Nesne Tespiti İçin YOLO Tabanlı Derin Öğrenme Uygulaması

Halil İbrahim Dürmüş

Özet

Bu çalışmada, gerçek zamanlı nesne tespiti amacıyla YOLOv8 tabanlı bir yapay zeka modeli eğitilmiş ve bu model bir web arayüzü aracılığıyla kullanıma sunulmuştur. Veri seti, araştırmacı tarafından toplanan ve etiketlenen görüntülerden oluşturulmuştur. Bu görüntüler, çeşitli ön işleme ve veri artırma teknikleri kullanılarak genişletilmiş ve daha sağlam bir model eğitimi sağlanmıştır. Eğitim sonucu elde edilen model, Flask ile geliştirilen bir API üzerinde barındırılmış ve Angular tabanlı kullanıcı arayüzünden gelen istekleri işleyerek nesne tespiti gerçekleştirmiştir. Sonuçlar, modelin belirli nesneleri başarıyla tanıyabildiğini ve gerçek zamanlı görüntülerde doğru sınıflandırmalar yapabildiğini göstermektedir.

Anahtar Kelimeler: YOLO, nesne tespiti, derin öğrenme, Flask, Angular, gerçek zamanlı görüntü işleme

1. Giriş

Görüntü işleme, bilgisayarla görme (computer vision) alanının önemli bir parçasıdır ve son yıllarda yapay zeka (AI) ile birleşerek önemli bir ilerleme kaydetmiştir. Yapay zeka algoritmalarının gelişmesiyle birlikte, bilgisayarlar artık insanlar gibi görsel veriler üzerinde analiz yapabilmekte, nesneleri tanıyabilmekte ve etkileşime girebilmektedirler. Bu alandaki en yaygın uygulamalardan biri, nesne tespitidir. Nesne tespiti, bir görüntü içinde belirli nesnelerin yerini ve sınıfını (kategori) tespit etme işlemidir. Nesne tespiti, otonom araçlardan sağlık uygulamalarına, güvenlik sistemlerinden robotik sistemlere kadar geniş bir yelpazede kullanılmaktadır.

Son yıllarda derin öğrenme (deep learning) teknikleri, nesne tespiti alanında önemli bir devrim yaratmıştır. Özellikle YOLO (You Only Look Once) algoritması, nesne tespiti konusunda hızlı ve doğru sonuçlar elde etmesi ile dikkat çekmektedir. YOLO, görüntüdeki tüm nesneleri tek bir adımda tespit edebilmesi sayesinde, diğer geleneksel algoritmalara

kıyasla çok daha hızlı sonuçlar elde etmektedir. Bu özellik, YOLO'yu özellikle gerçek zamanlı uygulamalarda kullanmak için ideal hale getirmektedir.

Bu çalışmanın amacı, Angular tabanlı bir web arayüzü ile kullanıcıdan alınan görüntüleri işleyerek nesne tespiti yapmaktır. Flask tabanlı bir API, Roboflow üzerinde eğitilmiş YOLO modelini kullanarak görselleri analiz eder ve sonuçları kullanıcının arayüzüne ileterek tespit edilen nesneleri görselleştirir. Bu sayede, kullanıcılar yükledikleri görüntülerdeki nesneleri hızlı bir şekilde tespit edebilmekte ve bu bilgiler görsel olarak geri döndürülmektedir.

Bu çalışmada kullanılan temel teknolojiler şunlardır:

- **Angular**: Web arayüzü için modern JavaScript framework'ü.
- **Flask**: Python tabanlı, hafif bir web framework'ü ile API geliştirme.
- **YOLO (You Only Look Once)**: Derin öğrenme tabanlı nesne tespit algoritması.
- **Roboflow**: Etiketlenmiş veri setleri ile model eğitimi ve yönetimi için bir platform.

Bu çalışmanın sonunda, geliştirilen sistemin verimliliği değerlendirilecek ve gelecekte yapılabilecek iyileştirmelere yönelik önerilerde bulunulacaktır.

2.Literatür Taraması

Görüntü işleme ve nesne tespiti, bilgisayarla görme (computer vision) alanındaki en önemli araştırma alanlarından biridir. Nesne tespiti, görüntülerdeki belirli nesnelerin konumlarını ve sınıflarını doğru bir şekilde belirlemeye yönelik bir tekniktir. Bu alanda yapılan çalışmalar, derin öğrenme algoritmalarının uygulanmasıyla önemli ilerlemeler kaydetmiştir. Özellikle, YOLO (You Only Look Once) algoritması, nesne tespiti alanında hızlı ve doğru sonuçlar elde etmesi nedeniyle geniş bir kullanım alanına sahiptir.

2.1.Nesne Tespiti Yöntemleri

Nesne tespiti için ilk başlarda geleneksel yöntemler, özellik tabanlı yaklaşımlar kullanıyordu. Örneğin, **Haar Cascade** ve **HOG (Histogram of Oriented Gradients)** gibi yöntemler, nesne tespiti için özelliklerin çıkarılmasına dayanıyordu. Ancak bu yöntemler, sınırlı doğruluk ve düşük işlem hızına sahipti.

Derin öğrenme tekniklerinin gelişmesiyle birlikte, **Convolutional Neural Networks (CNN)** nesne tespitinde önemli bir yer edinmiştir. CNN'ler, görüntülerin özelliklerini öğrenme ve çıkarma konusunda insan benzeri başarılar elde etmiştir. Bu yöntem, özellikle **Region-based CNN (R-CNN)** ve **Fast R-CNN** gibi modellerle nesne tespitinde önemli adımlar atmıştır. Ancak bu modeller, genellikle düşük işlem hızı ve verimsiz işlem süreleri ile sınırlıdır.

