TEKNOFEST Havacılık, Uzay ve Teknoloji Festivali Ulaşımda Yapay Zeka Yarışması Final Tasarım Raporu

AVION AI
381923

Murat Bayraktar [*Takım Kaptanı*], **Buket Naz Zeren** [*Yazılım Birleştirme ve Algoritmalar Sorumlusu*], **Ahmet Alp Güneş** [*Veri Seti ve Ön İşleme Sorumlusu*], **Hasan Oğuzhan Gök** [*Görüntü Ön İşleme Sorumlusu*]

Özet

Uçan arabalar ve drone teknolojilerinin insan hayatına entegre olmaya başlamasıyla bu sistemlerin alt görüş kameraları kullanılarak nesne tespiti yapılması kritik önem kazanmaya başlamıştır. Klasik nesne tespiti problemlerinden farklı olarak bu otonom sistemlerin gerçek zamanlı olarak, farklı çevre koşullarında ve yüksek irtifada tespit yapması gerekmektedir. Bu problemi çözmek için geliştirdiğimiz sistem, VisDrone problemine çözüm olarak önerilen TPH-YOLOv5'i kullanarak bunu bir CNN modeli ile birleştirmektedir. Bu birleştirilmiş sistemi 16000 resimden oluşan veri setimiz ile eğiterek yarışma komitesi tarafından sağlanan 2. oturum verileri üzerinde 0.846 mAP değerine ulaşmayı başardık.

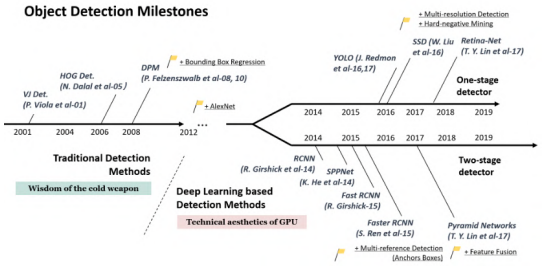

Figure 1: Nesne Tespiti Yöntemlerinin Tarihsel Gelişimi

1.Giriş

Nesne Algılama/Nesne Tespiti bilgisayar görüntüleri ve görüntü işlemeyle ilgili, dijital görüntü ve videolarda belirli bir sınıftaki semantik nesnelerin örneklerinin algılanmasıyla ilgilenen bir araştırma alanıdır. [1] Nesne tespiti son yıllarda uçan araba ve drone teknolojilerinde çokça uygulanmaya başlamıştır. Uçan arabalar ve drone teknolojilerinin son yıllarda çok ilgi çekmesi ve gelişmesiyle birlikte bu sistemlere otonom özellikler eklenmeye ve bu teknolojiler insan hayatına entegre olmaya başlamıştır. Bu durum sensör ve kamera verilerinin değerlendirilerek otonom sistemlerin çevresel farkındalığının artırılması ihtiyacını doğurmuştur. Bu bağlamda kazaları önlemek için alt görüş kamera görüntüleri kullanılarak nesne tespiti yapılması kritik önem arz etmektedir. Bu yüzden bu otonom sistemlerin gerçek zamanlı olarak, farklı çevre koşullarında yüksek doğrulukta tespit yapması gerekmektedir.

Önceki Çalışmalar Daha önce yapılan araştırmalarda ilk gerçek zamanlı tek adımlı tespit yapan **Viola Jones Dedektörüdür** [2]. Ancak Viola Jones algoritmasının en başarılı olduğun alan insan yüzü tespiti olduğu için diğer alanlarda geliştirilmeye ihtiyacı vardı. Onu takip eden **HOG** [3] ve **DPM** [4] dedektörleri ise yine tek adımlı tespit mekanizmaları olup uygulama alanı olarak kısmen yetersizdi ve insan müdahalesi gerekiyordu. 2012 yılında bütün dünyanın evrişimsel sinir ağlarının gücüne tanık olmasıyla beraber gerçek zamanlı obje tespiti de bundan nasibini aldı [5]. Bu gelişmenin ilk örneklerinden olan RCNN yöntemi[6] **seçici arama** (selective search) kullanarak nesne önermeleri oluşturur ve ardından çıkardığı özellikleri bir CNN'den (evrişimsel sinir ağı) geçirir. Son olarak bir SVM sınıflandırıcı kullanarak tespit yapar. RCNN başarı olarak büyük bir gelişme sağlamış olsa da tespit süresi fazla olduğundan gerçek zamanlı kullanım için çok yavaş kalmaktadır. Onu takiben **Fast-RCNN** [7] ve **Faster-RCNN** [8] algoritmaları geliştirilmiştir. Fast-RCNN'de nesne önermeleri hala RCNN'deki gibi seçici arama yöntemiyle yapılırken Faster-RCNN'de ise nesne önermeleri bir evrişimsel sinir ağı yardımıyla oluşturulur. Bu durum Faster-RCNN'in hız olarak büyük bir gelişme göstermesini sağlamıştır. 2015 yılında araştırmacılar "nesne önerme" ve "doğrulama" metotlarını bir kenara bırakıp tamamen yeni bir paradigma getirerek YOLO algoritmasını geliştirdiler [9]. YOLO, tek adımda bütün bir sinir ağından görüntüyü geçirerek her bir bölge için olasılıklar oluşturup bunun üzerinden ilerleyerek tespit yapmaktadır. YOLO'nun çok hızlı çalışması ve yüksek doğrulukta sonuçlar vermesi nedeniyle son zamanlarda oldukça tercih edilen ve üzerinde çokça araştırma yapılan bir proje olmuştur. Daha sonraları v2 [10], v3 [11], v5 [12] ve 2022 yılında da v7 [13] ortaya çıkmıştır.

