MİNİK ŞEF TAKIMI

- Obje tespit modeli Yolov5.(Bu algoritmanın seçilme sebebi hızlı eğitim, hızlı inference ve yüksek başarı oranıdır. Ayrıca PyTorch üzerine kurulu olması sebebiyle değişiklik ve güncelleme yapmak kolay ve anlaşılır olmaktadır.)

- Obje takibi modeli olarak eski adı DeepSort yeni adı StrongSort olan bir pytorch temelli kütüphane kullanılmıştır. Bu model Osnet altyapısını kullanarak hızlı bir şekilde takip yapabilmektedir. Osnet'e alternatif olarak araladında Resnet gibi modellerin de bulunduğu 25 model kıyaslanmıştır. Hız ve başarı dengesi açısından en uygun modelin Osnet olduğuna karar verilmiştir.

-Eksik tespit toleransı için Opencv temelli KCF Tracker algoritması kullanılmıştır.

- Veri seti olarak "DOTAv2, Visdrone" kullanılmış. (DOTAv2 ve Visdrone veri setlerinin seçilme amacı kendi veri setimizi üretmek için temel bir model oluşturmaktır. Bu iki veri setinin yarışma içeriği ile benzerliği sebebiyle başlangıç veri seti olarak seçilmiş ve daha sonra da veri setinde kalmasına karar verilmiştir.)

- Stok videolar Pexels ve Envato sitelerinden indirilmiştir. Drone kullanılarak alınan görüntülerin amacı ise Türkiye'yi daha iyi tanıtma amaçlarıymış.

EFLATUN AI TAKIMI

-DOTA Dataset, VisDrone, Eradataset ve adı bilinmeyen Japonya'da çekilmiş bir dataset'ten yararlanılmış.(Bu veri setlerinin seçilme sebebi çeşitlilik, video açısı ve yüksekliğidir.)

-Obje tespitinde Yolov5 modeli kullanılmış.(Yolov5 modeli, eğitim ve süresinin düşüklüğü sebebiyle tercih edilmiştir. Ayrıca takımın denediği 5 farklı model arasından en iyisidir.)

-Çeşitli videolar üzerinde algoritma çalıştırılmış ve tek girdiye göre çoklu girdinin insan gibi ufak objelerde yaklaşık olarak 40% daha başarılı olduğu gözlemlenmiştir. Yapılan testlere göre çoklu girdi yöntemi bu tip bir obje tespit görevi için uygun bir yöntem olduğu saptanmış.

-Yaşadıkları tek sorun görüntüleri tekrardan birleştirme algoritması olmuş. (Çünkü görüntüleri kırparken farklı bilgileri de saklamak gerekiyordu ve çok fazla görüntü için bu karmaşık bir işlem oluyordu. Ayrıca daha fazla parçaya bölmek istersek yöntemin uygulanabilirliği azalıyordu.) Bu sorunu aşmak için Python üzerinde bir Class oluşturulmuş ve görüntü kırpma çıktılarını bu Class içerisine kaydedilmiş. Böylece işletim sistemlerinde de kullanılan Metadata içeren bir görüntü objeleri olmuş. Bu da onlara normalizasyon denklemlerini kolay implemente etme ve ölçeklenebilir bir sistem kurma imkanı vermiş.

SPATIUM VISION TAKIMI

- AYRICA PAYLAŞILACAKTIR !

SEMRUK TAKIMI

- Gelişen süreçte RetinaNet mimarisinden istenilen sonuç elde edilememesinden dolayı, ön tasarım raporunda geliştirilecek olan RetinaNet mimarisi, YOLOR mimarisi ile değiştirilmiştir. (YOLO, konvolüsyonel sinir ağlarını (CNN) kullanarak nesne tespiti yapan bir algoritmadır ve görüntülerdeki nesneleri ve bu nesnelerin koordinatlarını aynı anda tespit etmektedir.)

- Yolor mimarisi kullanılmış. (Bunun nedeni, aynı türe ait fakat farklı boyut ve şekillerde yer alan nesnelerin tespitinde, modelin her görüntüde benzer başarıyı gösterememesidir. Şekil 2'de örnek bir görsel eklenmiş ve 'insan' nesnesi olmayan yapılar 'insan' şeklinde etiketlenirken aynı zamanda iş araçları da 'taşıt' olarak etiketlenmemiştir. Özellikle aynı sahne görüntüleri üzerinde doğruluk değerlerindeki büyük farklılık (%30-%80), modele olan güveni azaltmış ve farklı çözümler aranmıştır. Geliştirilmesi planlanan modelin performansının değerlendirilmesi ve karşılaştırılması amacı ile eş zamanlı olarak kullanılan YOLOR [11] ve Faster R-CNN [12] model performanslarının çok daha iyi sonuç vermesi (%80~) farklı model seçimi ve çözüm yönteminin değiştirilmesi açısından değerlendirilmiş ve uygun bulunmuştur.)