2.2.YOLO Modeli

YOLO (You Only Look Once), 2015 yılında Joseph Redmon ve arkadaşları tarafından geliştirilmiştir ve nesne tespitinde devrim yaratmıştır. YOLO'nun en önemli özelliği, bir görüntüyü tek bir aşamada işleyerek nesneleri tespit edebilmesidir. Bu, geleneksel nesne

tespiti yöntemlerine kıyasla çok daha hızlı sonuçlar alınmasını sağlar. YOLO, tam görüntüyü bir ağ üzerinden geçirecek şekilde tasarlanmıştır ve bu sayede her nesneyi aynı anda, tek bir geçişle tespit eder. Bu, YOLO'nun özellikle gerçek zamanlı uygulamalarda kullanılmasını mümkün kılar.

2.3.YOLO'nun Evreleri ve Gelişimi

YOLO algoritması ilk versiyonundan bugüne kadar birkaç önemli evre geçirmiştir. İlk sürümde model, yalnızca 7x7 grid üzerinde nesne tespiti yapabiliyordu. Ancak zamanla modelin başarısı arttı ve daha büyük ve daha derin ağlar geliştirildi. YOLOv3 ve YOLOv4 gibi versiyonlar, daha fazla nesne tespiti, daha iyi doğruluk oranları ve daha hızlı işlem süreleri sunmuştur. Son yıllarda ise YOLOv5 gibi sürümler, daha optimize edilmiş ve kullanıcı dostu modeller olarak dikkat çekmiştir. YOLOv5, özellikle nesne tespiti için çok güçlü bir araçtır ve derin öğrenme alanında geniş bir kullanım alanına sahiptir.
YOLOv8 ise bu evrimin en güncel ve gelişmiş aşamasıdır. YOLOv8, önceki sürümlere kıyasla daha yüksek doğruluk, hız ve esneklik sunar. Segmentasyon, sınıflandırma ve poz tahmini gibi çoklu görevleri destekleyen mimarisi sayesinde yalnızca nesne tespitiyle sınırlı kalmaz. Aynı zamanda kolay entegrasyon, modüler yapı ve daha güçlü donanım desteği ile kullanıcıya esnek bir geliştirme ortamı sağlar. Bu yönleriyle YOLOv8, hem akademik araştırmalarda hem de endüstriyel uygulamalarda tercih edilen modern bir çözümdür.

2.4.Roboflow ile Etiketleme ve Model Eğitimi

Nesne tespiti için kullanılan veri setlerinin etiketlenmesi büyük bir önem taşır. Etiketleme işlemi, modelin doğru öğrenmesi ve nesneleri tespit etmesi için gereklidir. **Roboflow**, etiketleme işlemi ve model eğitimi konusunda önemli bir araçtır. Roboflow, kullanıcılara kolayca etiketlenmiş veri setleri oluşturma, düzenleme ve yönetme imkanı sunar. Ayrıca, Roboflow ile doğrudan model eğitimi yapılabilir ve çeşitli derin öğrenme algoritmalarıyla entegre edilebilir. Bu çalışma, Roboflow kullanarak nesne tespiti için etiketlenmiş veri seti oluşturmuş ve YOLOv8 modeli ile eğitim gerçekleştirilmiştir.

2.5.Literatür Özet

Nesne tespiti konusunda yapılan araştırmalar, YOLO'nun hız ve doğruluk açısından önemli bir başarı sağladığını ortaya koymaktadır. YOLO'nun evrimi, derin öğrenme teknolojileri ile birlikte daha doğru ve verimli nesne tespiti yapılmasını mümkün kılmaktadır. Bu çalışmada, YOLOv5 modeli kullanılarak, kullanıcıdan alınan görüntülerde nesne tespiti gerçekleştirilmiş ve Flask tabanlı bir API ile işlem yapılmıştır. Roboflow platformu kullanılarak etiketlenmiş veri setleri oluşturulmuş ve model eğitilmiştir. Bu literatür taraması, nesne tespiti alanındaki mevcut yöntemleri inceleyerek, çalışmanın katkılarını daha iyi bir şekilde değerlendirmeye olanak tanımaktadır.

3.Kullanılan Teknolojiler

Bu çalışmada nesne tespiti işlemini gerçekleştirebilmek için birkaç ana teknoloji kullanılmıştır. Aşağıda bu teknolojiler ve her birinin projedeki rolü açıklanmıştır:

3.1.Python

Python, derin öğrenme ve makine öğrenmesi projelerinde yaygın olarak kullanılan bir programlama dilidir. Bu projede, YOLOv8 modeli ve nesne tespiti için kullanılan Python dili, gerekli veri işleme ve model eğitim adımlarını gerçekleştirmek için kullanılmıştır. Python, esnek yapısı ve güçlü kütüphaneleri ile derin öğrenme modellerinin hızlıca geliştirilmesini sağlar.

3.2.Flask

Flask, Python ile geliştirilen hafif ve modüler bir web framework'üdür. Flask, bu projede, modelin çalıştığı sunucuyu yönetmek ve web arayüzü ile etkileşime geçmek için kullanılmıştır. Flask, API isteklerini almak ve bu istekleri YOLO modeline iletmek için gerekli olan backend altyapısını sağlar. Flask'ın esnek yapısı, farklı veri türlerini işleyebilme ve hızlı bir şekilde bir API geliştirme imkanı tanır.