Problem Tanımı Uçan arabalar için önem arz eden insanlar, arabalar ve iniş alanları tespit edilmeli ve bu iniş alanlarının uygun olup olmadığı bilgisi aktarılmalıdır.

Figure 2: Sistemimizin Çıktısını Gösteren Örnek Bir Görüntü

Bu alanda yapılan çalışmalarda bizim problem ve kısıtlamalarımıza benzer olan VisDrone problemi ve çözümleri bize ilham kaynağı olmuştur. VisDrone yarışmasında yüksek bir skor alan **TPH-YOLOv5** [14] bizim ana modelimizin temelini oluşturmuştur. Biz bu raporumuzda bu model üzerinde iyileştirmeler yaparak ve **model birleştirme** (model ensembling) uygulayarak problem tanımına uygun bir çözüm geliştirdik.

2. Kullanılan Veri Setleri

Devrim Veri Seti

Kritik Tasarım Raporu'nda da belirtildiği üzere Orta Doğu Teknik Üniversitesi kampüsünde çekim yaparak kendi veri setimizi oluşturduk. Bu veri setini oluşturmak istememizdeki amaç istediğimiz açı ve sayıda insan ile iniş alanı verisi bulamamız olmuştur. Belirlenen ölçülerde (4.5m çap) bastırdığımız Uçan Araba Park (UAP) ve Uçan Ambulans İniş (UAP) örneklerini kampüsün çeşitli bölgelerine koyarak drone ile çekim yaptık. Bu çekimleri yaparken yarışmada karşılaşabileceğimiz senaryolar düşünülerek çekimler yapıldı. Bunun için karşılaşabileceğimiz tüm senaryolar dikkate alındı. Bunlar :

- İrtifa (25-100 metre arası)
- Açı (60-90 derece arası)
- İniş Alanı Uygunluk Durumu (İnilebilir-İnilemez)
- Gölge ve Işık seviyesi
- İnsanlar için Farklı Senaryolar (Yürüme-Oturma-Uzanma-Eğilme)

Ayrıca Şekil 3'de görüldüğü üzere hem iniş alanının dolu olduğu hem de güneş ışınlarının yansıdığı görüntüler de elde edilmiştir. Farklı senaryoların kombinasyonlarına da çekim yapılırken dikkat edildi.

Bu fotoğraflardan açısı ve irtifası uygun olanları ayıklayıp her 6 kareden 1 tanesini seçerek yaklaşık 2000 adet veri elde ettik. Arada birkaç kare atlamamızın sebebi olabildğince farklı görüntüleri elde etmek ve olabildğince hızlı bir şekilde veri etiketleme işini bitirebilmektir. Veri sorumlularımız liderliğinde tüm takımımız ve bize destek veren alt takımımız ile birlikte verileri etiketledik. Özellikle insanların çok olduğu fotoğrafları etiketlemenin zaman alması sebebiyle beklediğimiz kadar hızlı etiketleyemesek de

Figure 3: Devrim Veri Seti

büyük bir özveriyle bütün hepsini etiketlemeyi tamamlayıp training setimize ekledik. Verileri etiketlerken zaman kazanmak için öncelikle 600 adet fotoyu etiketleyip modelimizi eğittik. Daha sonra eğittiğimiz modelle kalan yaklaşık 1400 fotoğrafı test ettik ve takım olarak modelin tespitlerini gözden geçirdik, düzelttik. Ardından veri sorumlularımız teker teker tüm fotoğrafları kontrol ederek veri setimizi ve etikletlerimizi olabildiğince hatasız hale getirdi.

Figure 4: Devrim Veri Seti

VisDrone Veri Seti

Araştırmalarımız sonucunda bu veri setinin bu tarz yarışmalarda ve araştırmalarda sıkça kullanıldığını tespit ettik. Ekibimiz bu veri setini gözden geçirerek istenilen açı ve yükseklikteki yaklaşık 5000 fotoğrafı tespit etti sınıflarını ayarladı ve training setimize ekledi.

