Gazi Üniversitesi
Fen Bilimleri Dergisi
PART C: TASARIM VE TEKNOLOJİ

Gazi University
Journal of Science
PART C: DESIGN AND TECHNOLOGY

GU J Sci, Part C, 12(2): 691-706 (2024)

Farklı YOLO Algoritmalarının Gerçek Zamanlı İHA Tespitine Etkileri ve Karşılaştırmalı Analizi

Hediye ORHAN[1*] Ferda Nur ARICI[1]

[1]*Necmettin Erbakan University, Department of Computer Engineering, Konya, Turkey*

Makale Bilgisi

Araştırma makalesi
Başvuru: 19/12/2023
Düzeltme: 12/03/2024
Kabul: 29/03/2024

Anahtar Kelimeler

İha
Görüntü işleme
Derin öğrenme
Opencv
Yapay zekâ
Yolo

Article Info

Research article
Received: 19/12/2023
Revision: 12/03/2024
Accepted: 29/03/2024

Keywords

Uav
Image processing
Deep learning
Opencv
Artificial intelligence
Yolo

Grafik Özet (Graphical/Tabular Abstract)

Bu çalışmada, kullanım kolaylığı için tasarlanan ara yüz örneği, Şekil A'da sunulmaktadır. Bu ara yüz, çalışmanın başarıyla tamamlanan kısmını kullanıcıya net bir şekilde iletmek için tasarlanmıştır. / In this study, an example of the interface designed for ease of use is presented in Figure A. This interface is designed to clearly communicate to the user the successfully completed part of the work.

Şekil A: *Tasarlanan ara yüz /* ***Figure A:*** *Designed interface*

Önemli noktalar (Highlights)

- *İHA'nın otonom görevi sırasında gerçek zamanlı olarak hedef tespit etmesi üzerine odaklanılmıştır. / The focus is on real-time target detection of the UAV during its autonomous mission.*
- *Hedef İHA'nın tespiti için farklı Yolo algoritmaları kullanılmıştır ve bu algoritmalar karşılaştırılmıştır. / Different Yolo algorithms were used to detect the target UAV and these algorithms were compared.*
- *Gerçek zamanlı hedef tespiti için özgün veri seti oluşturulmuş ve Yolo algoritmaları ile eğitilmiştir. / An original data set was created and trained with Yolo algorithms for real-time target detection.*

Amaç (Aim): *Bu çalışmadaki amaç bir İHA'nın otonom görevi sırasında gerçek zamanlı olarak hedefi tespit etmesini sağlamaktır. / The aim of this study is to enable a UAV to detect the target in real time during its autonomous mission.*

Özgünlük (Originality): *Gerçek zamanlı hedef tespiti için bu çalışmaya özgü veri seti oluşturulmuştur. / A data set specific to this study was created for real-time target detection.*

Bulgular (Results): *Bu çalışmanın özgün veri seti kullanarak geniş kapsamlı bir derin öğrenme tabanlı nesne tespiti algoritmaları karşılaştırması sunması ve bu algoritmaların farklı versiyonlarını içermesi, literatüre önemli bir katkı yapabileceğini göstermektedir. / The fact that this study presents a comprehensive comparison of deep learning-based object detection algorithms using an original data set and includes different versions of these algorithms shows that it can make a significant contribution to the literature.*

Sonuç (Conclusion): *FPS ve doğruluk değerleri üzerinden yapılan analizler neticesinde, yolov4-tiny-tensorrt algoritmasının kullanılmasının daha verimli olduğuna karar verilmiştir. Bu algoritma ile yaklaşık olarak 30 FPS değeri elde edilmiştir, bu da çalışmadaki nesne algılama ve sınıflandırma işlemlerinin hızlı ve güvenilir bir şekilde gerçekleştirilebileceğini göstermektedir. / As a result of the analysis made on FPS and accuracy values, it was decided that using the yolov4-tiny-tensorrt algorithm was more efficient. Approximately 30 FPS value was obtained with this algorithm, which shows that object detection and classification operations in the study can be carried out quickly and reliably.*

*Corresponding author, e-mail: hediyeorhan2015@gmail.com
DOI: 10.29109/gujsc.1406837

GU J Sci, Part C, 12(2): 691-706 (2024)

Gazi Üniversitesi
Fen Bilimleri Dergisi
PART C: TASARIM VE TEKNOLOJİ

Gazi University
Journal of Science
PART C: DESIGN AND TECHNOLOGY

http://dergipark.gov.tr/gujsc

Farklı YOLO Algoritmalarının Gerçek Zamanlı İHA Tespitine Etkileri ve Karşılaştırmalı Analizi

Hediye ORHAN[1*] Ferda Nur ARICI[1]

[1] *Necmettin Erbakan University, Department of Computer Engineering, Konya, Turkey*

Makale Bilgisi

Araştırma makalesi
Başvuru: 19/12/2023
Düzeltme: 12/03/2024
Kabul: 29/03/2024

Anahtar Kelimeler

İha
Görüntü işleme
Derin öğrenme
Opencv
Yapay zekâ
Yolo

Öz

Ülkemizde ve dünyada havacılık sektörü sürekli olarak gelişmektedir. Değişen ve gelişen teknolojiler ile birlikte insansız hava araçları (İHA) da pek çok sektörde farklı amaçlar doğrultusunda kullanılmaya başlanmıştır. İHA'ların kullanım alanlarına; başta askeri uygulamalar olmak üzere, jeolojik ve meteorolojik araştırmalar, doğal afet yönetimi, tarımsal keşifler, ulaştırma, yeryüzünün haritalanması ve üç boyutlu modelleme örnekleri verilebilir. Ülkemizde askeri alanda İHA kullanımı her geçen gün artmaktadır. Bunların başında hedef tespiti, hedef vuruşu ve hedef takibi gelmektedir. Hedef vuruşunda İHA üzerindeki kameralar ile hedef tespit edildikten sonra vuruş gerçekleştirilmektedir. Hedefin doğru tespit edilmesi çok önemlidir. Bir İHA'nın tam otonom görevini gerçekleştirebilmesi için hedefleri tespit edip kaçış manevraları uygulaması gerekmektedir. Bunun için hedef tespitinin doğruluk değeri yüksek olmalıdır ve gerçek zamanlı olarak çalışmalıdır. Bu araştırmadaki amaç bir İHA'nın otonom görevi sırasında gerçek zamanlı olarak hedefi tespit etmesini sağlamaktır. Araştırma amacı doğrultusunda hedef İHA'nın tespiti için farklı Yolo (You Only Look Once) algoritmaları kullanılmıştır. Gerçek zamanlı hedef tespiti için özgün veri seti oluşturulmuştur. Bu veri seti Yolo algoritmaları ile eğitilip, sonuçları karşılaştırılıp, orantılı olarak yüksek doğruluk değeri ve saniyede yüksek görüntü sayısına (frame per second (FPS)) sahip yolov4-tiny-tensorrt algoritması tercih edilmiştir. Algoritmanın mAP (mean average precision) değeri yaklaşık olarak %95'tir. Elde edilen sonuçlar analiz edilmiştir. Böylece gerçek zamanlı hedef tespiti yapılmıştır.