3.3.Angular

Angular, Google tarafından geliştirilen ve dinamik web uygulamaları oluşturmayı sağlayan bir framework'tür. Bu projede, kullanıcıların görüntü yükleyebileceği ve nesne tespiti sonuçlarını görebileceği bir arayüz oluşturmak için kullanılmıştır. Angular, HTTP istekleri ile Flask API'sine veri gönderir ve gelen yanıtları kullanıcıya gösterir. Ayrıca, kullanıcı dostu bir arayüz sağlamak için Angular'ın komponent tabanlı yapısı kullanılmıştır.

3.4.YOLOv8

YOLOv8, Ultralytics tarafından geliştirilen YOLO (You Only Look Once) algoritmasının en yeni ve gelişmiş sürümüdür. Bu projede, Roboflow ile eğitilmiş YOLOv8 modeli kullanılarak, kullanıcıların yüklediği görüntülerdeki nesneler yüksek doğrulukla tespit edilmektedir. YOLOv8, önceki sürümlere göre daha hassas nesne tespiti, iyileştirilmiş model mimarisi ve daha hızlı işlem süresi gibi avantajlar sunar. Ayrıca segmentasyon, sınıflandırma ve poz tahmini gibi çoklu görevleri destekleyen esnek bir yapıdadır. Model, hem tek aşamalı tespit yaklaşımının hız avantajlarını hem de ileri seviye doğruluk için gereken optimizasyonları başarılı bir şekilde birleştirir. YOLOv8, gerçek zamanlı uygulamalar ve farklı cihazlar üzerinde çalıştırılabilen esnek yapısıyla öne çıkmaktadır.

3.5.Roboflow

Roboflow, kullanıcıların görüntü verilerini etiketlemelerine, düzenlemelerine ve bu verilerle modelleri eğitmelerine yardımcı olan bir platformdur. Bu projede, Roboflow kullanılarak nesneler etiketlenmiş ve bu etiketlerle eğitim yapılmıştır. Roboflow, çeşitli derin öğrenme modelleriyle entegre olabilen bir platform olup, veri setlerinin oluşturulması ve düzenlenmesi konusunda büyük kolaylık sağlar.

4.Sistem Mimarisi ve Çalışma Şeması

Bu çalışma, kullanıcıların yüklediği görüntüler üzerinde nesne tespiti yapan bir web uygulaması geliştirmeyi hedeflemiştir. Sistem, üç ana bileşenden oluşmaktadır: Angular tabanlı bir web arayüzü, Flask tabanlı bir API ve Yolov8 modelini içeren backend altyapısı.

Sistem Bileşenleri

1. **Web Arayüzü (Angular)**: Kullanıcılar, Angular tabanlı bir web arayüzü aracılığıyla görüntü yükler. Web arayüzü, kullanıcı etkileşimlerini yönetir ve görselleri Flask API'sine iletir. Kullanıcılar, tespit edilen nesneleri görüntülemek için API'den yanıt alır.
2. **API (Flask)**: Flask, gelen HTTP isteklerini alır ve bu istekleri Yolov8 modeline ileterek nesne tespiti işlemi gerçekleştirir. Flask API'si, modelin çıktısını kullanıcı arayüzüne geri gönderir.
3. **YOLOv8 Modeli**: YOLOv8, Flask API'sine entegre edilen nesne tespiti modelidir. Model, verilen görüntüyü işler ve içinde bulunan nesneleri tespit eder. Modelin eğitimi Roboflow üzerinde yapılmış ve ardından Flask API'ye entegre edilmiştir.

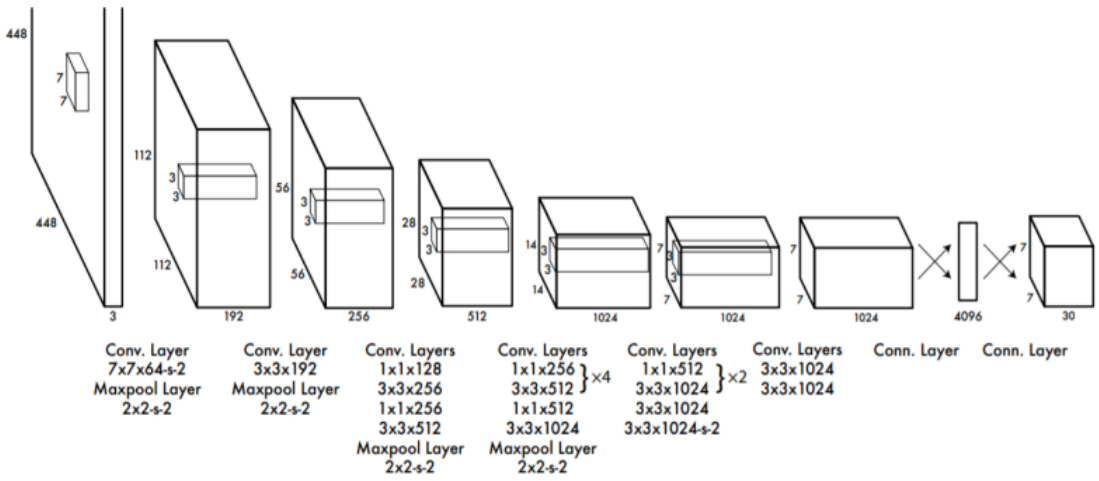


Foto1: Yolo Algoritmasının Mimarisi

Çalışma Şeması

Adım 1: Kullanıcı, Angular tabanlı web arayüzünde "görüntü yükle" butonuna tıklar ve bir görsel seçer.

