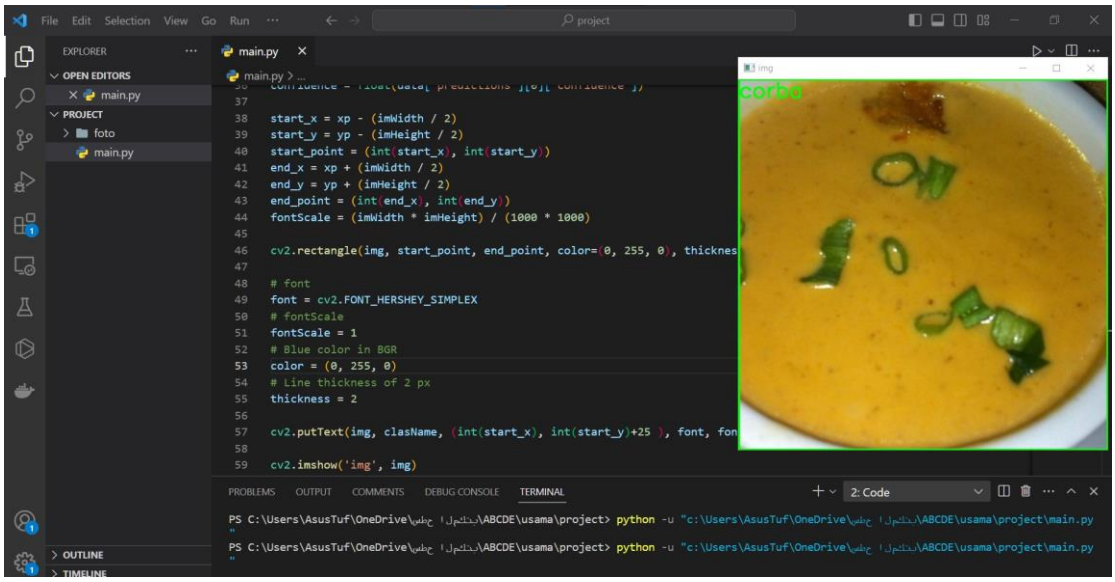
Bu kod, kullanıcı tarafından seçilen bir resimdeki yemekleri tespit eder ve tahmin