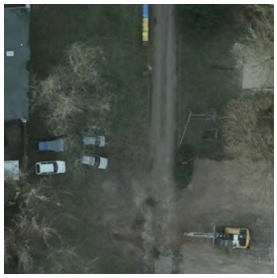

Figure 5: VisDrone Veri Seti

Ulaşımda Yapay Zeka Yarışması 2021 Verileri

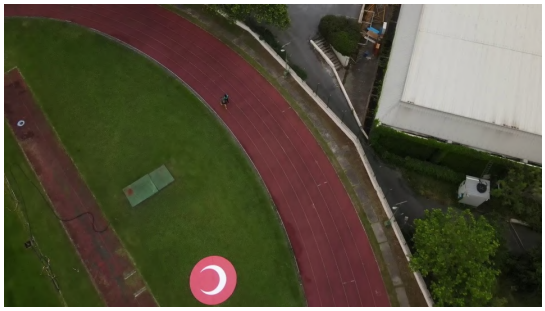

Figure 6: UYZ Veri Seti

Bu yarışmada başarıya ulaşmanın en kritik yollarından birinin doğru ve zengin bir veri seti oluşturmak olduğunun kanaatindeyiz. Bu sebeple bulabildiğimiz temiz bütün verileri kullandık. Bu veri seti özelinde içerdiği UAP ve UAİ framelerini validation setimize koymak üzere ayırdık. Kalanlarını training setimize ekledik.

Ulaşımda Yapay Zeka Yarışması 2019 Verileri

Bu veri setinin bize verilen örnek videonun daha zengin hali olduğunu fark ettik. Bu sebeple bu veri setiniin bi kısmını training setimize bi kısmını da validation setine ekledik. Özellikle validation seti ile training seti arasında olabildiğince az benzerlik olmasına dikkat ederek, araya frame farkı koyarak verileri ayırdık.

Figure 7: UYZ Veri Seti

İnternetten Bulunan Diğer Veriler

Ekibimizin çalışması ve araştırmaları sonucunda 10 adet farklı veri seti bulundu.Bu veriler iStock, Kaggle ve Pexel gibi sitelerden bulundu. Hepsi özelliklerine içeriklerine ve kullanıma uygun olup olmamasına göre sınıflandırıldı. Veri setinde her sınıf arasında bir denge gözettiğimizden dolayı ihtiyacımıza uygun olarak insan içerikli veri setleri düzenlendi etiketlendi ve bizim için uygun hale getirildi.

Çevrimiçi Simülasyon Verileri

Topladığımız bütün verilerde yarışma koşullarına uymasına ve çeşitlilik olmasına dikkat edildi. Toplamda

Figure 8: Internet Veri Seti

yaklaşık 16.000 veri çevrimiçi simülasyon öncesi training setine eklendi. Oluşturuduğumuz modeli doğru bir şekilde incelemek ve isabetli metrikler elde etmek, doğru şekilde geliştirebilmek için validation setinin öneminin farkındaydık. Bunun için training seti ile benzer frameler içermeyen validation setimize çevrimiçi simülasyon verilerini de etiketleyip ekleyerek yaklaşık 1500 adet veri ile validation setimizi oluşturduk.

Figure 9: ÇİS Veri Seti

Kırpılmış UAİ-UAP Veri Seti

İniş alanlarının inişe uygun olup olmama durumunu belirlemek için yeni bir CNN modeli geliştirdik. Bu CNN modelini eğitmek için de ana veri setimizde bulunan bütün UAP-UAİ içeren resimleri kırparak kendimize Figure 10 de gösterildiği gibi 2000 adet kırpılmış gerçek UAİ-UAP verisi elde ettik. Çalışmalarımızın ilk safhalarında bu verileri kullnarak çeşitli CNN mimarileri denedik. Ancak elimizdeki verilerin kısıtlı olmasından dolayı modellerin genelleme yeteneğinin kısıtlı olduğunu fark ettik. Bu nedenle elimizdeki verilerin yöne olan bağımlılığını azaltmak için döndürme ve yansıtma uyguladık. Ayrıca renge ve parlaklığa olan bağımlılığını azaltmak için bu değerleri belli oranda değiştirdik. Bunların dışında drone çekimi şartlarını taklit etme amacı ile bu karelere bulanıklaşma, drone irtifasını da göz önüne alarak büyütme ve küçültme uyguladık. Bu teknikleri rastgele bir şekilde elimizdeki 2000 gerçek kareye uygulayarak yaklaşık 10 000 veriyi elde ettik. Bu elde ettiğmiz veri seti ile iniş alanı belirlemek için kullanacağımız CNN modeli için eğitimlere devam ettik.