Effects and Comparative Analysis of Different YOLO Algorithms on Real-Time UAV Detection

Article Info

Research article
Received: 19/12/2023
Revision: 12/03/2024
Accepted: 29/03/2024

Keywords

Uav
Image processing
Deep learning
Opencv
Artificial intelligence
Yolo

Abstract

The aviation industry is constantly developing in our country and around the world. With changing and developing technologies, unmanned aerial vehicles (UAVs) have begun to be used for different purposes in many sectors. Usage areas of UAVs; Military applications, geological and meteorological research, natural disaster management, agricultural exploration, transportation, earth mapping and three-dimensional modeling can be given as examples. The use of UAVs in the military field in our country is increasing day by day. The most important of these are target detection, target shooting and target tracking. In target shooting, the hit is carried out after the target is detected with the cameras on the UAV. It is very important to identify the target correctly. In order for a UAV to perform its fully autonomous mission, it must detect targets and perform escape maneuvers. For this, the accuracy of target detection must be high and it must work in real time. The aim of this research is to enable a UAV to detect the target in real time during its autonomous mission. In line with the research purpose, different Yolo (You Only Look Once) algorithms were used to detect the target UAV. A unique data set was created for real-time target detection. This data set was trained with Yolo algorithms, the results were compared, and the yolov4-tiny-tensorrt algorithm, which has a proportionally high accuracy value and high frame per second (FPS), was preferred. The mAP (mean average precision) value of the algorithm is approximately 95%. The results obtained were analyzed. Thus, real-time target detection was achieved.

1. GİRİŞ (INTRODUCTION)

İçerisinde güç sistemi olan, otomatik veya uzaktan kontrol sistemi ile uçurulan pilotsuz hava araçlarına İnsansız Hava Araçları (İHA) denilmektedir [1]. Teknolojinin gelişmesiyle kullanım alanı bir hayli genişleyen İHA'lar ilk olarak askeri amaçla kullanılmıştır. Gelişen teknoloji ile paralel olarak günümüzde İHA'lar tarım, taşımacılık, jeolojik ve

*Corresponding author, e-mail: hediyeorhan2015@gmail.com
DOI: 10.29109/gujsc.1406837

meteorolojik araştırmalar, ticaret gibi pek çok alanda kullanılmaktadır. Bu alanlarda İHA kullanımı otonom olarak hızla ilerlemektedir. Ülkemizde İHA'lar askeri alanda pek çok amaç için kullanılmakta ve bu alandaki başarısını her geçen gün arttırmaktadır. İHA'ların askeri alanda kullanımı hem başarı hem de güvenilirlik açısından son derece önemlidir. TAI-TUSAŞ'ın geliştirdiği ANKA isimli İHA [2], Vestel Savunma Sanayi firmasının ürettiği Karayel isimli silahlı insansız hava aracı (SİHA) [3], Baykar tarafından keşif İHA'sı olarak üretilip sonradan silahlandırılan Bayraktar Taktik İHA, Bayraktar TB2 [4, 5] , kısa pistli gemilerden kalkış ve iniş kabiliyetine sahip SİHA Bayraktar TB3 [6] ülkemizde askeri alanda yapılan İHA çalışmalarına örnek olarak verilebilir.

Ülkemizde ve dünyada askeri gibi çok önemli alanlarda kullanılan İHA'lar, bu çalışmada motivasyon kaynağı olmuştur. Askeri alanda yapılan çalışmaların başında hedef tespiti, hedef vuruşu ve hedef takibi gelmektedir. Hedef vuruşunda İHA üzerindeki kameralar ile hedef tespit edildikten sonra vuruş gerçekleştirilmektedir. Hedefin doğru tespit edilmesi bu noktada çok önemlidir. Yanlış hedefin tespit edilme durumunda hedef vuruşu yanlış olur ve bu durum büyük problemlere yol açar. Bu nedenle hedef tespit algoritmalarının doğruluk değerlerini kesinleştirmek ve arttırmak önemlidir. Hedef tespitinde amaç, hedef İHA'yı geliştirilen algoritma yardımı ile tespit etmektir. Literatürde nesne tespiti üzerine birçok çalışma bulunmaktadır. Bu çalışmalara örnekler literatür taraması kısmında verilmiştir.

Derin öğrenme, yapay sinir ağları ve insan beyninin işlevlerini taklit eden hesaplama sistemleri kavramına dayanır [7]. Derin öğrenme tabanlı algoritmaların kullanılması durumunda bir veri seti gereklidir. Veri seti, tespit edilecek hedefin farklı açılardan görsellerini ve görsellerde hedefin nerede olduğunu belirten etiket dosyalarından oluşmaktadır. Veri setindeki görsellerin pikselleri ve hedefin konumunun doğru etiketlenmesi hedef tespit algoritmasının doğruluğuna etki etmektedir. Kullanılan algoritmalarda belirli parametreler optimum değere ayarlanmalıdır.

Araştırma kapsamında gerçek zamanlı İHA tespitinde yüksek FPS değeri ve yüksek doğruluk değeri elde edilmesi hedeflenmiştir. Bir İHA'nın tam otonom görevini gerçekleştirebilmesi için hedefleri tespit edip kaçış manevraları uygulaması gerekmektedir. Bunun için hedef tespitinin doğruluk değeri yüksek olmalıdır ve gerçek zamanlı olarak çalışmalıdır. FPS değeri algoritmanın doğruluğunu etkilediğinden dolayı gerçek zamanlı görüntü işleme algoritmalarında önem arz etmektedir. Hedef tespitinde optimum algoritmanın seçimi için FPS ve doğruluk değeri parametrelerine dikkat edilmelidir. Hata payı en aza indirilmelidir. Örneğin; İHA üzerinden hedefe atış gerçekleştirilmesi gerektiğinde yanlış bir tespit ve hedef koordinatı atışın başarısız olmasına sebep olur. Bu durum kötü sonuçlara sebebiyet verebilir. Bu nedenle algoritma doğruluğuna çok dikkat edilmelidir.

Bu araştırma, gerçek zamanlı İHA tespiti konusunda algoritmalarının evrimini, araştırmacı tarafından özgün olarak oluşturulan bir veri seti üzerinden değerlendirerek ve inceleyerek, Yolov3'ten başlayarak Yolov8'e kadar olan YOLO (You Only Look Once) versiyonlarını kapsayan geniş bir değerlendirme sunmaktadır. Bu çalışma, özgün bir veri seti kullanarak Yolo algoritmalarının evrimini anlamak ve gerçek zamanlı İHA tespiti alanında en uygun çözümü bulmak isteyen araştırmacılara yönelik kritik bir kaynak olup, algoritma sürümleri arasındaki performans farklarını ayrıntılı bir şekilde ele almaktadır. Çalışmada özgün veri seti olması, çalışmanın güvenilirliğini artırırken, aynı zamanda gerçek dünya koşullarında elde edilen verilerin algoritmaların gerçek performansını daha doğru bir şekilde yansıttığını vurgular. Test sonuçları, Yolo algoritmasının farklı versiyonlarının hem doğruluk hem de hız açısından farklı performanslar sergilediğini göstermiştir. Çalışma, mevcut araştırmalarda genellikle belirli Yolo versiyonlarına odaklanıldığını ve geniş bir karşılaştırmanın eksik olduğunu vurgular. Ayrıca, gerçek zamanlı İHA tespiti konusunda mevcut literatürde Yolov8'in potansiyelini değerlendiren sınırlı sayıda çalışma bulunması üzerinde durarak, bu alanda daha fazla kapsamlı analizlerin ve karşılaştırmaların önemine vurgu yapar. Özetlenirse; araştırmada elde edilen hedef tespit algoritması istenen doğruluk ve FPS değerine ulaşınca uygun işlemci gücüne sahip bir bilgisayar ve kamera üzerinde test edilmiştir. Elde edilen sonuçlar analiz edilmiştir.

Çalışmanın Materyal ve Metod bölümünde, projenin geliştirme aşamaları, özgün veri setinin oluşturulması, veri setinin farklı Yolo algoritmaları ile eğitilmesi, eğitimlerden elde edilen performans metriklerinin değerleri ve eğitim sonucunda oluşan model dosyalarının donanım üzerinde test edilmesi adımları detaylı olarak açıklanmıştır.