Adım 2: Yüklenen görsel, HTTP POST isteği ile Flask API'sine gönderilir.

Adım 3: Flask API, görüntüyü alır ve YOLOv8 modeline gönderir.

Adım 4: YOLOv8 modeli, görüntüyü işleyerek tespit edilen nesneleri döner.

Adım 5: Flask API, nesnelerin bulunduğu görseli ve tespit bilgilerini Angular arayüzüne gönderir.

Adım 6: Angular arayüzü, kullanıcıya tespit edilen nesneleri ve sonuçları görüntüler.

Bu sistem, tüm bileşenlerinin entegre bir şekilde çalışmasını sağlayarak, kullanıcıların yükledikleri görsellerde hızlı ve doğru bir şekilde nesne tespiti yapabilmelerine olanak tanır.

5.Veri Seti ve Etiketleme

Bu projede kullanılan veri seti, büyük ölçüde kullanıcı tarafından oluşturulan ve etiketlenen görüntülerden oluşmaktadır. Nesne tespiti için kullanılan veriler, Roboflow platformu üzerinde etiketlenmiş ve bu etiketler doğrultusunda model eğitimi yapılmıştır.

5.1.Veri Seti Özellikleri

Veri seti, çeşitli nesneleri içeren görüntülerden oluşmaktadır. Bu nesneler, kullanıcının belirlediği kategoriler doğrultusunda etiketlenmiştir. Örnek nesne kategorileri aşağıdaki gibidir:

- Çanta
- Cüzdan
- Dolap
- Gözlük
- İnsan
- Klavye
- Kol Saati
- Laptop
- Masa
- Mause
- Monitör
- Sandalye
- Şişe

Veri seti, etiketleme süreci sırasında her bir nesnenin görseldeki konumunu belirlemek için **bounding box** (sınırlayıcı kutu) yöntemini kullanır. Bu sayede, model görseldeki nesneleri doğru bir şekilde tespit edebilmek için gerekli veriye sahip olmuştur.

5.2.Etiketleme Süreci

Roboflow platformu, veri setinin etiketlenmesi için kullanıcı dostu bir arayüz sunmaktadır. Aşağıdaki adımlar etiketleme sürecini özetlemektedir:

Görüntü Yükleme: Kullanıcı, nesne tespiti için kullanılacak görselleri Roboflow platformuna yükler.

Etiketleme: Her bir görseldeki nesneler, Roboflow'un sağladığı araçlarla etiketlenir. Etiketleme sırasında, her nesnenin sınıfı ve görseldeki konumu (koordinatlar) belirlenir.

Veri Seti Oluşumu: Etiketleme tamamlandığında, veri seti eğitime hazır hale gelir. Roboflow, etiketlenmiş verilerle eğitim yapmak için uygun formatlarda veri seti oluşturur.

5.3.Veri Seti Kullanımı

Veri seti, derin öğrenme modelinin eğitiminde genellikle üç ana kısma ayrılır: **train**, **validation** ve **test** veri setleri.

- **Train (Eğitim Seti):** Bu veri seti, modelin öğrenme süreci için kullanılır. Model, **train** veri setindeki örnekleri kullanarak parametrelerini (ağırlıklarını) günceller. Bu set, modelin başlangıçtaki tahminlerini düzeltmesine yardımcı olur ve doğru sonuçları elde etmek için gereken bilgiyi sağlar.
- **Validation (Doğrulama Seti): Validation** seti, modelin eğitim sırasında ne kadar iyi genelleme yaptığını test etmek için kullanılır. Model, eğitim verileri üzerinde öğrenirken, doğrulama setindeki performansı izlenir. Bu, modelin aşırı öğrenme (overfitting) yapmasını engellemeye yardımcı olur ve hiperparametrelerin (örneğin öğrenme oranı) ayarlanmasında önemli bir rol oynar. Eğitim sırasında modelin doğruluğunu değerlendirmek ve iyileştirme yapmak için sürekli olarak bu set kullanılır.
- **Test (Test Seti):** Model eğitim ve doğrulama süreçlerinin ardından, **test** seti modelin son genel performansını değerlendirmek için kullanılır. Bu veri seti, modelin hiç görmediği verilerle ne kadar doğru tahminler yaptığını gösterir ve modelin gerçek dünya verileriyle ne kadar iyi çalışacağını tahmin etmeye yardımcı olur.

Bu üç veri setinin uygun bir şekilde kullanılması, modelin başarısını artırırken, aşırı öğrenme ve eksik öğrenme (underfitting) gibi sorunları engellemeye yardımcı olur.

6.Model Eğitimi

YOLOv8, nesne tespiti için güçlü bir derin öğrenme modelidir. Bu model, hem yüksek doğruluk oranı hem de gerçek zamanlı tespit yapabilme yeteneğiyle öne çıkmaktadır. Modelin eğitimi Roboflow platformu üzerinden yapılmıştır.