- VEDAI, AU-AIR, Okutama Actions, VisDrone ve VAID veri setlerinden eşit sayıda 500'er adet görüntü alınmıştır.(Eşit sayıda görüntülerin alınma nedeni, farklı veri setlerin, konum, gölge, ışık, araç ve insan tipi vb. özelliklerinin model eğitiminde yanlılığa neden olmasını engellemektir.)

T-KOD SOFTWARE TAKIMI

- Ön tasarım raporunda belirtmiş oldukları algoritmalar (CNN ve YOLOv5) olmasına rağmen testler sonucunda yarışmaya YOLOX kullanarak devam etme kararı alınmıştır.(YOLOX algoritması diğer mimarilere göre oldukça hızlı çalışması ve sonuç üretmesi bu fikirde devam edilmesine sebep olmuştur.)

- VisDrone veri seti ağırlıklı kullanılmış. (Bunun nedeni teknofestin verdiği simülasyon görüntülerine daha benzer veriler elde etmeleri.) Aynı zamanda şartnamede belirtilen yarışmada verilecek görüntülerin sahip olduğu hava olayları (karlı, yağmurlu, güneşli sisli), 60 ve 90 derece açı kriterleri, kısmen görünen taşıt ve insan kareleri, aynı ortamda bulunan birden fazla çeşitli taşıt ve insan gölgeleri özelliklerine de daha çok uymaktadır. Model eğitimine destek için Generative Adversarial Networks (GAN) algoritmasıda kullanılarak eldeki veri setini hem çoğaltmış hem de örnek videodakine benzer daha çok veri elde edebilir böylece daha tutarlı bir model eğitebiliriz düşüncesine sahipler.

- Nesne tespiti yapmak için Python yazılım dili ile birlikte CNN (Convolutionnal Neural Networks) algoritmasını kullanan Yolov5 Kullanılması planlanmıştır. (YOLO Kullanılmasının temel sebeplerinden biri kullanımının basit olması hızlı ve tutarlı çalışması aynı zamanda şu anda piyasada bulunan son teknolojilerden biri olmasıdır. YOLO algoritmasının diğer algoritmalardan daha hızlı olmasının sebebi resmin tamamını tek seferde nöral bir ağdan geçiriyor olmasıdır.)

- Ancak yapılan testler sonucunda Yolov5, eğittiğimiz veri setindeki taşıtlarda yüksek performansla tespit yapılmasına rağmen Teknofest'in örnek olarak paylaştığı videoda istediğimiz sonucu verememiştir. Bu yüzden Yolov5 Modelinden daha gelişmiş olan YoloX Modeli kullanılacaktır.(YoloX hem performans hem de tutarlılık olarak Yolov5 ten daha gelişmiş bir düzeydedir.)

- YOLOV5'in daha üst modeli olan YOLOX'in kullanılmasına ek olarak SAHI ve Deep Sort gibi yardımcı algoritmalara daha çok ağırlık verilmiştir.

- CNN, girdi verisi olarak görüntüleri kullanan bir derin öğrenme algoritmasıdır. Görüntülerin özelliklerini (features) kullanarak görüntüleri birbirinden ayırt etmektedir diğer bir deyişle sınıflandırma işlemini gerçekleştirmektedir.

- Deep Sort (Derin sıralama) algoritması tespit edilen nesnenin bir sonraki frame de bulunduğu noktayı baz alarak nesnenin gitti yönü tespit edip takibini gerçekleştiren bu sayede daha yüksek tutarlılıkta yapay zekanın nesne tespiti yapmasını sağlayan bir algoritmadır. Eğittiğimiz modele yardımcı olarak nesne tespit yazılımına Deep Sort algoritması kullanılmıştır.

- Başta python kullanılması ön görülsede c++ dilinden devam edilmiştir.(Bunun sebebi, C++ yazılım dilinin, Python yazılım diline göre 8 kat daha hızlı çalışmasıdır. Bu avantajla birlikte, nesne tanıma yazılımına daha çok yardımcı algoritma eklenecek ve performans düşmesi probleminin önüne geçilecektir.)

- JSON dosyası olarak kaydedilme işlemi C++ yazılım dili ve OpenCV kütüphanesinin yardımı ile Linux işletim sisteminde yapılacaktır.

AI LAB-SEMRÜK TAKIMI

- Veri setinin çeşitlendirilmesi amacıyla seçilen bazı frameler üzerine Adobe Photoshop CS6 ile sis, yağmur ve kar efektleri verilmiştir.(güzel fikir)

- Sistem gereksinimleri düşük ve kullanımı kolay olan YOLOv5m6 nesne tespit algoritması eğitim süreci için seçilmiştir. Yapay zeka eğitimi için Google Colab Pro üzerinden eğitim yapılmasına karar verilmiştir. Google Colab (Google Colaboratory), yapay zeka ve derin öğrenme projeleri üzerinde çalışanlar için etkileşimli, tamamen bulut tabanlı, kullanımı kolay ve ortak çalışmaya dayalı bir programlama ortamıdır. Yapılan eğitimlerde Google Colab Pro tarafından sağlanan Tesla P100 16 GB GPU ve 25 GB RAM donanımları kullanılmıştır.