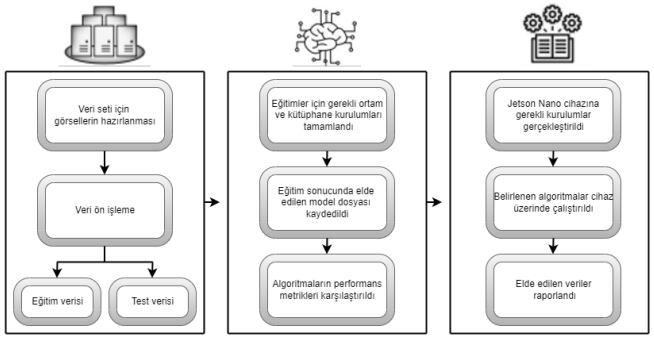
Bulgular ve Tartışma bölümünde, literatürdeki diğer çalışmaların ana konuları incelenmiş ve

mevcut çalışma ile literatürdeki diğer çalışmalar arasındaki artılar ve eksikler değerlendirilmiştir.

Sonuç bölümünde, kullanılmasına karar verilen algoritma ile ilgili bilgiler ve çalışmanın sonuçlandırılması hakkında özetleyici bilgiler yer almaktadır.

2. MATERYAL VE METOD (MATERIALS AND METHODS)

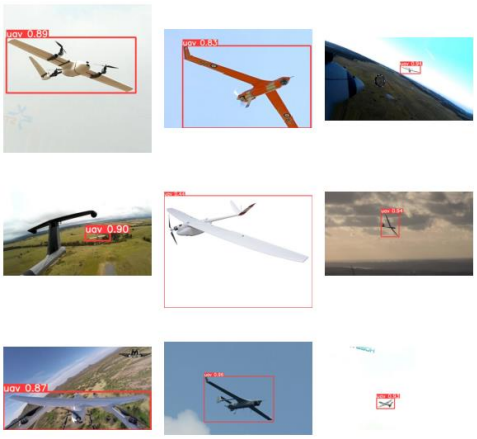
Araştırmada öncelikle derin öğrenme algoritmasında eğitilmek üzere bir veri seti hazırlanmıştır. Hazırlanan veri seti ile birden fazla derin öğrenme algoritması farklı parametreler ile eğitilmiştir. Eğitimler sonucunda elde edilen performans metrikleri raporlanmıştır. Elde edilen performans metrikleri karşılaştırılmıştır ve donanım üzerinde test edilecek derin öğrenme algoritmaları belirlenmiştir. Derin öğrenme algoritmalarına karar verildikten sonra bu algoritmalar için gerekli kütüphaneler Jetson Nano [8] cihazı içerisine kurulmuştur. Kurulumlar tamamlandıktan sonra algoritmalar cihaz üzerinde çalıştırılmıştır ve FPS değerleri gözlemlenmiştir. Buradan elde edilen çıktılar raporlanmıştır. Elde edilen tüm veriler sonucunda hangi algoritmanın proje isteklerine daha uygun olduğuna karar verilmiştir. Projenin geliştirilme aşaması şematik olarak Şekil 1'de görülmektedir.



Şekil 1. Projenin geliştirilme aşamaları (Development stages of the project)

2.1. Veri Seti Hazırlanması (Data Set Preparation)

Veri setinin hazırlanma aşamasında öncelikle nasıl verilere ihtiyaç olduğu belirlenmiştir. Sonrasında bu verilerin elde edilebileceği internet sitelerinde araştırma yapılmıştır. İstenilen nitelikte veriye ulaşılamadığı için Youtube platformu üzerinden sabit kanat İHA'ların uçuş videoları elde edilmiştir. Bu videolardan belirli zaman aralıklarıyla görüntü kaydetmek için bir Python kodu yazılmıştır. Alınan bu görüntülerden veri setine eklenmesi uygun olanlar seçilmiştir. Bu şekilde yaklaşık olarak 200'e yakın videodan veri elde edilmiştir. Elde edilen bu veriler tekrardan gözden geçirilerek veri setine uygun olmayanlar çıkarılmıştır. Örneğin, yerdeki uçak görüntüleri ilk başta dâhil edilse de sonrasında algoritma doğruluğunu olumsuz etkileyeceği düşünüldüğünden veri setinden çıkarılmıştır. Veri seti görüntüleri genel olarak havadaki İHA görüntülerinden oluşmaktadır. Çünkü projede amaç, havadaki sabit kanat İHA'ları tespit etmektir. Bu sebeple veri seti için uygun olan görüntüler havada sabit kanat İHA içeren görüntülerdir. Bu tespite göre İHA kaçış manevraları uygulayabilecektir veya hedefi etkisiz hale getirebilecektir. Veri setindeki verilerin sayısı arttığında algoritma doğruluğunun da artacağı düşünüldüğünden verilere veri artırma yöntemi uygulanmıştır. Keras kütüphanesi [9] kullanılarak verilere döndürme, öteleme ve renk tonlarında değişim gibi işlemler uygulanmıştır. Veri artırma işlemi gerçekleştirildikten sonra uygun veriler seçilerek veri setine eklenmiştir. Bu şekilde veri setinin oluşturulma aşaması tamamlanmıştır. Veri setinde toplamda 5858 adet sabit kanat İHA görseli bulunmaktadır. Veri setinden örnek görseller Şekil 2'de görülmektedir.



Şekil 2. Veri setinden örnek görüntüler (Sample images from the data set)

Veri setindeki görseller tamamlandıktan sonra bu görsellerin etiketlenmesi aşamasına geçilmiştir. LabelImg [10] programı kullanılarak veriler tek tek etiketlenmiştir. Etiket dosyaları ile birlikte veri setinin boyutu 11716 olmuştur. Bu veriler etiket dosyaları ile birlikte %80 eğitim ve %20 test olarak ayrılmıştır. Eğitim verileri algoritmaların eğitimi sırasında kullanılan verilerdir bu nedenle daha fazladır. Test verileri ise algoritma eğitimi esnasında doğrulama doğruluğu ve doğrulama kaybı değerlerinin hesaplanmasında kullanılmıştır. Veri sayısı çok fazla olmadığı için doğrulama verisi ayrılmamıştır. Doğrulama işlemi için de test verileri kullanılmıştır. Algoritma eğitimleri tamamlandıktan sonra test verileri ile algoritma test edilmiştir.

2.2. Veri Seti Eğitimi ve Algoritmaların Karşılaştırılması (Dataset Training and Comparison of Algorithms)

Veri seti hazırlanma aşaması tamamlandıktan sonra kullanılabilecek algoritmalar hakkında araştırma yapılarak gerekli ortam kurulumları yapılmıştır. Faster R-CNN ve SSD algoritmalarının kullanılması planlanmıştı fakat eğitimde kullanılacak Tensorflow kütüphanesi [11] ve birkaç kütüphanede yaşanan sürüm çakışmaları ve hatalar sebebi ile bu algoritmalarda eğitim denenememiştir. Bu algoritmalarda alınan hatalar sebebiyle farklı algoritmalar üzerinde karşılaştırma yapabilmek amacıyla Yolo (you only look once) [12] algoritmasının farklı vesiyonları üzerinde eğitim gerçekleştirilmiştir. Yolo algoritmalarının arka planında evrişimli sinir ağları (convolutional neural network (CNN)) [13] yapısı bulunmaktadır. Yolo bir nesne takip algoritmasıdır. Görüntüyü bölgelere ayırarak işlem yapar. Görüntüyü tek seferde nöral ağlardan geçirir bu sebeple diğer nesne takip algoritmalarına göre daha hızlı çalışmaktadır. Diğer algoritmaların yavaş olmasının sebebi, nesne olması muhtemel alanları belirleyip hepsini ayrı ayrı nöral ağlardan geçirmeleridir. Görüntü üzerindeki işlem sayısı arttıkça hız performansı düşmektedir.

