YOLO’lar genel olarak: Jetson üzerinde olan Nvidia CUDA mimarisini ve cuDNN kütüphanesini kullandığı için oldukça verimli çalışmaktadır. Diğer sistemlere göre daha hızlı çalışıyor ve doğruluk oranını diğerlerine göre daha yüksek. Ve eğitilmesine, üzerinde değişiklik yapılabilmesine olanak veriyor olması da seçiminde etken. YOLO algoritması doğruluk oranı, Hız, ve takip algoritmalarına uyumluluğu açılarından ve anlık görüntü işleme yapabilmesi diğer algoritmalara göre Uav için daha uygun bir seçenek oluyor. Yani toparlamak gerekirse anlık görüntü işlemede kullanmamız gereken algoritmanın YOLO olduğu çok açık asıl kararlaştırılması gereken hangi versiyonun kullanılacağı, görüntü işlememenin yer istasyonunda mı yoksa uav’nin içerisinde mi yapılacağı ve hangi optimizasyon tekniklerini kullanmamız gerektiği bizim için önemli.

YOLOV3: YOLO’nun bir versiyonu. Darknet53 ağını kullanıyor.En Popüler YOLO algoritması. Önceki versiyonlar üzerinden çevreleme kutuları üzerine obje olmama skoru eklenmesi, backbone network katmanlarına eklenen bağlantılar ile tahminlerini 3 ayrı ayrıntı katmanında da yapması küçük objeler üzerindeki performansını arttırıyor.

YOLOV3-Tiny: YOLO’nun bir versiyonu. K-means gruplama yöntemi kullanarak olabilecek anchor boxların yerini tahmin edip modelin hızını arttırır. Bu optimizasyonlar sayesinde sadece cpu üzerinde de eğitilebilir hale geliyor.

YOLOV4: YOLO’nun bir versiyonu. YOLOV3 ün gelişmiş bir versiyonu. YOLOV3 e göre mAP değerlerinde yüzde 10’luk ve FPS değerlerinde yüzde 12 lik artış elde edilmiş.

YOLOV5: YOLO’nun bir resmi olmayan versiyonu Ultralytics tarafından geliştirildi. YOLOV4 e benziyor ama DarkNet yerine PyTorch framework’ünü kullanıyor. Benchmark testlerinde YOLOV4 ile benzer sonuçlar vermiştir. 3 kısımdan oluşur: spine, head ve neck

YOLOV5s: YOLO’nun bir versiyonu. Diğer v5’lere göre daha az performanslı ama uav donanımlarına daha uygun diye tercih edilmiş.

YOLOV6: YOLO’nun bir resmi olmayan versiyonu Çinli bir şirket olan Meituan tarafından geliştirildi.

YOLOV7: YOLO’nun en yeni versiyonu. Genel olarak fps ve AP değerleri diğer versiyonlara göre daha yüksek. Genel olarak çoğu ticari network ün YOLOV4 ten direkt olarak YOLOV7 ye geçmesi planlanıyor. İçerisinde Aslında YOLOV4 ten sonra çıkan ilk resmi versiyon ve firmaların YOLO algoritmalarından çıkar elde etmek için fazla bir performans artımı olmamasına rağmen yeni YOLO algoritmaları çıkarmalarından dolayı çoğu şirket bu güncellemeyi gerçek bir güncelleme olarak görüyor.

EFFİCENT.DET: Efficent.Net üzerinden PyTorch ile Google tarafından geliştirilmiş bir obje tespit etme modelidir. YOLOV3’ten genel özellikler olarak biraz öndedir.

Mask R-CNN: Yaklaşımımız, bir görüntüdeki nesneleri verimli bir şekilde algılarken aynı anda her örnek için yüksek kaliteli bir segmentasyon maskesi oluşturur. Mask R-CNN adı verilen yöntem, sınırlayıcı kutu tanıma için mevcut dala paralel olarak bir nesne maskesini tahmin etmek için bir dal ekleyerek Daha Hızlı R-CNN'yi genişletir.Mask R-CNN'nin eğitilmesi kolaydır ve 5 fps'de çalışan Daha Hızlı R-CNN'ye yalnızca küçük bir ek yük ekler. Ayrıca, Mask R-CNN'in diğer görevlere genelleştirilmesi kolaydır, örneğin aynı çerçevede insan pozlarını tahmin etmemize izin verir.

YOLOR: YOLOR, mimari ve model altyapısındaki farklılıklar nedeniyle YOLOv1-YOLOv5'ten farklı, nesne algılama için son teknoloji bir görüntü işleme/bilgisayarlı görü algoritmasıdır. YOLOR, “Yalnızca Bir Temsil Öğrenirsiniz” anlamına gelir; YOLO’nun “Yalnızca Bir Kez Bakarsınız” anlamına geldiği 1'den 4'e kadar olan YOLO sürümleriyle karıştırılmamalıdır .YOLOR, “örtülü bilgiyi ve açık bilgiyi birlikte kodlamak için birleşik bir ağ” olarak önerilmiştir. “Yalnızca Bir Temsil Öğrenirsiniz : Çoklu Görevler için Birleşik Ağ” başlıklı YOLOR araştırma makalesinin bulguları , sonuçların örtük bilgiyi kullanmanın faydasını gösterdiğini belirtir.