Figure 10: Kırpılmış UAİ-UAP Veri Seti

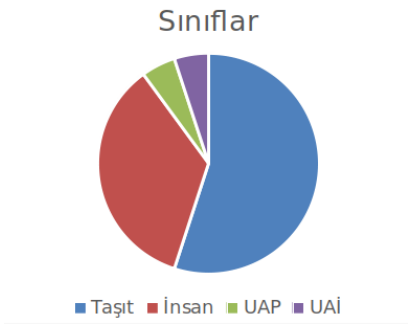

Figure 11: Veri Seti Dağılımı

Veri Zenginleştirme

Kırpılmış UAİ-UAP verilerinde kullanılan yöntemler o bölümde anlatılmıştır. Ana veri setimizde kullandığımız veri zenginleştirme ayarları şöyledir:

- hsv_s:0.5
- hsv_v:0.4
- degrees:0.0
- translate:0.1
- scale:0.9
- shear:0.0
- perspective:0.0
- flipud:0.0
- fliplr:0.5
- mixup:0.1
- mosaic:1.0 (Küçük nesne tanıma problemlerinin çoğunda bu zenginleştirme tekniği kulanılmıştır.)
- hsv_h:0.0 (UAP-UAİ alanlarının tespitinin zorlaştırıldığı gözlemlenmiş olup bu sebeple bu ayar 0 olarak belirlenmiştir.)

3. Yöntem

Problem tanımında da belirtildiği gibi 4 adet sınıf tespit etmemiz gerekiyor. İnsan, araba tespitlerinin yanında UAİ ve UAP alanlarının tespitinden sonra bir de bu alanların iniş durumlarının müsaitliği tespit edilmesi gerekiyordu. İniş alanının uygunluk durumunu kontrol ederken alanın üzerindeki nesneyi tespit etmek sistem kısıtlamaları dahilinde mümkün değilken, alanın boş veya dolu olduğunu sadece insan ve arabayla kıyaslayarak da bulmanın genel bir çözüm olmadığı kanaatindeyiz. Ancak ana modelimizin asıl görevinin objeleri tespit etmek olduğundan iniş alanlarının üzerinde insan veya araba dışında başka bir obje olduğunda bunu tespit etmekte oldukça zayıf olduğunu gördük. Bu sebeple iniş alanı müsaitliğini kontrol edebilmek için ayrı bir evrişimsel sinir ağı eğitmeye karar verdik. Bu modelin görevi ise verilen kırpılmış iniş alanı fotoğrafından 0 ya da 1 şeklinde iniş durumunu sınıflandırmaktır. Bu evrişimsel sinir ağının görevi görece basit olduğundan aşırı öğrenmenin önüne geçmek adına bu ağı oldukça sığ ve ilkel tuttuk. Bu sayede süre anlamında da oldukça yol kat ettik.

Ana model TPH-YOLOv5'ten oluşuyor. TPH (Transformer Prediction Head) küçük objeler için YOLOv5 mimarisine eklenmiş bir baş katmandır. Bu katman diğer 3 katmanla birlikte çalışarak farklı boyuttaki aynı nesneleri tespit etmekte % 7 daha iyi çalışıyor [14]. Problem tanımında ise uçan arabanın 25-100 metre arasında uçacağı ve boyut farklılıklarının sorun yaratacağı belirtiliyor. Bu sebeple TPH bu sorunu çözmek için önemli bir adım. **Dönüştürücü Kodlayıcı Blok** (Transformer Encoder Block) ise global özelliklerinin yakalanmasına olanak sağlıyor ve modelin gereksiz bilgileri depolamasını engelliyor. Bunun en güzel örneklerinden birisi sürekli kaldırımda insan görmeye alışmış bir TPH-YOLOv5 modelinin kaldırımda insan gördüğünde bunu çok daha kolay ve yüksek doğrulukla tespit edebilmesidir.

A. Model Mimarisi

Model mimarisi 3 adımdan oluşuyor. Öncelikle aldığımız görüntü obje tespiti modelinden geçerek objelerin fotoğraf üzerindeki yerleri belirleniyor. İnsan ve Taşıt tespitlerinde bu adım yeterli olurken UAP ve UAİ tespitlerinde obje tespiti modelinin verdiği koordinatlara göre iniş alanı kırpılarak evrişimsel bir sinir ağına veriliyor. Bu ağ ise iniş alanının uygun olup olmadığını tespit ediyor.

B. Model Gerçekleştirme Adımları

Metrikler

Modelimizin başarısını ölçmek için çalışmalarımız boyunca hassasiyet, anımsama, F1 oranı, güvenilirlik oranı, ortalama kesinlik değerlerinin ortalaması gibi değerler; PR grafiği ve hata matrisleri kullanılmıştır.

Hata matrisi (confusion matrix) , verilerin asıl etiketleri ile modelimizin doğru ve yanlış tahminlerinin karşılaştırmasını gösterir. Aşağıda 2×2'lik bir hata matrisi görülmektedir. Bu matrisin boyutu tahmin edilecek sınıf sayısına göre değişebilir. Bu matris modelimizin veriyi aşırı öğrenip öğrenmediği, hangi sınıfı nasıl tahmin ettiği gibi modelimiz ve hatta veri setimiz hakkında bize geri dönüş sağlamaktadır.Bizim çalışmamızda kullandığımız hata matrisleri ise problem tanımındaki 4 adet sınıfımız ve etiketsiz arka plan görüntülerinin oluşturduğu sınıflarla ile birlikte 5x5'lik bir boyuttadır.