Yolo, nesne tespit aşamasında sınırlayıcı kutu kullanır. Sınırlayıcı kutu içerisinde nesne varsa orta noktası bulunur. Sınırlayıcı kutu içerisinde nesne boyutu, koordinatları ve güven skoru bulunur [14]. Bu değerler vektör şeklinde tutulmaktadır. Her sınırlayıcı kutu için ayrı ayrı tahmin vektörleri oluşturulur. Denklem 1'de oluşturulan vektör yapısı görülmektedir.

$$y^T = [pc, bx, by, bw, bh, c] \quad (1)$$

- pc: Nesne var ise 1 yoksa ise 0'dır.
- bx: Nesnenin orta noktasının x koordinatıdır.
- by: Nesnenin orta noktasının y koordinatıdır.
- bw: Nesnenin genişliğidir.
- bh: Nesnenin yüksekliğidir.
- c: Bağıl skor değeridir. Sınıf sayısı ne kadarsa o kadar sınıf değeri barındırır.(Örneğin, iki sınıflı bir algoritma için araba:1, yaya:0 şeklindedir.)

Sınırlayıcı kutu içerisinde birden fazla nesne olduğunda karışıklık olduğu için Anchor kutu eklenmiştir. Anchor kutu, her nesne için sınırlayıcı kutu ve güven skoru hesaplar. Bunu yaparken Intersection over Union (IoU) [15] kullanır. IoU, genellikle tahmin edilen bölge ve gerçek bölgenin ne kadar örtüştüğünü ölçer. IoU, bu örtüşme miktarını yüzde cinsinden ifade eder. Çok fazla sınırlayıcı kutu olması durumunda Non Max Suppression algoritması ile güven skoru düşük olan kutular silinir. En yüksek güven skoruna sahip nesne çıktı olarak alınır.

Yolo algoritmaları her versiyonunda gelişmeye devam etmiştir. Yolov3'te daha küçük nesne tespiti için residual blocks, skip connection ve up sampling özellikleri eklenmiştir. Önceki Yolo versiyonlarında nesne tespitinin sonucu, öznitelik

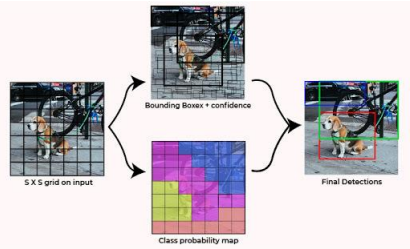
haritası üzerine uygulanmış 1x1 detection kernels sonucuna göre belirleniyor. Yolov3'te ise bu durum farklıdır. Üç farklı yerde üç farklı ölçekteki görüntüye uygulanmaktadır. Yolo [16], Yolov2 [17], Yolov3 [18] ve Yolov4 [19] algoritmalarının mimarisi Darknettir. Yolov5'te [20] ise Pytorch üzerine kurulu bir mimari kullanılmaya başlanmıştır. Yolov5'te mozaik veri artırma, otomatik öğrenme sınırlayıcı kutu bağlantıları bulunmaktadır.

Yolo algoritmasının nesne tespit aşamaları Şekil 3'te görülmektedir. Burada yola çıkarak adım adım açıklandığında:

1. Öncelikle girdi olarak verilen görseli SxS kısımlara bölmektedir. (S değeri genellikle 3, 5, 7, 9 olmaktadır.)
2. Sınırlayıcı kutular ile vektör oluşturulmaktadır.
3. Her bir bölüt kendi içerisinde nesne bulunup bulunmadığını kontrol etmektedir. Eğer varsa nesnenin orta noktasının kendisinde bulunup bulunmadığını kontrol etmektedir. Nesne yüksekliğini, genişliğini bulmakla sorumludur. Orta nokta hangi bölüt içerisinde yer alıyorsa tespitten o bölüt sorumludur.
4. Bir bölütte çok fazla nesne varsa, her bölüt için anchor kutu sayısı kadar vektör değerleri hesaplanmaktadır. Birden çok nesne olduğu durumlarda vektör yapısı Denklem 2'de görülmektedir.

$$y^T = [c1,..,c20,pc1,bx1,by1,bw1,bh1,pc2,bx2,by2,bw2,bh2\ ] \quad (2)$$

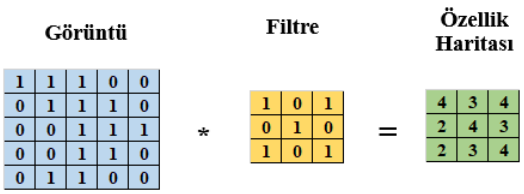
5. Birden fazla bölüt yanılgıya düşerse en yüksek güven skoruna sahip kutu çizilir. Burada Non Max Suppression algoritması kullanılır.



Şekil 3. Yolo algoritmasının çalışma prensibi (Working principle of the yolo algorithm) [21]

Yolov3'te özellik piramit ağı (feature pyramid network (FPN)) kullanılmıştır. Piramitte kendinden önceki görüntünün boyutunu iki katına çıkararak komşu görüntü ile aynı boyuta getirir ve ikisini birbirine ekleyerek görüntü elde eder. Bu işlem bitince piramitin tüm katmanlarındaki nesneleri tahmin eder. Yolov4'te FPN yerine SAM, PAN, SPP kullanılmıştır. SAM, maksimum havuzlama ve ortalama havuzlama işlemleri ayrı ayrı uygulanmaktadır. Sonuçlar sigmoid ile beslenmektedir. Yolov5, Pytorch implementasyonudur. CSP backbone ve PANet neck kullanılmaktadır. Head kısmı Yolov4 ile aynıdır. Orta katmanlarında aktivasyon fonksiyonu olarak leaky relu son katmanında ise sigmoid kullanılmaktadır. Optimizer olarak stokastik gradyan inişi (stochastic gradient descent (SGD)) kullanılmaktadır. Yolov7 [22], Yolov4 yapısını kullanmaktadır. Spatial Pyramid Pooling (SPP) kullanılmaktadır. Farklı boyutlardaki nesnelerin daha iyi tahminini sağlamaktadır. Daha az hesaplama ve parametre içermektedir. Yolov6 [23], Pytorch yapısını kullanmaktadır. Yolov8 ise CSP Darknet mimarisi üzerine kurulmuştur.

Yolo algoritmasının arka planında bulunan CNN yapısı girdi olarak bir görüntü almaktadır. Girdi; evrişim, havuzlama ve tam bağlantılı katmanlardan geçerek eğitilir. Burada evrişim katmanı; görüntüyü ele alan ilk katmandır. Görüntü, belirli değerler taşıyan piksellerden oluşan matrislerdir. Bu katmanda orijinal görsel boyutundan daha küçük bir filtre görsel üzerinde gezer ve bu görsellerden belirli özellikleri yakalamaya çalışır. Bir görselin matris haline getirilmesi ve evrişim işlemi uygulanması Şekil 4'te görülmektedir. 3x3'lük bir filtre uygulanarak bir çıktı matris elde edilir.



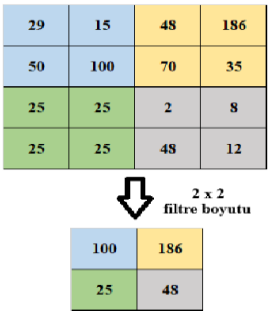
Şekil 4. 3x3'lük filtre ile evrişim işleminin gerçekleştirilmesi (Performing convolution with a 3x3 filter)

Burada filtre uygulanması sonucunda oluşturulan matrise özellik haritası denilmektedir. CNN'de öğrenilen parametreler bu matrislerdeki değerlerdir. Model sürekli bu değerleri günceller ve özellikleri daha iyi öğrenmeye başlar. Bu filtreler; köşe bulma, kenar bulma ve bulanıklaştırma gibi özellik filtreleridir. Kenar bulma, giriş bilgisinin yüksek frekanslı bölgelerini temsil etmektedir. Sobel, prewitt, gabor gibi kenar bulma filtreleri mevcuttur. Elde edilen çıkış, görüntülerin kenar bilgisini verir. Kenarlar genelde CNN katmanının en başında hesaplanır. Filtrelerin girdi üzerinde kaç piksel boyunca kayacağı adım (stride) sayısı ile belirlemektedir. Buna göre özellik haritası boyutu artabilir veya azalabilir.