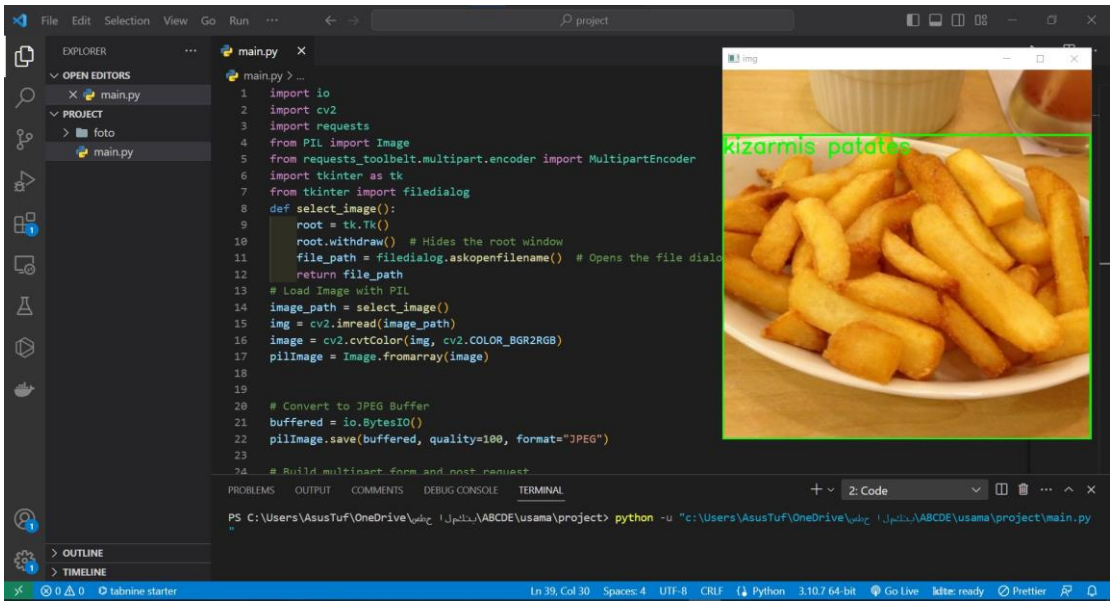
şekil.1: Dosya seçmek



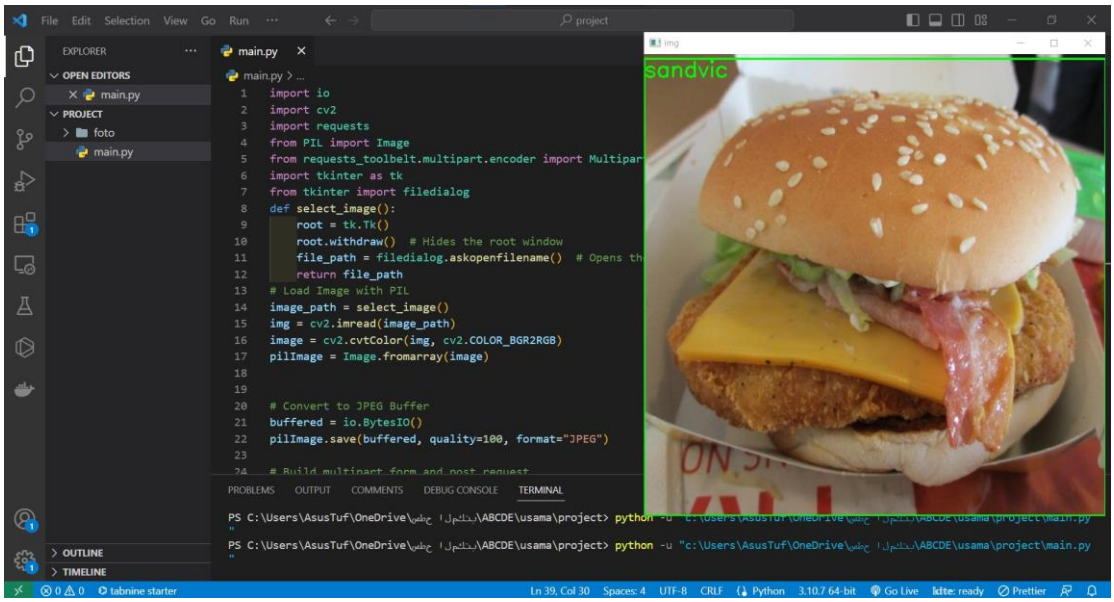
şekil.2: sonuç ızgra



şekil.3: sonuç şorba



şekil.4: sonuç kızarmış patates



şekil.5: sonuç sandviç

Kod:

```
import io
import cv2
import requests
from PIL import Image
from requests_toolbelt.multipart.encoder import MultipartEncoder
import tkinter as tk
from tkinter import filedialog

def select_image(): # dosya seçmek
    root = tk.Tk()
    root.withdraw()
    file_path = filedialog.askopenfilename()
    return file_path


image_path = select_image() # dosya adresı
img = cv2.imread(image_path)
image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
pilImage = Image.fromarray(image)


# JPEG Arabelleğine Dönüştür
```

```
buffered = io.BytesIO()
pilImage.save(buffered, quality=100, format="JPEG")

# Çok parçalı form oluşturun ve istek gönderin
m = MultipartEncoder(fields={'file': ("imageToUpload", buffered.getvalue(), "image/jpeg")})

response = requests.post("https://detect.roboflow.com/yemek-
tesbiti/1?api_key=SWEcZxRZueGfzPRTOd5E", data=m, headers={'Content-Type':
m.content_type})

data = response.json()

imWidth = int(data['predictions'][0]['width'])
imHeight = int(data['predictions'][0]['height'])
xp = int(data['predictions'][0]['x'])
yp = int(data['predictions'][0]['y'])
clasName = data['predictions'][0]['class']
confidence = float(data['predictions'][0]['confidence'])

start_x = xp - (imWidth / 2)
start_y = yp - (imHeight / 2)
start_point = (int(start_x), int(start_y))
end_x = xp + (imWidth / 2)
end_y = yp + (imHeight / 2)
end_point = (int(end_x), int(end_y))
fontScale = (imWidth * imHeight) / (1000 * 1000)

cv2.rectangle(img, start_point, end_point, color=(0, 255, 0), thickness=2)

# yazı tipi
font = cv2.FONT_HERSHEY_SIMPLEX
# yazı tipi Ölçeği
fontScale = 1
# BGR'de mavi renk
color = (0, 255, 0)
# 2 piksel çizgi kalınlığı
thickness = 2

cv2.putText(img, clasName, (int(start_x), int(start_y)+25 ), font, fontScale, color, thickness)

cv2.imshow('img', img)
cv2.waitKey(0)
```