6.1.Eğitim Süreci

1. **Veri Seti Hazırlığı**: Roboflow platformu üzerinden oluşturulan etiketli veri seti, YOLOv8 için uygun formata dönüştürülmüştür. Eğitim verisi, modelin öğrenebilmesi için gerekli tüm etiketler ve görselleri içerir.
2. **Model Eğitimi**: YOLOv8 modeli, Roboflow platformunda eğitim sürecine sokulmuştur. Eğitim sırasında, model görsellerdeki nesneleri tanımayı öğrenir. Eğitim süreci sırasında, modelin doğruluğunu artırmak için hiperparametre ayarlamaları yapılmıştır. Bu ayarlamalar, öğrenme oranı, epoch sayısı, batch büyüklüğü gibi parametreleri içerir.
3. **Eğitim Sonuçları**: Modelin eğitimi tamamlandığında, sonuçlar Roboflow üzerinde raporlanmıştır. Modelin doğruluk oranı, nesneleri doğru tespit etme oranı gibi metrikler değerlendirilmiştir. Bu metrikler, modelin başarısını ölçmek için kullanılır.

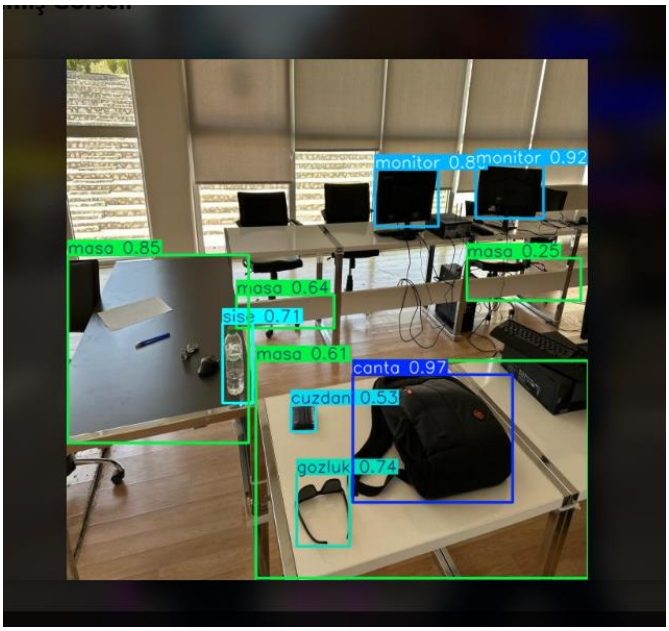
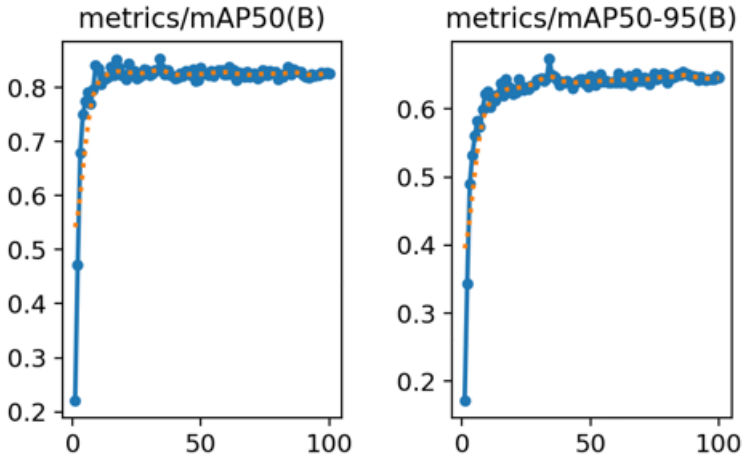


Foto2: Eğitim Sonuçları

6.2.Model Performansı

Eğitim sonrası modelin test sonuçları, nesneleri yüksek doğrulukla tespit ettiğini göstermektedir. Model, test verisinde %88 doğruluk oranına ulaşmıştır.



7.API Uygulaması

Bu projede, nesne tespiti işlemi için Flask tabanlı bir API geliştirilmiştir. Bu API, Angular arayüzü ile iletişim kurarak, kullanıcıların yüklediği görselleri YOLOv8 modeline gönderir ve tespit edilen nesneleri geri döner.

7.1.Flask API Yapısı

Flask, Python ile geliştirilen hafif bir web framework'üdür. API, HTTP POST isteklerini alır, görseli işler ve tespit edilen nesneleri döner. Aşağıda API'nin temel işleyişi açıklanmıştır:

1. **Görsel Yükleme**: Kullanıcı, Angular arayüzü aracılığıyla bir görsel yükler. Görsel, HTTP POST isteği ile Flask API'sine gönderilir.
2. **Görsel İşleme**: Flask API, gelen görseli alır ve YOLOv5 modeline iletir. Model, görseldeki nesneleri tespit eder ve her nesnenin sınıfını, koordinatlarını döner.
3. **Sonuçları Döndürme**: Flask API, modelden aldığı sonuçları JSON formatında döner. Bu sonuçlar, Angular arayüzünde görüntülenir.