Kenar bulma, bulanıklaştırma gibi filtreler kullanılırken girdi verisinin boyutu korunmak istenirse padding işlemi uygulanmaktadır. Padding işlemi çıktı boyutunu korur. Padding işlemi girdi verisinin etrafını 0 ile doldurabilir ya da girdi verisini kopyalayarak çoğaltabilir. Burada temel amaç, bilgi kaybını minimuma indirgemektir.

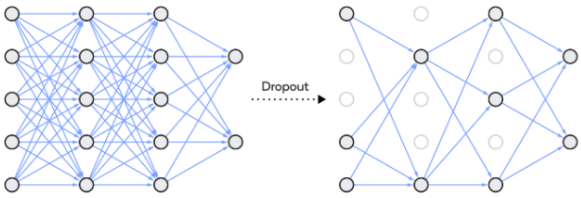
Aktivasyon fonksiyonu olarak ara katmanlarda genel olarak Relu [24] tercih edilmektedir. Relu, negatif ve doğrusal olmayan bir fonksiyondur. Sıfıra eşit ve sıfırdan küçük bir girdi geldiğinde sıfır değeri döndürür. Pozitif bir girdi geldiğinde ise girdinin kendisini döndürür. Burada temel amaç, negatif değerlerden kurtulmaktır. Fakat negatif değerlerin türevinin sıfır olması geriye yayılım (back propagation) esnasında parametrelerin güncellenemeyeceği, yani öğrenme işlemi olmayacağı anlamına gelir. Bu probleme dying relu adı verilmektedir. Bu sebeple Leaky Relu [25] geliştirilmiştir. Leaky Relu ile negatif değerlere tam sıfır değeri atanması yerine sıfıra yakın küçük değerler atanmaktadır. Böylece türevin sıfır olmasının önüne geçilmektedir. Çıktı katmanında genel olarak sigmoid ve softmax tercih edilmektedir. Sigmoid, [0,1] arasında değerler almaktadır. Türevlenebilir, doğrusal olmayan bir fonksiyondur. Genelde ikili sınıflandırma problemlerinde kullanılmaktadır. Softmax, çoklu sigmoid olarak bilinmektedir. İkiden fazla sınıf barındıran problemlerde kullanılmaktadır.

Havuzlama katmanının amacı, girdinin boyutunu azaltmaktır. Hem işlem yoğunluğu azalır hem de gereksiz özellikler yok sayılarak önemli özelliklere odaklanılır. Ağın bu katmanında öğrenilen parametre yoktur. Maksimum havuzlama işleminde filtre uygulanan kısımdaki en büyük değerler seçilir. Ortalama havuzlama katmanında ise filtre uygulanan kısımdaki değerlerin ortalaması alınır. Şekil 5'te maksimum havuzlama işlemi görülmektedir.



Şekil 5. Havuzlama katmanında işlemlerin gerçekleştirilmesi (Performing operations in the pooling layer)

Tam bağlantılı (fully connected) katman genelde CNN mimarisinin sonunda bulunmaktadır. Sınıf skorları gibi hedefleri optimize etmek için kullanılmaktadır. Bu katmanda evrişim işlemleri ve havuzlama işlemleri sonucunda oluşan matris düz vektör haline getirilmektedir. Bu katman, her girişin tüm nöronlara bağlı olduğu bir giriş üzerine çalışır. Seyreltme katmanı (dropout), aşırı öğrenme (overfitting) problemini önlemek için kullanılmaktadır. Birbirine bağlı nöronlar arasında seyreltme işlemi yapar. Rastgele nöronların aktif ve pasif hale getirilmesi ile gerçekleştirilen seyreltme işlemi Şekil 6'de görülmektedir.



Şekil 6. Seyreltme işlemi (Dropout) [26]

Kullanılan Yolo algoritması sürümleri şunlardır: V3, V4, V5, V6, V7 ve V8. Bu algoritmalardan bazı sürümlerde x, m, s ve l gibi farklı modeller bulunmaktadır. Yolo algoritmalarının eğitimi uzun sürdüğünden dolayı eğitim işlemi için Google Colaboratory [27] ortamı tercih edilmiştir. Google Colaboratory ortamı kullanıcılara ücretsiz Graphics Processing Unit (GPU) desteği sağlamaktadır. Bu algoritmalardan ilk olarak Yolov3, Yolov4, Yolov3-tiny ve Yolov4-tiny [28] algoritmalarının eğitimi gerçekleştirilmiştir. Farklı yığın boyutu (batch size) değerleri, subdivision değerleri ve girdi boyutları ile eğitimler gerçekleştirilmiştir. Yığın boyutu, her döngüde (epoch) kaç adet görüntü ile eğitim gerçekleştirileceğini belirtmektedir. Subdivision ise yığınları mini-yığın gruplarına ayırmaktadır. Böylece, büyük görüntüler ile gerçekleştirilen eğitimler sırasında belleğin daha verimli bir şekilde kullanmasını sağlamaktadır. Gerçekleştirilen eğitimlerden elde edilen veriler ve algoritmaların karşılaştırılması Tablo 1'de görülmektedir.

Tablo 1. Yolov3 ve Yolov4 algoritmalarının karşılaştırılması (Comparison of Yolov3 and Yolov4 algorithms)

Algoritma	Yığın Boyutu	Subdivision	Genişlik Yükseklik	F1 Skor (%)	mAP (%)
yolov3	64	8	256x256	0.90	0.9131
yolov3	64	16	320x320	0.89	0.9360
yolov3-tiny	64	16	416x416	0.76	0.8399
yolov3-tiny	64	4	416x416	0.78	0.8472
yolov3-tiny	64	2	416x416	0.79	0.8520
yolov4-tiny	64	4	416x416	0.90	0.9483
yolov4-tiny	64	1	416x416	0.88	0.9370

Tablo 1'de görülmekte olan mAP performans metriğinin formülü Denklem 3'te verilmiştir. F1-skor performans metriğinin formülü ise Denklem 4'te verilmiştir. F1-skor değerinin kullanılmasının en temel sebebi eşit dağılmayan veri kümelerinde hatalı bir model seçimi yapmamaktır. Ayrıca sadece yanlış negatif ya da yanlış pozitif değil tüm hata maliyetlerini de içerecek bir ölçme metriğine ihtiyaç duyulduğu için F1-skor çok önemlidir. F1-skor değeri doğru pozitif, yanlış pozitif ve yanlış negatif değerleri ile hesaplanmaktadır.

$AP_{k=the\ AP\ of\ class\ k}$

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \quad (3)$$

$n = the\ number\ of\ classes$

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad (4)$$

Burada verilen AP (Average precision) değeri ortalama precision değerini belirtmektedir. "k" parametresi ise her bir sınıf için hesaplanan ortalama precision değerini temsil etmek için kullanılmaktadır. "n" parametresi ise veri setindeki toplam sınıf sayısını belirtmektedir. Denklem 4'te bahsedilen precision (P) performans metriği ve Recall (R) performans metriğinin hesaplanması Denklem 5'te verilmiştir. Precision, pozitif olarak tahmin edilen değerlerin gerçekten kaç tanesinin pozitif olduğunu göstermektedir. Recall ise pozitif olarak tahmin edilmesi gereken işlemlerin ne kadarının pozitif olarak tahmin edildiğini gösteren bir metriktir. Bu performans metrikleri yolov5, yolov6, yolov7 ve yolov8 algoritmalarının

değerlendirilmesinde de kullanılmaktadır.

$$Recall\ (R) = \frac{TP}{TP + FN}\ ,\quad Precision\ (P)\ = \frac{TP}{TP + FP} \quad (5)$$

Denklem 5'te verilen terimlerin açıklamaları şu şekildedir:

- Doğru Pozitif (TP): Gerçek değerin 1 ve tahmin edilen değerin de 1 olduğu örneklerdir.
- Doğru Negatif (TN): Gerçek değerin 0 ve tahmin edilen değerin de 0 olduğu örneklerdir.
- Yanlış Pozitif (FP): Gerçek değerin 0 ancak tahmin edilen değerin 1 olduğu örneklerdir.
- Yanlış Negatif (FN): Gerçek değerin 1 ancak tahmin edilen değerin 0 olduğu örneklerdir.