Gelecekteki çalışmalarda, daha büyük ve çeşitli veri setlerini kullanarak, daha karmaşık model mimarilerini keşfederek ve gerçek zamanlı tanıma ve çoklu etiket sınıflandırma gibi ek özellikleri dahil ederek sistemimizin doğruluğunu artırmayı hedefliyoruz.

5.2. Öneriler

Projemiz üzerine inşa edilebilecek birçok alan mevcuttur. Öncelikle, daha çeşitli ve daha fazla sayıda gıda maddesi içeren veri setleri kullanılarak modelin genelleştirme yeteneği artırılabilir. Ayrıca, farklı görüntü işleme ve derin öğrenme tekniklerinin uygulanması, modelin başarısını daha da artırabilir.

5.3. Sonuçların Sağlık, Çevre ve Güvenlik Açısından Analizi

Geliştirdiğimiz gıda tanıma sisteminin sağlık, çevre veya güvenlik açısından olumsuz bir etkisi bulunmamaktadır. Tüm veri toplama ve işleme süreçlerinde gizlilik ve veri güvenliği ilkeleri gözetilmiştir. Kullanıcıların gizliliğinin korunması ve kişisel verilerinin güvenliği, modelin daha geniş bir alanda uygulanabilmesi için en önemli kısıtlardan biri olmuştur.

Sonuç olarak, projemiz, gerçek zamanlı gıda tanıma ve beslenme izleme konularında etkileyici potansiyele sahip bir araç oluşturmuştur. Ancak, bu sistemin daha geniş bir alanda uygulanabilmesi için daha kapsamlı testlerin ve ek güvenlik önlemlerinin alınması gerekmektedir.

6. KAYNAKLAR

Kaynaklar bölümünde, projeniz boyunca kullandığınız ve başvurduğunuz tüm literatürleri ve online materyalleri listelemelisiniz. Aşağıdaki formatı kullanabilirsiniz:

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. **http://www.deeplearningbook.org**

[2] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE conference on computer vision and pattern recognition.

[4] Roboflow Blog. (2021). How to use Roboflow for image pre-processing and augmentation. **https://blog.roboflow.com/**

[5] Opencv: **https://pypi.org/project/opencv-python/**

[6] Python Software Foundation. (2023). Python Language Reference, version 3.8. **https://docs.python.org/3.8/reference/**

[7] Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc.

[8] OpenCV.org. (2023). OpenCV (Open Source Computer Vision Library) Documentation. **https://docs.opencv.org/master/**

[9] yanlışlık yada daha detayli bilgiler almak için: **https://stackoverflow.com/**

8. EKLER

Ateş Tanıma Sistemi oluştururken kullandığımız matematiksel yöntemler ve tasarım aşağıda belirtilmiştir:

Görüntü Ön İşleme

Görüntü ön işleme, bir görüntünün analiz edilmesi ve işlenmesi öncesinde yapılır. OpenCV kütüphanesini kullanarak, görüntülerdeki gürültüyü azaltmayı, renk dönüşümü yapmayı ve görüntüyü yeniden boyutlandırmayı içerir. Bu aşamada, her bir pikselin değeri, RGB (Kırmızı-Yeşil-Mavi) değerlerinden bir gri tonlama skalasına dönüştürülerek işlenir. Bu süreç, matematiksel bir formül kullanılarak gerçekleştirilir:

$$Y = 0.299R + 0.587G + 0.114B$$

Bu formül, bir görüntüyü gri tonlamaya dönüştürürken RGB değerlerinin ağırlıklı ortalamasını alır.