- Veri seti örnekleri T3 vakfı, teknofest eskiden paylaşılan videolar,drone görüntüleri vs. Paylaşılan bu veri setleri üzerindeki etiketleme işlemi GitHub üzerinden açık kaynak kodlu paylaşılan LabelImg yazılımı ile devam etmektedir.

- Takım görüntü işlemede YOLO (You Only Look Once) algoritmasını tercih etmiş.Ek olarak Non-Maximum Supression algoritması kullanılmış.(Non-Maximum Supression algoritması oluşturulan bounding boxların güven skorlarına bakarak en yüksek güven skoruna sahip bounding box ekrana çizer. Böylece gereksiz kutucuk oluşumunun önüne geçilir.) !örnek fotoğrafı ekstra paylaşacağım.

- YOLO algoritması Mask-RCNN, Detectron gibi maskeleme ile nesne algılaması yapan algoritmalara kıyasla çok daha az yer kaplar.

- Etiket Büyüklük Kontrolü (Label Size Control).(Eğer kullandığımız görüntü işleme algoritması insandan küçük bir araba bulduysa veya arabadan büyük bir insan bulduysa tasarladığımız bu algoritma devreye girmekte ve düzeltmelere başlamaktadır. Arka tarafta insandan küçük araba olamaz ve arabadan büyük insan olamaz gibi karar mekanizmaları çalışmaktadır.)-özgünlük

- İki Algoritmayı Birleştirme (Combination of Two Algorithms). Bu sistemde ise asıl hedeflenen sonuç YOLOv5l6 modelinin araçlar üzerindeki kalitesi ve YOLOv5m6 modelinin ise küçük nesneleri algılamadaki kararlılığını birleştirip ortaya daha kaliteli bir sonuç çıkarmaktır. Sonuç olarak YOLOv5m6 modelinden gelen veriler ile YOLOl6 modelinden gelen verilerin birleştirilip tekrardan bir Non-Maximum Supression işlemi uygulayarak en yüksek güven skoruna ait kutucuk çizdirilecektir.-özgünlük

SÜMERZEKA TAKIMI

- VisDrone veri seti kullanılmış.(Bu veri setine içerisinde barındırdığı fotoğraflar yarışmada istenen formatlara oldukça yakın 8000 fotoğraf bulundurması sebebiyle karar kılınmış.)

- Veri setlerinde kendi etiketlemelerini yaparken “LabelImg” programı ile YOLO modunda etiketleme yapılmış. Tüm veri setleri YOLO formatında depolanmış. Faster-RCNN modeli eğitimi için veri seti kendi yazdıkları kodlar ile dönüştürülmüştür.

- YOLOv5 tercih edilmiş.(Bu başarının en önemli nedenlerinden biri de YOLOV5 in “Backbone” olarak kullandığı Darknet-53 mimarisinin SSD ve Faster RCNN’in kullandığı ResNet-152 mimarisiyle eşit performansı iki kat hızlı bir şekilde elde edebilmesidir.)

- YOLOv5l modeli üzerinde yapılan testler sonucu belirttiğimiz sebeplerden ötürü (nedeni çoklu ve yanlış insan tespiti) “İnsan” sınıfı tespitinde başarı oranı düşük olmaktadır. İncelediğimiz diğer modeller arasında insan sınıfının tespitinde en iyi sonuç veren Faster-RCNN modelinin kullanılmasına karar verilmiş.

- Modelleri ve haberleşme kodları yazılırken Python yazılım dili tercih edilmiş.(Kodların küçük bir kısmı C kodları çoğunluğu Python standart kütüphaneleri ve harici bazı kütüphanelerden oluşmakta.)

- Veri setlerindeki fotoğrafların çoğunluğu yüksek kaliteli olması ve yarışmadaki resimleri arasında bazı nesnelerin fazla küçük olduğundan veya sarsıntılardan dolayı bulanık çıktığını ve modellerinin bu nesneleri algılamakta çeşitli sıkıntılar çektiğini tespit etmişler.(ÖNEMLİ BİR KONU, anlatımları aynen aşağıdaki gibi)
- “*Bunun önlenebilmesi ve modelin nesneleri daha iyi kavrayabilmesi için veri setimizdeki bazı fotoğrafları Median Blur, Gaussian Blur ve Bilateral gibi blur algoritmalarında test ettik. Lakin bu bulanıklaştırma yöntemleri kullanılarak kalitesiz resim etkisini yakalayamadık. Ardından kendi geliştirdiğimiz yöntem ile resim kalitesi istenildiği gibi düşürülmüştür:

- • Öncelikle 8*8 filtre ile ortalama alma yöntemi ile 2 kat küçültülür.
- • Ardından resim pixel kopyalama yöntemi ile tekrar eski boyutuna getirilir. Bu sayede resimdeki ayrıntılar azaltılmıştır.”