Eğitimler sonucunda elde edilen veriler incelendiğinde hem FPS değerinin daha verimli olması hem de değerlendirilen performans metriklerinin daha yüksek olması sebebi ile yolov4-tiny algoritmasının kullanılmasına karar verilmiştir. FPS karşılaştırması için yolov3 modeli de Jetson Nano cihazı üzerinde çalıştırılarak çıktılar elde edilmiştir. Elde edilen çıktılar "Algoritmaların Seçilmesi ve Donanım Üzerinde Çalışması" başlığı altında sunulmuştur.

Yolov5 algoritmasının farklı versiyonları bulunmaktadır. Bunlar; 5l, 5m, 5x ve 5s modelleridir. Bu algoritmalarda farklı yığın boyutu değerleri ile eğitimler gerçekleştirilmiştir. Gerçekleştirilen eğitimlerden elde edilen performans metrikleri ve algoritmaların karşılaştırılması Tablo 2'de görülmektedir.

Tablo 2. Yolov5 algoritmalarının eğitim sonuçları ve karşılaştırılması (Training results and comparison of Yolov5 algorithms)

Algoritma	Yığın Boyutu	Döngü	P (%)	R (%)	mAP (%)	Parametre (M)	BFLOPs
yolov5-s	4	100	0.97	0.957	0.976	7.2	16.5
yolov5-x	8	100	0.968	0.946	0.977	86.7	205.7
yolov5-m	8	100	0.971	0.941	0.975	21.2	49.0
yolov5-l	8	100	0.979	0.966	0.983	46.5	109.1
yolov5-s	8	100	0.961	0.952	0.978	7.2	16.5

Tablo 2' de görüldüğü üzere en başarılı Yolov5 algoritması 5-l modeli olmuştur. Yolov5-l algoritması, yolov5-x algoritmasına göre daha düşük BFLOPs (Billion Floating Point Operations Per Second) ve parametre değerlerine sahiptir. Bu durum, yolov5-l modelinin daha az karmaşık bir yapıya sahip olduğunu ve daha düşük işlem gücüne ihtiyaç duyduğunu göstermektedir. Ancak, modelin performansını değerlendirmek için yalnızca BFLOPs ve parametre sayısına bakmak yeterli değildir. Modelin içyapısı, kullanılan katmanlar, aktivasyon fonksiyonları ve diğer faktörler de önemlidir. Algoritmaların değerleri birbirine yakın olması sebebi ile en çok FPS değeri elde edilen algoritmanın kullanılabileceği düşünülmüştür.

Yolov6 algoritmasının da farklı versiyonları bulunmaktadır. Bunlar; 6n, 6l, 6m ve 6s modelleridir. Bu algoritmalarda aynı yığın boyutu ve döngü değerleri ile eğitimler gerçekleştirilerek farklı versiyonların performansları değerlendirilmiştir. Yapılan karşılaştırmalar Tablo 3'te görülmektedir.

Tablo 3. Yolov6 algoritmalarının eğitim sonuçları ve karşılaştırılması (Training results and comparison of Yolov6 algorithms)

Algoritma	Yığın Boyutu	Döngü	AP (%)	AR (%)	mAP (%)	Parametre (M)	GFLOPs
yolov6-l	8	20	0.955	0.744	0.9553	63.6	95.5
yolov6-n	8	20	0.975	0.719	0.974	4.9	7.0
yolov6-m	8	20	0.979	0.718	0.9787	37.1	54.3
yolov6-s	8	20	0.978	0.724	0.9781	19.6	27.7

Tablo 3'e bakıldığında en başarılı yolov6 algoritmasının s versiyonu olduğu görülmüştür. 6-m ve 6-s algoritmalarının performans metrikleri birbirlerine çok yakındır ancak 6-s modelinde parametre ve GFLOPs (Giga Floating Point Operations Per Second) değerleri incelendiğinde daha düşük işlem gücüne ihtiyaç duyduğu görülmektedir.

Yolov7 algoritmasının çalışma süresi çok uzun sürdüğünden dolayı Google Colab süresi yetersiz kalmıştır buna ek olarak yapılan eğitimde elde edilen sonuç çok başarılı olmadığı için sadece bir eğitim gerçekleştirilmiştir. Yolov7 algoritmasının eğitiminden elde edilen sonuç Tablo 4'te görülmektedir.

Tablo 4. Yolov7 algoritmasının eğitim sonucu (Training result of Yolov7 algorithm)

Algoritma	Yığın Boyutu	Döngü	P (%)	R (%)	mAP (%)
yolov7-x	4	20	0.817	0.821	0.87

Yolov8 algoritmasının da farklı versiyonları bulunmaktadır. Bunlar; 8s, 8m, 8n ve 8x modelleridir. Bu algoritmalarda aynı batch size ve döngü değerleri ile eğitimler gerçekleştirilerek farklı versiyonların performansları değerlendirilmiştir. Yapılan karşılaştırmalar Tablo 5'te görülmektedir.

Tablo 5. Yolov8 algoritmalarının eğitim sonuçları ve karşılaştırılması (Training results and comparison of Yolov8 algorithms)

Algoritma	Yığın Boyutu	Döngü	P (%)	R (%)	mAP (%)	Parametre (M)	BFLOPs
yolov8-s	8	20	0.936	0.931	0.969	11.2	28.6
yolov8-x	8	20	0.9725	0.9157 9	0.975	68.2	257.8
yolov8-m	8	20	0.968	0.944	0.98	25.9	78.9
yolov8-n	8	20	0.954	0.901	0.961	3.2	8.7

Tablo 5'e bakıldığında yolov8 algoritmalarından en başarılı versiyonun m olduğu görülmektedir. Yolov8-x modeli, yüksek mAP değeriyle dikkat çekmektedir ancak, parametre ve BFLOPs değerleri incelendiğinde, bu modelin daha karmaşık olduğu ve daha fazla işlem gücü gerektirdiği görülmektedir. Tabloda görülen performans metriklerinde iyi sonuçlar vermesi sebebi ile Jetson Nano cihazında çalıştırılmasına karar verilmiştir.

Eğitimlerden elde edilen sonuçlar değerlendirilerek yolov3, yolov4, yolov5, yolov7 ve yolov8

algoritmaları Jetson Nano cihazı üzerinde çalıştırılarak elde edilen çıktılar sunulmuştur.

2.3. Algoritmaların Seçilmesi ve Donanım Üzerinde Çalışması (Selection of Algorithms and Running on Hardware)

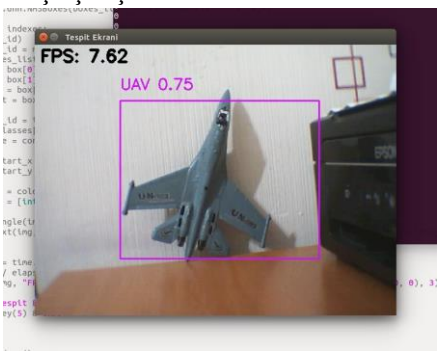
Algoritmalar eğitim sırası ile cihaz üzerinde çalıştırılmıştır. Öncelikle Nvidia Jetson Nano Developer Kit üzerinde çalışma ortamı kurulmuştur. Cihaz üzerine takılan sd karta Nvidia'nın sitesinden erişilen img uzantılı dosyanın kurulumu balenaEtcher [29] programı ile gerçekleştirilmiştir. Jetson Nano'nun çalışması için gerekli kurulumlar tamamlandıktan sonra kütüphane kurulumlarına geçilmiştir. Açık kaynak kodlu görüntü işleme kütüphanesi olan Opencv cuda desteği ile derlenmiştir [30]. Bu şekilde cuda çekirdeklerinin aktif kullanılması sağlanmıştır. Gerekli diğer kütüphaneler de kurulduktan sonra yolov3 ve yolov4 algoritmaları cihaz üzerinde çalıştırılmıştır. İlk olarak yolov3 algoritması çalıştırılmıştır. Elde edilen çıktı Şekil 7'de görülmektedir.

