YOLO(You Only Look Once) Algoritması gerçek zamanlı nesne takibi yapabilen, halihazırda kullanılan algoritmalara oranla çok hızlı çalışan ve bunun için CNN(Evrişimsel Sinir Ağları) kullanan bir algoritmadır. 2016 yılında YOLOv1 numarasıyla ilk kez kullanıma sunulmuştur. Şu an ise YOLOv10 versiyonu kullanılabilmektedir. YOLO yalnızca diğer algoritmalara oranla değil, kendi versiyonları içerisinde de hızına hız katmıştır.

YOLO algoritmasının çalışma mantığı ve neden hızlı olduğunu açıklamak gerekirse; Algoritma nesne tanımlamayı adından da anlaşılacağı gibi tek seferde yapmaktadır. Diğer algoritmalardan farklı olarak nesne takibini yaparken görüntüyü bölgelere ayırır (Grid). Gridler içerisindeki görüntülere ait nesneleri tek tek kutucuklara (Bounding Box) koyar. Her bir bounding box içerisinde nesne olma olasılığını hesaplar ve Güven Skoru (Trust Score) çıkararak bize kutucuklar içerisindeki nesnelerin yüzde kaç tahmin edilen nesneyle benzerlik gösterdiğini söyler. Bu kutucuklara ait değerler yalnızca güven skoruyla sınırlı değildir. Ayrıca her bir kutunun içerisinde (eğer nesne varsa) orta noktası bu kutunun içerisinde mi diye kontrol eder ve son olarak da uzunluk,yükseklik ve bağıl sınıf olasılığının tespitini yapar.(Bağıl sınıf modelde ne kadar sınıf olduğuna göre değişkenlik göstermektedir.)

Her bir kutucuğun kendi içerisinde aranan nesneyi bulma olasılığı vardır bu da algoritmada karışıklığa neden olmaktadır. Bunun önüne geçmek için bounding box’lardan güven skoru en yüksek olan kutucuğu (non-maximum suppression algorithm) baz alır. Temel mantığıyla YOLOv1 böyle çalışmaktadır. Aynı görüntü içerisinde birden fazla nesnenin orta noktasının tespiti yapıldığı durumlarda YOLOv2 algoritması ortaya çıkmıştır.

Yolov2 algoritmasında anchor(çapa) kullanılarak birden fazla nesnenin orta noktası tespit edildiğinde her nesneye özel kutucuklar oluşturur ve her kutucuğun hesaplanmasını ayrı ayrı yapar. (Güven skoru, yükseklik,uzunluk,bağıl sınıf olasılığı). YOLOv2 algoritmasında Darknet-19 derin ağ mimarisi kullanılmaktadır. Bu ağ mimarisine göre CNN ve nesne takibinde performansı arttırmak için YOLOv1’e ek olarak 11 katman daha eklenmiştir.

Görüntüdeki küçük nesnelerin tespitinde eksiklik olduğu hissedildiğinden YOLOv3 gereksinimi ortaya çıkmıştır. YOLOv2’nin performansı küçük nesnelerin tespiti için yeterli olmamıştır. Katman sayısını arttırarak Darknet-53 mimarisini kullanmaya başlamışlardır. Fazla katman, hesaplanması gereken daha fazla iş olduğundan hız konusundan taviz verilmiştir. YOLOv3 ile her katmanda Batch Normalization işlemi yapılmış Leaky ReLU aktivasyon fonksiyonları kullanılarak model eğitim süreci hızlandırılmış, öğrenme kapasitesi arttırılmış ve daha derin ağlarda kararlılık sağlanmıştır. Ayrıca farklı ölçeklerde nesne tespiti yapabilmek için Spatial Pyramid Pooling fonksiyonu da eklenmiştir. YOLOv3’ün öne çıkan özelliği ise 3 farklı ölçekte tespit yeteneği olmuştur.

YOLOv4 versiyonu ile artık Darknet ağ mimarisinden vazgeçilmiş ve PyTorch kullanılmaya başlanmıştır. Buna göre K-Ortalama Kümesi kullanılarak bağlantı kutuları oluşturuldu ve GHM kaybından yaralanıldı.Bu geliştirmeler ile doğruluk ve hız oranında artış sağlandı. YOLOv3 versiyonu gibi eğitim sürecinde kullanılan tekniklerini arttırarak Self-Adversarial Training ve DropBlock Regularization gibi daha iyi öğrenme teknikleri kullanılmıştır. YOLOv4 ile gerçek zamanlı algılama öncelik haline getirilmiştir ve sadece bir GPU üzerinde eğitimini yürütür. Google Brain tarafından geliştirilen EfficientDet ile ağın boyun kısmındaki blokların en iyi formunu bulmak için sinir mimarisini kullanmıştır.