YOLOX: YOLOX, tek aşamalı gerçek zamanlı bir nesne dedektörüdür. YOLOX'un temel modeli, Darknet53 omurgasına sahip YOLOv3 SPP'dir. YOLOX nesne dedektörü, YOLO ailesinin çok ilginç bir üyesidir. Bazı benzersiz özellik eklemeleriyle YOLOX, son teknoloji modellerle eşit sonuçlar sunabilmektedir.

PPYOLO: PP-YOLO, YOLOv3 tabanlı bir nesne dedektörüdür. Hızın neredeyse değişmeden kalmasını sağlarken, dedektörün doğruluğunu mümkün olduğu kadar artırma hedefine ulaşmak için, temel olarak model parametrelerinin ve FLOP'ların sayısını neredeyse artırmayan çeşitli mevcut hileleri birleştirmeye çalışır.DarkNet-53 omurgasını ResNet50-vd ile değiştirip kullanılmıştır.

RetinaNet: Birden çok ölçekteki nesneleri verimli bir şekilde algılamak için piramit ağı kullanır ve ön plan arka plan dengesizliği sorununu hafifletmek için odak kaybı işlevini sunar.

FAST-RCNN: Tüm görüntüyü ve bölge tavsiyelerini CNN mimarisinde girdi olarak tek bir ileri yaymada alan algoritmadır.Fast R-CNN’de odaklanılması gereken en büyük geliştirme R-CNN’de kullanılan CNN, SVM ve Regressor’u kombinlemesidir. Bu kombinleme ile muazzam bir performans üstünlüğü elde etmektedir.

SSD: SSD'nin iki bileşeni vardır: omurga modeli ve SSD kafası. Omurga modeli genellikle bir özellik çıkarıcı olarak önceden eğitilmiş bir görüntü sınıflandırma ağıdır. Bu, tipik olarak, tamamen bağlı sınıflandırma katmanının kaldırıldığı, ImageNet üzerinde eğitilmiş ResNet benzeri bir ağdır.Bu nedenle, daha düşük bir çözünürlükte de olsa görüntünün uzamsal yapısını korurken, girdi görüntüsünden anlamsal anlam çıkarabilen derin bir sinir ağıyla baş başa kalırız. ResNet34 için omurga, bir giriş görüntüsü için 256 7x7 özellik haritasıyla sonuçlanır.Son katman aktivasyonlarının uzamsal konumlarındaki sınırlayıcı kutular ve nesne sınıfları olarak yorumlanır.

|  |  |
| --- | --- |
| Modeller | Kullanım Sayıları |
| YOLO (Model Belirtilmemiş) | 2 |
| YOLOV3 | 2 |
| YOLOV3-Tiny | 1 |
| YOLOV4 | 3 |
| YOLOV5 | 4 |
| YOLOV5s | 2 |
| YOLOV6 | 0 |
| YOLOV7 | 0 |
| EFFİCENT.DET | 0 |
| Mask R-CNN | 0 |
| YOLOR | 1 |
| YOLOX | 0 |
| PPYOLO | 0 |
| Retina.Net | 0 |
| FAST-RCNN | 0 |
| SSD (MobileNetSSDv2 ve MobileDets kullanılmış) | 2 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Modeller | Anlık Mı |
| YOLO | YES |
| YOLOV3 | YES |
| YOLOV3-Tiny | YES |
| YOLOV4 | YES |
| YOLOV5 | YES |
| YOLOV5s | YES |
| YOLOV6 | YES |
| YOLOV7 | YES |
| EFFİCENT.DET | YES |
| Mask R-CNN | NO |
| YOLOR | YES |
| YOLOX | YES |
| PPYOLO | YES |
| Retina.Net | NO |
| FAST-RCNN | NO |
| SSD (MobileNetSSDv2 ve MobileDets kullanılmış) | YES |

Kaynakça:

<https://paperswithcode.com/sota/object-detection-on-coco>

<https://dergipark.org.tr/tr/download/article-file/2406452>

<https://models.roboflow.com/object-detection>

<https://pjreddie.com/media/files/papers/yolo_1.pdf>

<https://ieeexplore.ieee.org/abstract/document/7832305>

<https://teknofest.org/tr/competitions/competition/33>

<https://dergipark.org.tr/tr/pub/ejosat/issue/54511/742789>

<https://kaanugurluoglu123.medium.com/nesne-tan%C4%B1ma-algoritmas%C4%B1-faster-r-cnn-nedir-1738f0cca8b7>