Şekil 7. Yolov3 algoritmasından elde edilen çıktı (Output from the yolov3 algorithm)

Görüldüğü üzere yaklaşık olarak 1,30 FPS alınmıştır. Bu FPS değeri yetersiz olduğu için yolov3-tiny modeli test edilmiştir. Yolov3-tiny modelinden elde edilen çıktı Şekil 8'de görülmektedir.

Şekil 8. Yolov3-tiny algoritmasından elde edilen çıktı (Output from the yolov3-tiny algorithm)

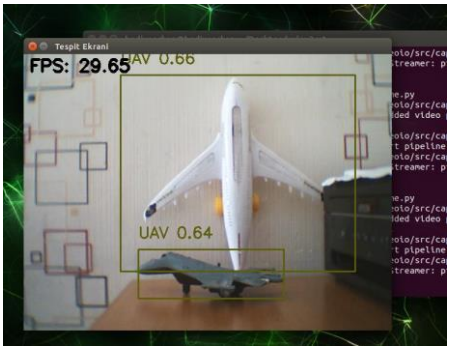
Görüldüğü üzere tiny modelinde katman sayısı daha az olduğu için algoritma daha hızlıdır. Bu sebeple FPS değeri daha yüksektir. Fakat doğruluk olarak baktığımızda örnekte %60 civarında eşleşmiştir. Bu nedenle yolov4-tiny algoritması değerlendirilmiştir. Şekil 9'da yolov4-tiny algoritmasının donanım üzerindeki çıktısı ile ilgili bir örnek verilmiştir.



Şekil 9. Yolov4-tiny algoritmasından elde edilen çıktı (Output from the yolov4-tiny algorithm)

Görüldüğü üzere yaklaşık olarak 8 FPS değeri elde edilmiştir. Fakat bu değer gerçek zamanlı görüntü işleme uygulamaları için yeterli değildir bu nedenle artırılması gerekmektedir. FPS değerini arttırmak için kullanılan weights uzantılı model dosyasının tensorrt'ye çevrilmesine karar verilmiştir. Tensorrt [31], Nvidia tarafından geliştirilen ve derin öğrenme modellerini optimize ederek hızlandırmak için kullanılan bir yazılım platformudur. Bellek kullanımını optimize etmektedir ve algoritmaları

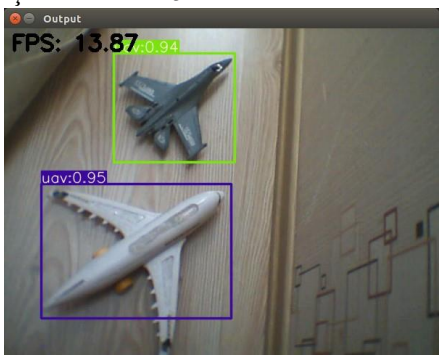
hızlandırmaktadır. GPU üzerinde optimize edilmiş bir şekilde çalışmaktadır. Bu sayede GPU kaynaklarını daha verimli kullanmaktadır. Bu sebeple yolov4-tiny algoritması tensorrt'ye dönüştürülmüştür. Dönüşüm sonucunda elde edilen trt uzantılı model dosyası ile hedef tespiti gerçekleştirilmiştir. Donanım üzerinde trt uzantılı model dosyasının çalıştırılması ile elde edilen çıktı Şekil 10'da görülmektedir.



Şekil 10. Yolov4-tiny-tensorrt algoritmasından elde edilen çıktı (Output from the yolov4-tiny algorithm)

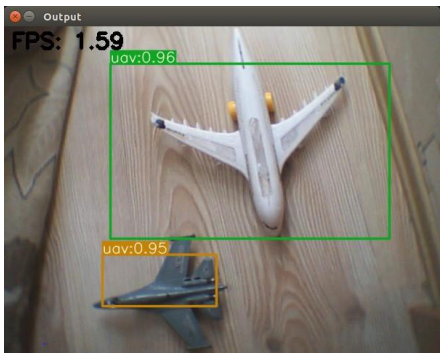
Şekil 10'da görüldüğü üzere tensorrt ile birlikte yolov4-tiny algoritmasının FPS değeri yaklaşık olarak 30'dur. 30 FPS değeri ile yolov4-tiny-tensorrt algoritmasının kullanılabileceği düşünülmüştür.

Algoritmaların karşılaştırılması için en yüksek doğruluk değeri elde edilen yolov5-l algoritması donanım üzerinde çalıştırılmıştır. Yolov5-l algoritmasının donanım üzerinde çalıştırılması için tensorrt dönüşümü gerekmektedir. Yolov5 algoritması Pytorch tabanlı bir algoritma olduğu için yolov3 ve yolov4'ten farklı olarak dönüşüm sonucunda engine uzantılı bir model dosyası oluşmaktadır. Yolov5-l algoritmasından yaklaşık olarak 15 FPS değeri elde edilmiştir. Elde edilen çıktı Şekil 11'de görülmektedir.



Şekil 11. Yolov5-l algoritmasından elde edilen çıktı (Output from the yolov5-l algorithm)

Yolov7-x algoritmasının elde edilen doğruluk değeri düşük olmasına rağmen FPS değerinin test edilmesi için donanım üzerinde çalıştırılmıştır. Yolov7-x algoritmasının donanım üzerinde çalıştırılması için tensorrt dönüşümü gerekmektedir. Dönüşüm sonucunda engine uzantılı bir model dosyası oluşmaktadır. Yolov7-x algoritmasının çalıştırılması ile yaklaşık olarak 2 FPS değeri elde edilmiştir. Yolov7-x algoritmasından elde edilen çıktı Şekil 12'de görülmektedir.

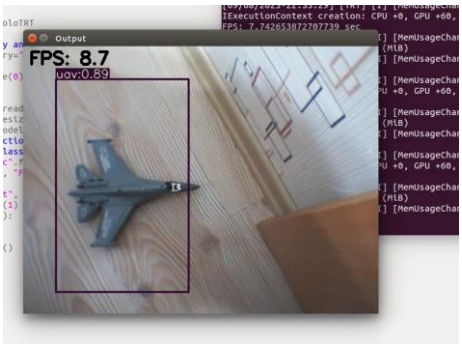


Şekil 12. Yolov7-x algoritmasından elde edilen çıktı (Output from the yolov7-x algorithm)

Donanım üzerinde son olarak yolov8 algoritması test edilmiştir. Yolov8 algoritmasının donanım üzerinde çalıştırılması için tensorrt dönüşümü gerekmektedir. Dönüşüm sonucunda engine uzantılı bir model dosyası oluşmaktadır. Yolov8-m algoritmasında en yüksek doğruluk değeri elde edildiği için donanım üzerinde test edilmiştir. Yolov8-m algoritması ile yaklaşık olarak 4 FPS elde edilmiştir. Elde edilen çıktı örneği Şekil 13'te görülmektedir.

Şekil 13. Yolov8-m algoritmasından elde edilen çıktı (Output from the Yolov8-m algorithm)

Yolov8-s algoritmasının CNN katmanları daha az olduğu için daha hızlı olduğu düşünülmüştür. Bu nedenle FPS karşılaştırılması için yolov8-s algoritması donanım üzerinde test edilmiştir. Yolov8-s algoritması ile donanım üzerinde yaklaşık olarak 9 FPS değeri elde edilmiştir. Elde edilen çıktı örneği Şekil 14'te görülmektedir. Gözlemlenen sonuçlar doğrultusunda yolov8-m algoritması yerine yolov8-s algoritmasının da tercih edilebileceği düşünülmüştür.