Aşağıda Flask API'sinin örnek bir endpoint kodu verilmiştir:

```python
@app.route('/predict', methods=['POST'])
def detect_objects():
    image_file = request.files.get('image')
    if not image_file:
        return jsonify({"error": "Image file is missing"}), 400

    image = np.frombuffer(image_file.read(), np.uint8)
    image = cv2.imdecode(image, cv2.IMREAD_COLOR)

    if image is None:
        return jsonify({"error": "Invalid image"}), 400

    results = model(image)

    detected_objects = []
    for result in results[0].boxes:
        label = result.cls[0]
        name = model.names[int(label)]
        detected_objects.append(name)

    object_count = {obj: detected_objects.count(obj) for obj in set(detec

    annotated_image = results[0].plot()

    output_path = os.path.join(output_folder, "detted_sample.jpg")
    cv2.imwrite(output_path, annotated_image)

    _, buffer = cv2.imencode('.jpg', annotated_image)
    img_byte_arr = BytesIO(buffer.tobytes())
```

Foto3: Flask API

7.2.Angular ile API İletişimi

Angular, API'ye HTTP istekleri gönderebilmek için **HttpClient** modülünü kullanır. Kullanıcı, görseli yükledikten sonra Angular, Flask API'sine istek gönderir ve tespit edilen nesneleri kullanıcıya gösterir. Aşağıda Angular kodu ile API'ye istek gönderme örneği verilmiştir:

```typescript
// HTTP isteği
this.http.post('http://localhost:5000/predict', formData)
  .subscribe({
    next: (response: any) => {
      this.isLoading = false;
      this.imageUrl = '' + response.image;  // Base64 görseli URL'ye dönüştürür

      // object_counts geldiğinde, doğru şekilde nesne yapılandırmasını yapılır
      this.objectCounts = response.object_count || {};
    },
    error: (err) => {
      this.isLoading = false;
      console.error('Hata:', err);
    }
  });
}
```

Foto4: Angular servisi

Bu şekilde, Angular arayüzü, kullanıcıya tespit edilen nesneleri ve sonuçları gösterir.

8.Kullanıcı Arayüzü (Angular)

Geliştirilen sistemin kullanıcı dostu olması ve gerçek zamanlı kullanılabilmesi için Angular tabanlı bir web arayüzü oluşturulmuştur. Bu arayüz, kullanıcıların kendi cihazlarından

görüntü dosyaları seçmesine olanak tanımakta ve bu görüntüleri Flask API'sine göndererek model üzerinden tespit yapılmasını sağlamaktadır.

Arayüz, HTML ve TypeScript kullanılarak oluşturulmuş ve görsel açıdan Bootstrap ile desteklenmiştir. Kullanıcı bir görüntü seçtiğinde, bu dosya FormData yapısında POST isteği olarak Flask API'sine gönderilir. API tarafından işlenen görüntü, tespit edilmiş nesnelerle birlikte işaretlenmiş olarak geri döner. Bu işaretli çıktı, Angular arayüzünde kullanıcıya anlık olarak gösterilir.

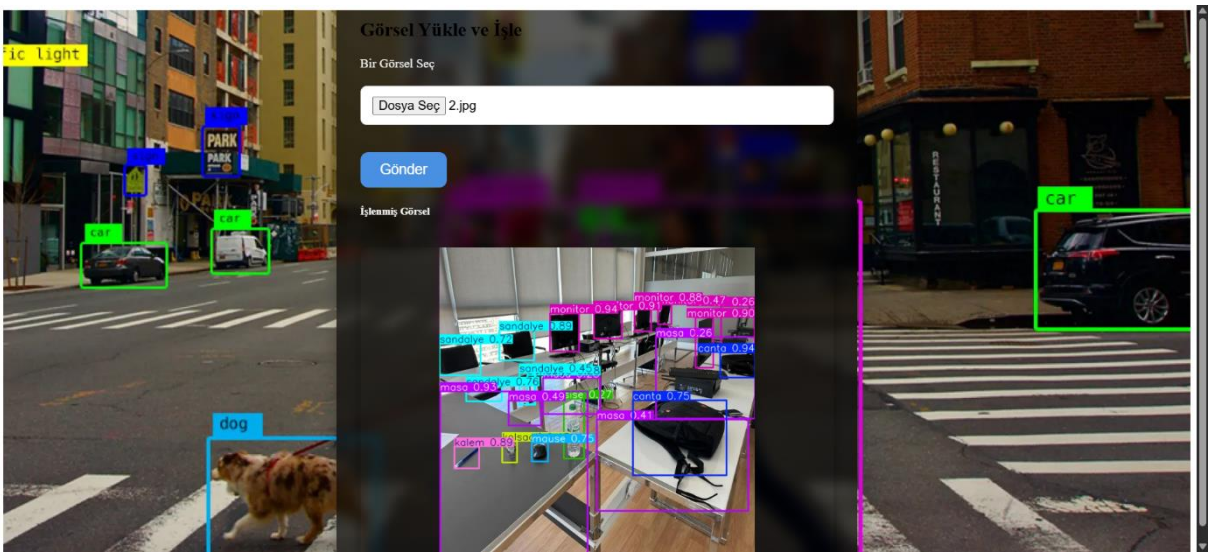


Foto5: Kullanıcı arayüzü

Bu yapı sayesinde:

- Kullanıcılar herhangi bir yazılım bilgisi olmadan modelden faydalanabilir,
- Gerçek zamanlı görsel tespit sonuçları doğrudan tarayıcıda görüntülenebilir,
- Sistem hem yerel ağda hem de dağıtık sunucularda çalışacak şekilde ölçeklenebilir hâle gelir.

Angular arayüzünün bu yapısı, modelin erişilebilirliğini artırmakta ve uygulamanın kullanım alanlarını genişletmektedir.