*Hassasiyet (Precision)*Yapılan tahminlerin ne kadarının gerçek etiketlerle uyuştuğunun bir göstergesidir.

$$Hassasiyet = \frac{TP}{TP + FP}$$

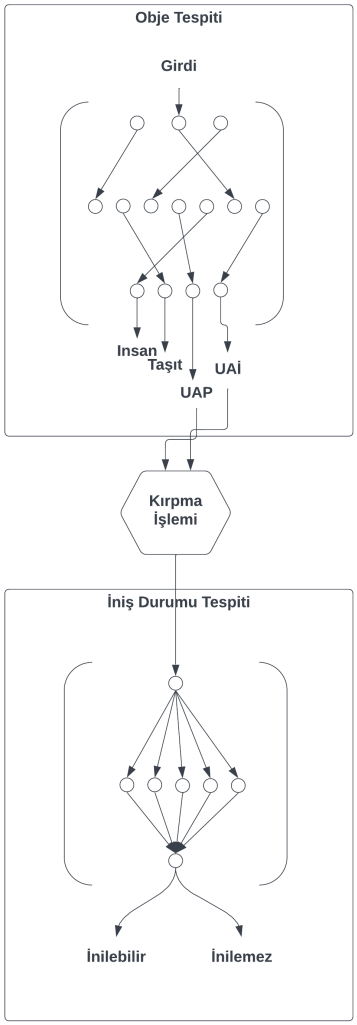

Figure 12: Genel Mimari

Anımsama (Recall) Pozitif durumların ne kadar başarılı tahmin edildiğini gösterir.

$$Anımsama = \frac{TP}{TP + FN}$$

F1 Oranı, hassasiyet ve anımsama değerlerinin harmonik ortalamasını göstermektedir. Basit bir ortalama yerine harmonik ortalama olmasının sebebi uç durumları da dikkate alabilmek içindir. Aşağıda verilen formülle hesaplanır.

$$F1 = \frac{2 * Hassasiyet * Anımsama}{Hassasiyet + Anımsama}$$

Güvenilirlik Oranı (Confidence Score) Modelimizin nesne sınıflandırmasının ne kadar güvenilir olduğunu gösterir. Bu metrik kullanılarak belli bir eşiğin altındaki bazı tespitler göz ardı edilebilir, bu sayede modelin verdiği sonuçlar daha emin bir şekilde kullanılabilir. Bu eşik problem tanımına göre değişmekle beraber yarışma özelinde bizim amacımız yüksek bir ortalama kesinlik

		Var olan Durum	
		Pozitif Durumlar	Negatif Durumlar
Tahmin	Pozitif	TP	FP
	Negatif	FN	TN

Figure 13: Örnek bir hata matrisi

değerlerinin ortalaması (mAP) değeri olduğu için ona göre bir güvenilirlik eşiği seçilmiştir.

Ortalama Kesinlik Değerlerinin Ortalaması (mean average precision, mAP) Nesne tespiti uygulamalarında kullanılan, modelin lokalizasyon konusundaki başarısını ölçmek için kullanılan bir değerdir. Bir sınıfa ait ortalama kesinlik oranı (AP), o sınıfa ait olan tüm nesnelerin IOU oranlarının ortalaması alınarak hesaplanır. IOU oranı ise nesnenin gerçek alanı ile tahmin edilen alanının kesişiminin birleşimine oranı olarak hesaplanır. Her sınıf için AP değerleri elde edildikten sonra modelin ortalama kesinlik değerlerinin ortalamasını (mAP) bulmak için tüm sınıflara ait AP değerlerinin ortalaması alınır.

$$\boldsymbol{mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k}$$
$$\boldsymbol{AP_k = \text{ the AP of class } k}$$
$$\boldsymbol{n = \text{ the number of classes}}$$

Figure 14: mAP değerinin hesaplanması

Modellerin Seçimi

YOLO Eğitimi Araştırmalarımız sonucunda yarışma kapsamındaki nesne tespiti problemine en uygun algoritmaların YOLOv5, YOLT [15], YOLOv7 ve TPH-YOLOv5 olduklarını gördük. Bu yöntemler arasında bir seçim yapabilmek için 2. bölümde anlatıldığı şekilde oluşturduğumuz eğitim ve doğrulama veri setlerimiz ile eğitimlerimizi tamamladıktan sonra doğrulama verilerini kullanarak hassasiyet (precision), anımsama (recall) ve ortalama kesinlik değerlerinin ortalaması (mean average precision, mAP) değerlerini hesapladık ve bu değerlere bakarak bahsedilen algoritmaları karşılaştırdık. Şekil 15'te görüleceği üzere YOLOv5 ve TPH-YOLOv5 oldukça benzer bir performans gösterirken YOLT ve YOLOv7 beklenen eşikleri geçemediği için çalışmalarımızı YOLOv5 ve TPH-YOLOv5 üzerinde yoğunlaştırdık. Bu iki yöntemin performansını karşılatırmak amacıyla aynı doğrulama veri seti kullanılarak oluşturulan hata matrislerini (confusion matrix) inceledik. Şekil 16'daki görüntülere baktığımızda iki algoritma da çok benzer performans göstermesine rağmen TPH-YOLOv5'in problemimizin en can alıcı noktası olan insanları ayırt

etmede daha başarılı olduğunu fark ettik. Bu nedenle TPH-YOLOv5'i ana nesne tespit algoritmamız olarak belirledik.