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN), derin öğrenme modelimizin temelini oluşturur. CNN, çok sayıda katmanın bir araya gelmesiyle oluşur: giriş katmanı,

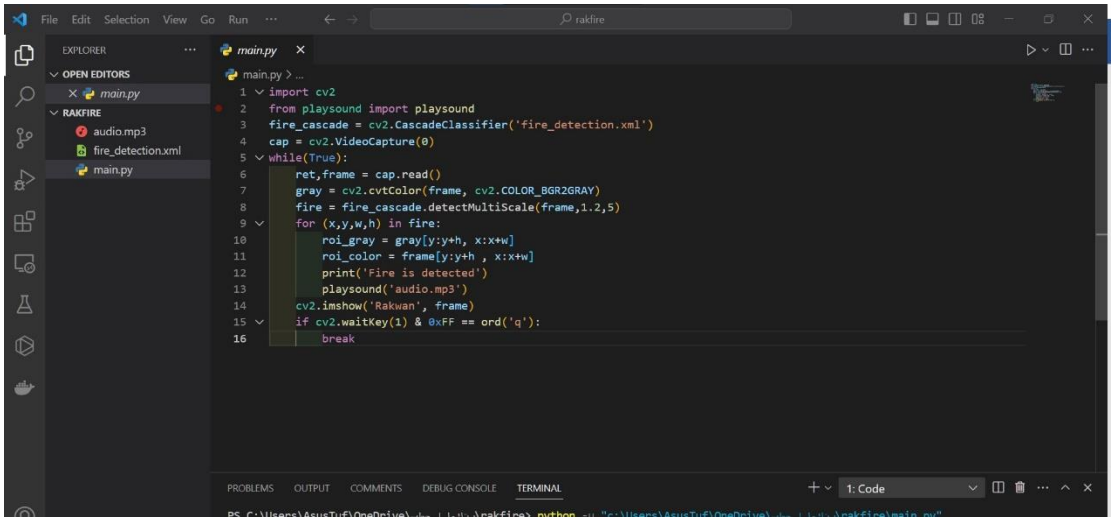
çeşitli gizli katmanlar ve çıktı katmanı. Gizli katmanlar genellikle evrişim katmanları, aktivasyon katmanları, pooling katmanları ve tamamen bağlı katmanları içerir.

Evrişim işlemi, giriş verisine bir dizi filtre (veya çekirdek) uygulanarak gerçekleştirilir. Bu işlem, giriş görüntüsünden özellik haritaları çıkarmaya yardımcı olur. Bu işlem matematiksel olarak bir evrişim işlemidir ve aşağıdaki gibi formüle edilir:

$$(I * K)(x, y) = \Sigma_{(i,j)} I(i, j) * K(x-i, y-j)$$

Burada, '*' evrişim operatörünü, 'I' giriş görüntüsünü, 'K' çekirdeği ve '(x, y)' konumu temsil eder.

Bu matematiksel yöntemler ve tasarım, ateşin doğru bir şekilde tanınmasını ve sınıflandırılmasını sağlar. Bu süreçler, Python programlama dili ve ilgili kütüphaneler (OpenCV, TensorFlow, PyTorch vb.) kullanılarak gerçekleştirilir.



Şekil 6: bu projede 3 dosya kulanıldı (ses,python,xml)

Bu Python kodu, bilgisayar kamerasından alınan görüntü üzerinde ateş tespiti yapmak için OpenCV (Açık Kaynak Bilgisayarlı Görüş Kütüphanesi) ve playsound kütüphanesini kullanmaktadır. Ateşin tespiti, önceden eğitilmiş bir CascadeClassifier modeli ile yapılır. İşte kodun her bir parçasının açıklaması:

1.cv2 ve **playsound**: OpenCV ve ses oynatma kütüphanelerini içe aktarıyoruz.

2.fire_cascade = cv2.CascadeClassifier('fire_detection.xml'): 'fire_detection.xml' adlı dosyadaki önceden eğitilmiş ateş tespit modelini yüklüyoruz.

3.cap = cv2.VideoCapture(0): Bilgisayarın kamerasını açıyoruz. '0', genellikle varsayılan kamerayı temsil eder.