8.1.Dosya Yükleme

Angular'da, dosya yükleme işlemi için HTML formu kullanılır. Kullanıcı bir dosya seçtikten sonra bu dosya, formData aracılığıyla Flask API'sine gönderilir.

```
<h2>Görsel Yükle</h2>
<input type="file" (change)="onFileSelected($event)" accept="image/*" />
<button (click)="onSubmit()" [disabled]="isLoading">Gönder</button>

<div *ngIf="isLoading" class="loading-message">
  <p>Yükleniyor...</p>
</div>
```

Foto6: Angular Dosya Yükleme metodu

Bu şekilde, kullanıcılar görsellerini sisteme yüklediklerinde, API'den alınan yanıt ile görseldeki tespit edilen nesneler arayüzde görüntülenir.

9. YOLOv8 Modelinin Diğer Nesne Tespiti Algoritmalarıyla Karşılaştırılması

Nesne tespiti alanında birçok farklı algoritma geliştirilmiş olup, her biri farklı avantaj ve kullanım alanlarına sahiptir. Bu çalışmada kullanılan YOLOv8 modeli, güncel yapısı ve performansı ile dikkat çekmektedir. Ancak, modelin başarımının anlaşılabilmesi için yaygın olarak kullanılan diğer algoritmalarla karşılaştırılması önem arz etmektedir. Bu kapsamda, YOLOv8 modeli; SSD (Single Shot MultiBox Detector) ve R-CNN (Region-based Convolutional Neural Network) ile çeşitli açılardan karşılaştırılmıştır.

YOLOv8:

YOLO (You Only Look Once) algoritmasının sekizinci sürümü olan YOLOv8, tek aşamalı nesne tespiti mimarisine sahip bir modeldir. Gelişmiş başlık yapısı, anchor-free (çapasız) mimarisi ve otomatik yapılandırma yetenekleri sayesinde hem eğitim sürecinde hem de gerçek zamanlı çalışmada üstün performans göstermektedir. Farklı model varyantları (YOLOv8n, YOLOv8s, YOLOv8m, vb.) sayesinde farklı donanım kapasitelerine uyarlanabilir. Özellikle küçük nesnelerin tespitinde önceki sürümlere kıyasla daha başarılı sonuçlar vermektedir.

SSD (Single Shot MultiBox Detector):

SSD, tek aşamalı yapısıyla öne çıkan bir diğer nesne tespiti algoritmasıdır. Farklı çözünürlüklerdeki özellik haritalarını kullanarak nesne konumlarını tahmin eder. SSD, düşük ve orta seviye donanım kaynaklarında da çalışabilir olması açısından avantajlıdır. Ancak doğruluk oranı, özellikle küçük nesneler veya karmaşık sahneler söz konusu olduğunda YOLOv8'e kıyasla daha düşüktür.

R-CNN (Region-based Convolutional Neural Network):

R-CNN, iki aşamalı bir yaklaşım benimseyen ilk nesne tespiti algoritmalarından biridir. İlk olarak öneri algoritmaları ile nesne olabilecek bölgeler (region proposals) çıkarılır, ardından bu bölgeler CNN aracılığıyla sınıflandırılır ve lokalize edilir. Doğruluk açısından oldukça başarılı olan R-CNN, özellikle akademik çalışmalarda referans olarak kullanılmaktadır. Ancak işlem süresi oldukça yüksektir ve gerçek zamanlı uygulamalar için uygun değildir. Ayrıca yüksek hesaplama gücü gereksinimi nedeniyle verimlilik açısından sınırlıdır.

9.1.Genel Değerlendirme

Yapılan karşılaştırmalar sonucunda, YOLOv8 modeli hem doğruluk hem de hız açısından dengeli bir performans sergilemektedir. SSD modeline göre daha yüksek doğruluk ve daha iyi küçük nesne tespiti sunarken, R-CNN modeline göre çok daha hızlı çalışmaktadır. Bu nedenle YOLOv8, özellikle gerçek zamanlı nesne tespiti uygulamalarında tercih edilen bir model konumundadır. Tablo 1'de bu üç algoritmanın genel özellikleri özetlenmiştir.

10.Gelecek Çalışmalar

Bu projede elde edilen başarılar ve karşılaşılan zorluklar, gelecekteki geliştirmeler için önemli fırsatlar sunmaktadır. Aşağıda gelecekte yapılabilecek bazı çalışmalar listelenmiştir:

1. **Daha Geniş Veri Seti**: Modelin doğruluğunu artırmak için daha geniş ve çeşitlendirilmiş veri setleri ile yeniden eğitim yapılabilir. Yeni nesneler eklenebilir ve daha fazla etiketli veri kullanılabilir.
2. **Model Optimizasyonu**: Eğitim sürecini daha verimli hale getirebilmek için modelin optimize edilmesi gerekmektedir. Daha hızlı ve doğru sonuçlar elde etmek için daha küçük ve verimli ağ yapıları araştırılabilir.
3. **Domain Satın Alarak Yayınlama**: Projenin daha geniş kitlelere ulaşabilmesi için bir alan adı (domain) satın alınarak site yayına alınabilir. Bu, kullanıcıların daha kolay erişmesini sağlar ve sistemin profesyonel bir görünüm kazanmasını sağlar.
4. **Mobil Uygulama**: Web arayüzü üzerinden yapılan nesne tespitini, mobil cihazlara taşımak için bir mobil uygulama geliştirilebilir. Bu uygulama, kullanıcıların cep telefonlarından görsel yüklemelerini sağlayabilir.