Figure 15: Nesne Tespiti Algoritmalarının Karşılaştırması

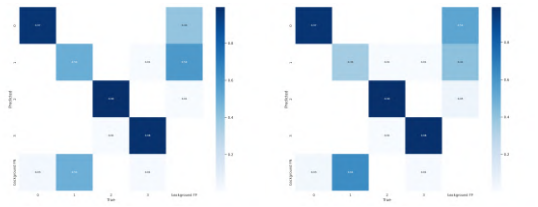

Figure 16: Sol:TPH Sağ:YOLOv5 ile elde edilen hata matrisi

Sistemimizin temelini belirledikten sonra bu yöntemi en iyi sonuçlar verecek şekilde ayarlamak için hiperparametreleri değiştirerek deneyler yaptık. Şekil 17'de görülebilen bu deneylerde Deney 1 geniş derinlikte bir ağ ve 1280 piksel boyutunda resimler, Deney 2 orta genişlikte bir ağ ve 1536 piksel boyutunda resimler, Deney 3 ise orta genişlikte bir ağ ile 1280 piksel boyutunda resimler kullandı. Grafiklerden ve hata matrislerinden görülebileceği üzere Deney 3'teki hiperparametreler en iyi sonuçları verdiği için bu ayarlarla ana sistemimizin kurulumunu tamamladık.

CNN Eğitimi Ana modelimiz beklediğimiz başarıyı göstermeye başladıktan sonra iniş durumu tespiti için kullandığımız evrişimsel sinir ağı (CNN) için deneylerimize başladık. Bu aşamada birçok CNN mimarisi içinden altı tanesini seçerek iniş durumu için oluşturduğumuz veri seti ile eğittik ve doğrulama veri seti üzerindeki doğruluk (accuracy) ve kayıp (loss) durumlarını inceledik. Şekil 19'de görülebileceği üzere SqueezeNet mimarisi en iyi sonuçları gösterirken aynı zamanda da hafif olmasıyla problemimizin çözümü için en iyi seçim oldu. Bu noktada elimizdeki iniş alanı görüntülerinin sayıca az ve sınıfça dengesiz olmasından dolayı modelimizin verilen

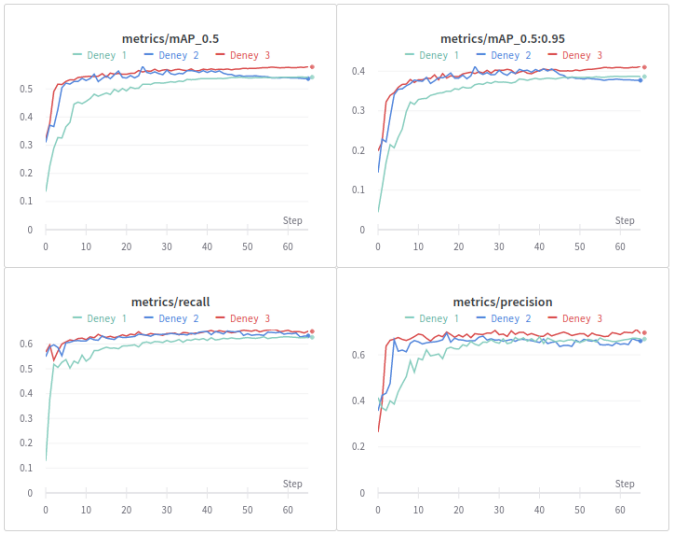

Figure 17: Hiper Parametre Deneyleri

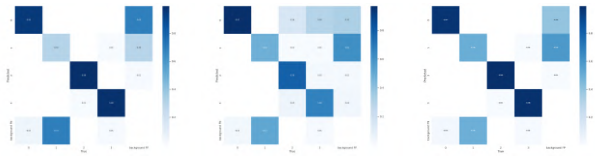

Figure 18: Deney 1,2,3 ile elde edilen hata matrisi

görüntüleri aşırı öğrendiğini ve genelleme yapamadığını fark ettik. Bu sıkıntıyı aşmak amacıyla elimizdeki UAP ve UAİ görüntülerinden oluşan veri setine 2. bölümde açıklandığı gibi veri zenginleştirme işlemi uyguladık. Bu adımın ardından, Şekil 20'de görüleceği üzere incelediğimiz metriklerde düşüş gözlense de yaptığımız testlerde aşırı öğrenme sorununun ortadan kalktığını gördük. Tüm bu çalışmalar sırasında Google Colab servisi aracılığıyla Tesla P-100 grafik kartlarını kullandığımız için eğitim ve doğrulama adımları normalden çok daha kısa sürdü, bu sayede kısa zamanda daha çok deney yapma imkanımız oldu.