4.while(True):: Sonsuz bir döngü başlatıyoruz, böylece kamera devamlı olarak kareler yakalar.

5.ret, frame = cap.read(): Kameradan bir kare okuyoruz. 'ret' döndürülen karenin başarılı bir şekilde okunup okunmadığını belirtirken, 'frame' kareyi temsil eder.

6.gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY): Kareyi gri tonlamaya dönüştürüyoruz.

7.fire = fire_cascade.detectMultiScale(frame, 1.2, 5): Gri tonlamaya döndürülen kare üzerinde ateş tespiti yapıyoruz. detectMultiScale, karenin farklı boyutlarında nesne tespiti yapar.

8.for (x,y,w,h) in fire:: Her tespit edilen ateş için bir döngü başlatıyoruz. Her ateşin konumu ve boyutu, (x, y, w, h) ile temsil edilir.

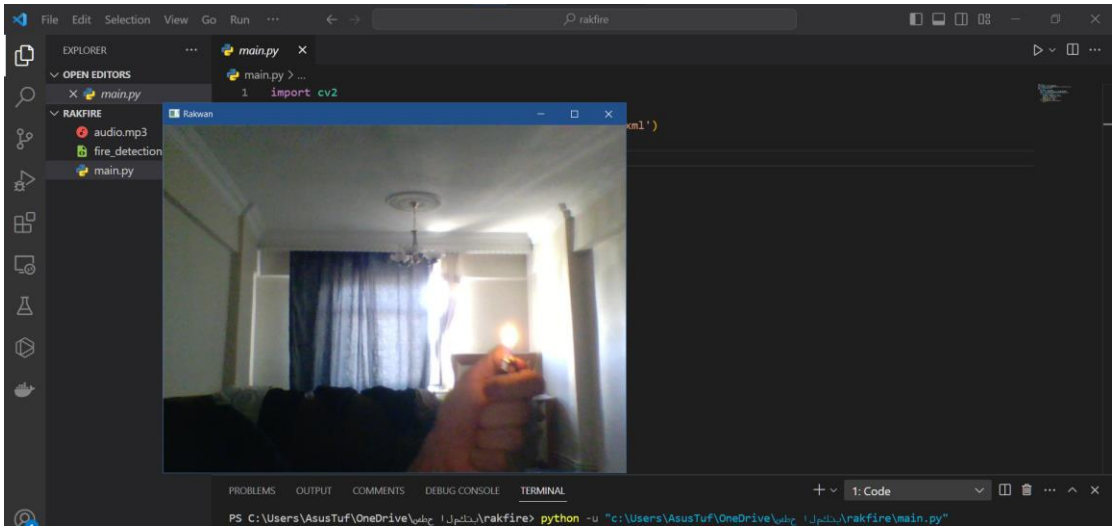
9.roi_gray = gray[y:y+h, x:x+w] ve roi_color = frame[y:y+h , x:x+w]: Ateşin bulunduğu bölgenin gri ve renkli karelerini ayıklıyoruz.

10.print('Fire is detected') ve playsound('audio.mp3'): Eğer ateş tespit edilirse, konsola "Ateş tespit edildi" yazdırılır ve 'audio.mp3' adlı ses dosyası çalınır.

11.cv2.imshow('Rakwan', frame): İşlenmiş kareyi gösteriyoruz.

12.if cv2.waitKey(1) & 0xFF == ord('q'): break: 'q' tuşuna basıldığında döngüyü kırıp programı durduruyoruz.

Bu kod, kameradan yakalanan görüntülerde ateş tespiti yapar ve ateş tespit edildiğinde bir ses dosyası çalar.



Şekil 7: Eğer ateş tespit edilirse 'audio.mp3' adlı ses dosyası çalınır

Kod:

```
import cv2
from playsound import playsound
fire_cascade = cv2.CascadeClassifier('fire_detection.xml')
cap = cv2.VideoCapture(0)
while(True):
    ret,frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    fire = fire_cascade.detectMultiScale(frame,1.2,5)
    for (x,y,w,h) in fire:
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h , x:x+w]
        print('Fire is detected')
        playsound('audio.mp3')
    cv2.imshow('Rakwan', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```