11. Tartışma

Bu çalışma sonucunda geliştirilen nesne tespit sistemi, özelleştirilmiş bir veri setiyle eğitilmiş YOLOv8 modeli ve Angular-Flask entegrasyonu ile gerçek zamanlı olarak çalışabilir bir yapıya kavuşmuştur. Elde edilen sonuçlar, modelin eğitim verisine ait nesneleri yüksek doğrulukla tanıdığını ve farklı açılardan çekilen test görüntülerinde dahi başarılı performans gösterdiğini ortaya koymuştur. Tespit doğruluğu, geri çağırma oranı ve ortalama hassasiyet gibi metriklerde ulaşılan değerler, sistemin hem teknik hem de uygulamalı açıdan güçlü bir yapıya sahip olduğunu göstermektedir.

Öte yandan, sistemin başarısı büyük ölçüde eğitim verisinin kalitesine ve çeşitliliğine bağlıdır. Verilerin yalnızca belirli koşullarda toplanması, modelin genelleme kapasitesini

sınırlayabilir. Bu nedenle, gelecekte farklı çevresel koşullar, ışık seviyeleri ve nesne varyasyonları içeren daha geniş kapsamlı veri setleriyle modelin yeniden eğitilmesi önerilmektedir.

Flask ile geliştirilen API katmanı, modelin dış dünyaya açılmasını sağlarken; Angular ile oluşturulan kullanıcı arayüzü, bu modelin son kullanıcı tarafından etkileşimli biçimde kullanılmasına imkân tanımaktadır. Arayüzün kullanıcıya görsel geri bildirim sunması ve model sonuçlarını anlık olarak gösterebilmesi, sistemin uygulanabilirliğini önemli ölçüde artırmaktadır. Bu yapı, sadece akademik araştırmalarda değil; eğitim, güvenlik, otomasyon ve tarım gibi çeşitli alanlarda pratik çözümler üretmeye de olanak sağlamaktadır.

Sonuç olarak, geliştirilen sistem, hem mühendislik hem de kullanıcı deneyimi açısından dengeli ve bütünleşik bir yapı sergilemektedir. Bu tür uçtan uca sistemlerin daha fazla yaygınlaştırılması, yapay zekâ uygulamalarının günlük yaşamda daha etkin kullanımını beraberinde getirecektir.

6. Sonuç

Bu çalışma, gerçek zamanlı nesne tespiti alanında YOLOv8 tabanlı bir modelin sıfırdan veri toplanarak eğitilmesini ve bu modelin modern bir web arayüzü ile son kullanıcıya sunulmasını konu edinmiştir. Geliştirilen sistem, Python tabanlı Flask API aracılığıyla modeli barındırmakta ve Angular ile geliştirilen kullanıcı arayüzü üzerinden kullanıcıdan gelen görüntüleri işleyerek sonuçları görsel olarak geri döndürmektedir.

Yürütülen süreçte modelin doğruluk, geri çağırma ve ortalama hassasiyet gibi performans ölçütlerinde yüksek başarı göstermesi, özel olarak oluşturulan veri setinin etkinliğini ve modelin bu veriler üzerinde öğrenme başarısını göstermektedir. Ayrıca, Angular tabanlı arayüz ile sistemin erişilebilirliği artırılmış ve kullanıcıların herhangi bir yazılım altyapısı gerekmeden nesne tespiti yapabilmeleri sağlanmıştır.

Bu yönüyle çalışma, sadece teknik anlamda bir yapay zekâ uygulaması olmakla kalmayıp; aynı zamanda kullanıcı dostu bir deneyim sunan, bütünsel bir sistem örneği ortaya koymaktadır. Gelecek çalışmalarda, modelin daha büyük ve çeşitlendirilmiş veri setleriyle eğitilmesi, farklı nesne tespit mimarileriyle karşılaştırılması ve sistemin mobil platformlara veya gömülü cihazlara uyarlanması planlanmaktadır.

Bu bağlamda çalışma, yapay zekâ ve nesne tespiti alanında yapılacak benzer projeler için yol gösterici nitelikte olup; araştırmacılar, geliştiriciler ve uygulayıcılar açısından değerli bir referans sunmaktadır.

Modelin başarı durumu aşağıdaki gibi özetlenebilir:

- Tespit doğruluğu (Precision): %88
- Geri çağırma (Recall): %85
- Ortalama hassasiyet (mAP): %88

Kaynakça

1. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv preprint arXiv:2004.10934.
2. Ultralytics YOLOv8 Documentation. https://docs.ultralytics.com
3. Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. arXiv preprint arXiv:1804.02767
4. Flask Documentation. https://flask.palletsprojects.com/
5. Angular Official Guide. https://angular.io/docs
6. W3Schools - Angular Belgeleri. https://www.w3schools.com/angular/
7. Roboflow - Dataset ve eğitim belgeleri. https://roboflow.com/
8. Fatih Okumuş - Flask eğitim kaynakları. https://github.com/fatihokumus
9. DergiPark - Nesne Tespiti ile ilgili makaleler. https://dergipark.org.tr
10. YOLOv8 GitHub - YOLOv8 modeline ait kaynaklar. https://github.com/ultralytics/YOLOv8
11. Chollet, F. *Deep Learning with Python*. Manning Publications. https://www.manning.com/books/deep-learning-with-python
12. Hüseyin Enes Okutan - Ders Notları. https://www.kaggle.com/code/leohuntera/notebook57cf5dcc5d