4. Sonuçlar

Elimizdeki anımsama (recall), hassasiyet (precision) ve ortalama kesinlik değerlerinin ortalaması (mAP) verilerine baktığımız zaman **TPH-YOLO** nun anımsama ve hassasiyet metriklerindeki başarısı ortadadır. YOLOv5'e göre mAP skorunun daha düşük olmasının sebebini ise temelde daha iyi genelleme yapabiliyor olmasına bağlıyoruz. Bizim aslında istediğimiz şey **insan** sınıfında modelin çok daha başarılı olmasını sağlamaktı. Bu noktada TPH-YOLO küçük objeler için daha başarılı olduğundan ana modelimiz olarak TPH-YOLO'yu tercih ettik. Tablodaki mAP skoru bütün sınıfların ortalama mAP skorunun ortalaması alınarak hesaplanmıştır.

Evrişimsel sinir ağları kıyaslandığındaysa **SqueezeNet**

Table 1: Yöntem Kıyaslaması

Metod	Precision	Recall	mAP_0.5	Accuracy
YOLT	0.6177	0.5063	0.4072	-
YOLOv7	0.6347	0.5465	0.4884	-
YOLOv5	0.6916	0.631	**0.5872**	-
TPH-YOLO	**0.704**	**0.6496**	0.5636	-
CNN-AlexNet	-	-	-	0.941
CNN-Inception	-	-	-	0.9595
CNN-Vgg	-	-	-	0.9723
CNN-DenseNet	-	-	-	0.9723
CNN-ResNet	-	-	-	0.9744
CNN-SqueezeNet	-	-	-	**0.9806**

Figure 19: CNN Mimarilerinin Karşılaştırması

Figure 20: Veri Zenginleştirme İşleminin Sonuçları

doğruluk (accuracy) metriğinde en iyi sonucu üretiyor. SqueezeNet'in en güzel yanı aynı zamanda boyut olarak çok küçük olması ve kendisini çok mobil bir ağ haline getirmesidir. Böylelikle küçük sistemlere entegre edilebilmekte ve hafif çalışmaktadır. Bu açıdan ana modelin performansını süre bazında önemli ölçüde etkilememektedir.

5. Değerlendirme

Uçan arabalar için geliştirilen sistemlerdeki obje tespiti çalışmalarında en önemli hususlar yüksek doğruluk ve hızdır. Bizim geliştirdiğimiz mimari ve deneylere baktığımızda hep daha hızlısını ve daha güvenilir olana erişmeye çalıştık. Bu açıdan elimizdeki probleme en uygun model olan **TPH-YOLOv5**'i kullandık. Deneyler kısmında bahsetmesek de SAHI [16] ile tespit çalışmalarında da bulunduk hız açısından çok yavaş kaldığı için bunu çalışmalarımıza katmadık. Diğer bir yandan tespit edilen iniş alanlarının uygunluk durumlarının kontrolü ise bir sınıflandırma problemiydi. Bu probleme en uygun hızlı ve mobil bir ağ olan **SqueezeNet**'i tercih ettik.

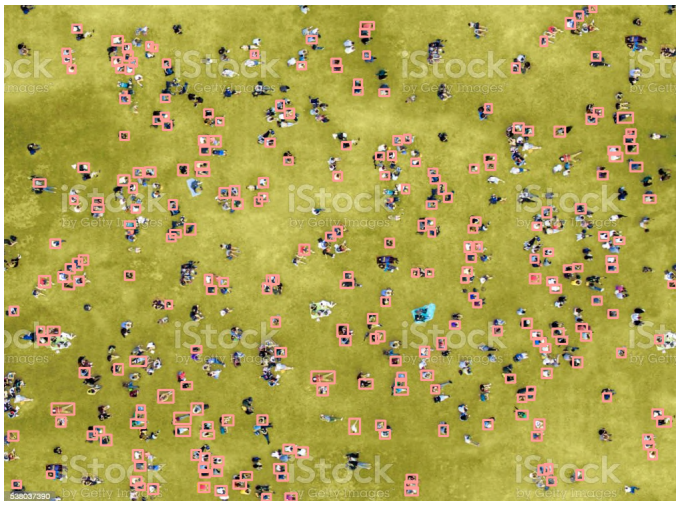

Figure 21: YOLOv5

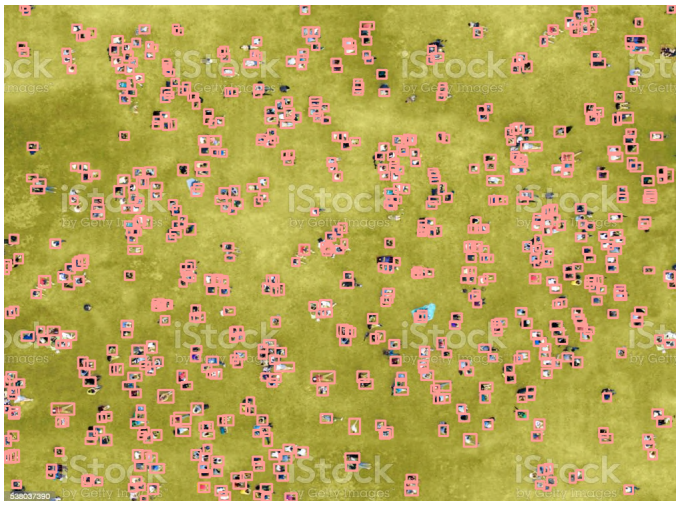

Figure 22: TPH-YOLOv5

Elde ettiğimiz sonuçlar iç açıcıydı. Bunlara ek olarak yarışmanın 2. oturumunda paylaşılan veri seti üzerinde doğrulama (validation) yaptık. Bu doğrulamaya göre

0.84 mAP_0.5 skoru ile gerçek zamanlı uygulamalarda uygulanabilecek bir sistem oluşturduğumuzu gördük.

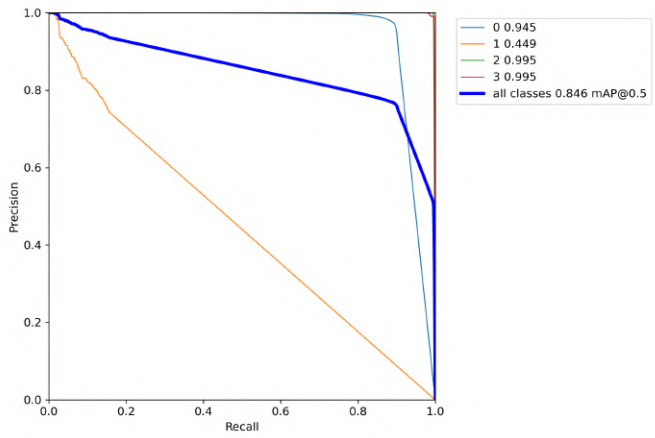

Figure 23: Yarışma 2. oturum sonuçları

Tartışma

YOLO ve TPH her ne kadar uyum içerisinda çalışsa da TPH'taki Dönüştürücü Kodlayıcı Blok (Transformer Encoding Blok) global özellikleri de yakaladığı için objenin etrafındaki ortama göre de çıkarımlar yapıyor. Bu sebeple araba yolunda yürüyen bir yayanın motora benzemesinden dolayı araç sanılması, çimde oturan insanlarla küçük ağaçların karıştırılması, çöp konteynırı direk gibi objelerle insanların karıştırılması ve inşaat yerlerindeki büyük dikdörtgen prizma şeklindeki kutuların araç ve tırlarla karşılaştırılması gibi nadir rastlanan durumlarından ortadan kaldırılması adına **loss fonksiyonu** değiştirilebilir. *"Balanced MSE for Imbalanced Visual Regression"* [17] makalesinde de bahsedildiği üzere dengesiz eğitim veri seti ve dengeli test seti için çok güçlü bir yaklaşım olan **Balanced MSE** eğitim sürecinde kullanılırsa bu tarz durumların da ortadan kaldıralabileceği düşüncesindeyiz.

6.Kaynakça

[1] Dasiopoulou, S., Mezaris, V., Kompatsiaris, I., Papastathis, V.-K. ., & Strintzis, M. G. (2005). Knowledge-assisted semantic video object detection. IEEE Transactions on Circuits and Systems for Video Technology, 15(10), 1210–1224. https://doi.org/10.1109/tcsvt.2005.854238

[2] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. https://doi.org/10.1109/cvpr.2001.990517.

[3] Dalal, N., Triggs, B., Schmid, C., Soatto, S., & Tomasi, C. (2005). Histograms of Oriented Gradients for Human Detection. IEEE Computer So-Ciety, 1, 886–893. https://doi.org/"¿10.1109/CVPR.2005.177.

[4] Felzenszwalb, P., Girshick, R., McAllester, D., & Ramanan, D. (2013). Visual object detection with deformable part models. Communications of the ACM, 56(9), 97. https://doi.org/10.1145/2500468.2494532.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenetclassification with deep convolutional neural networks,"in Advances in neural information processing systems, 2012,pp. 1097–1105.

[6] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2014.81.

[7] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[8] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[9] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[10] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

[11] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[12] Ultralytics (2020), YOLOv5, https://github.com/ultralytics/yolov5, Erişim Tarihi: Aralık 2021

[13] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696.

[14] Zhu, X., Lyu, S., Wang, X., & Zhao, Q. (2021). TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 2778-2788).

[15]Van Etten, A. (2018). You only look twice: Rapid multi-scale object detection in satellite imagery. arXiv preprint arXiv:1805.09512.

[16] Akyon, Fatih Cagatay, et al. “Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection.” ArXiv:2202.06934 [Cs], 15 Feb. 2022, arxiv.org/abs/2202.06934.

[17] Ren, Jiawei, et al. “Balanced MSE for Imbalanced Visual Regression.” ArXiv:2203.16427 [Cs], 30 Mar. 2022, arxiv.org/abs/2203.16427. Accessed 14 Aug. 2022.