Şekil 14. Yolov8-s algoritmasından elde edilen çıktı (Output from the yolov8-s algorithm)

Algoritmalar Jetson Nano cihazı üzerinde başarıyla çalıştırıldıktan sonra, kullanıcı dostu bir ara yüz tasarlamak amacıyla PyQt5 [32] kütüphanesi kullanılmıştır. PyQt5, Qt adlı C++ kütüphanesinin Python için bir bağlamasıdır ve Qt kütüphanesi üzerinde uygulama geliştirmeyi sağlar. Qt, geniş bir kullanım alanına sahip olan bir araç setidir ve masaüstü uygulamalardan mobil uygulamalara, oyunlardan grafiksel araçlara kadar birçok yazılım türü için kullanılır.

Projenin kullanım kolaylığı için tasarlanan ara yüz örneği, Şekil 15'te sunulmaktadır. Bu ara yüz,

projenin başarıyla tamamlanan kısmını kullanıcıya net bir şekilde iletmek için tasarlanmıştır.

Şekil 15. Tasarlanan ara yüz (Designed interface)

3. BULGULAR (RESULTS)

Bu kısımda literatür taraması verilerek yapılan çalışmanın, daha önce yapılan çalışmalarla ilişkisi/farklılıkları verilmiştir.

Mingjie ve ark. [33] çeşitli derin öğrenme algoritmaları kullanarak hedef tespiti yapmışlardır. Kullandıkları algoritmaların karşılaştırmalarını sunmuşlardır. Bir başka çalışmada Yuanyuan Hu ve ark. [34] nesne tespiti için veri seti oluşturup bir derin öğrenme algoritması önermişlerdir.

Şahin ve ark. Yolov3, Yolov5 ve YoloDrone+ [35] algoritmalarını iki farklı hazır veri seti ile test etmiştir ve karşılaştırmıştır. Mevcut çalışmada ise FPS ve mAP değerleri belirtilerek daha fazla algoritma ile karşılaştırma yapılmıştır ve özgün bir veri seti kullanılmıştır.

Çelik ve ark. [36] üç farklı sınıf için Yolov3 algoritması ile %84,81 doğruluk elde etmiştir. Çalışmada özgün bir veri seti yerine hazır bir veri seti tercih edilmiştir. Mevcut çalışmada özgün veri seti kullanılarak Yolov3 algoritması ile %93 doğruluk, Yolov3-tiny algoritması ile %85 doğruluk elde edilmiştir.

Albayrak çalışmasında [37] araç tespiti için Yolov3 algoritması ve Mask R-CNN algoritması kullanmıştır. Algoritmalar karşılaştırıldığında sonuçların birbirine yakın olduğu ama Mask R-CNN algoritmasının doğruluk açısından küçük bir farkla daha iyi olduğu görülmüştür. Yolov3 algoritmasının işlem yükünün daha az olması sebebiyle Mask R-CNN'den daha hızlı olduğu belirtilmiştir. Li ve ark. [38] VisDrone veri seti ile 10 farklı sınıf için Yolov8-s algoritması kullanılarak bir yapay zekâ modeli eğitmiştir. Yolov5-m, MobilNetv2-SSD, Yolov4-s, Yolov5-s, YoloX-s, Yolov7-tiny algoritmaları ile karşılaştırılmıştır. Yolov8-s ve yolov5-s algoritmalarının doğruluk değeri ve FPS değeri bakımından avantajlı olduğu belirtilmiştir. Mevcut çalışmada bu bulgulara ek olarak Yolov3, Yolov6, Yolov7 ve bunların s, m, l, x versiyonları karşılaştırılmıştır.

Liu ve ark. [39] Yolov5-s algoritmasının backbone yapısını Efficientlite ile değiştirmişlerdir. Yolov5-s algoritmasının daha gelişmiş bir versiyonu ile %94,82 doğruluk elde etmişlerdir. Mevcut çalışmada ise Yolov5-s algoritması ile %97,80 doğruluk elde edilmiştir. Shi ve ark. [40] Yolov4 modeli kullanarak düşük irtifa drone tespiti için bir tanıma yöntemi önermiştir. Karşılaştırma için Yolov3 ve SSD algoritmaları tercih edilmiştir. Çalışmada Yolov4 algoritması ile elde edilen doğruluk değeri %89,32 'dir. Mevcut çalışmada ise Yolov4-tiny algoritması ile elde edilen doğruluk değeri ise %94,83'tür.

Verilen literatür özeti, derin öğrenme tabanlı nesne tespiti alanında bir dizi çalışmayı kapsamaktadır. Çeşitli araştırmalar, farklı derin öğrenme algoritmalarının performanslarını değerlendirmiş ve farklı veri setleri üzerinde bu algoritmaların karşılaştırmalarını sunmuştur. Özellikle, YOLO serisi algoritmaları (YOLOv3, YOLOv5, YOLOv8, YOLOv4) ve bunların farklı versiyonları, yaygın olarak karşılaştırılan ve incelenen algoritmalar arasında yer almaktadır. YOLOv5, YOLOv8 gibi daha yeni ve geliştirilmiş versiyonlarının, yüksek doğruluk ve işlem hızı açısından avantajlı olduğu vurgulanmıştır. Özellikle, YOLOv5-S ve YOLOv8-S algoritmalarının, farklı çalışmalarda daha yüksek doğruluk değerleri ve hızlı işlem performansı

sunduğu belirtilmiştir. Sonuç olarak, verilen çalışmaların üzerinden yola çıkarak, mevcut çalışmada özgün veri seti kullanarak geniş kapsamlı bir derin öğrenme tabanlı nesne tespiti algoritmaları karşılaştırması sunması ve bu algoritmaların farklı versiyonlarını içermesi, literatüre önemli bir katkı yapabileceğini göstermektedir. Bu çalışma, araştırmacıların belirli bir algoritmayı seçme konusunda daha bilinçli kararlar vermelerine ve gerçek dünya uygulamalarına daha uygun algoritmaları belirlemelerine yardımcı olabilir.

4. SONUÇLAR (CONCLUSIONS)

Yapılan testler ve eğitimler sonucunda yolov3 algoritmasından elde edilen en yüksek doğruluk değeri %93,6'dır. Yolov3-tiny algoritmasından elde edilen en yüksek doğruluk değeri %84,72'dir. Yolov4-tiny algoritmasından elde edilen en yüksek doğruluk değeri %94,83'tür. Tiny modellerinin tercih edilme sebebi FPS değerlerinin daha yüksek olmasıdır. Yolov5 algoritmaları içerisinde en yüksek doğruluk değerinin elde edildiği yolov5 algoritması 5-l'dir ve %98,3 doğruluk elde edilmiştir. Yolov6 algoritmaları içerisinde elde edilen en yüksek doğruluk değeri 6-s algoritmasına aittir ve %97,8'dir. Yolov7-x algoritmasında elde edilen doğruluk değeri %87'dir. Çok yüksek bir değer elde edilemediği için diğer versiyonları ile eğitim gerçekleştirilmemiştir. Yolov8 algoritmaları içerisinde elde edilen en yüksek doğruluk değeri 8-m algoritmasına aittir ve %98'dir.

Eğitim sürecinin sonunda elde edilen verilerin değerlendirilmesi sonucunda, FPS ve doğruluk değerleri üzerinden yapılan analizler neticesinde, yolov4-tiny-tensorrt algoritmasının kullanılmasının daha verimli bir tercih olduğuna karar verilmiştir. Yolov4-tiny algoritmasından elde edilen model dosyası tensorrt'ye çevrilerek FPS değeri artırılmıştır. Bu algoritma ile yaklaşık olarak 30 FPS değeri elde edilmiştir, bu da sistemdeki nesne algılama ve sınıflandırma işlemlerinin hızlı ve güvenilir bir şekilde gerçekleştirilebileceğini göstermektedir.

Kullanılan algoritmanın eğitim parametreleri ve performans metrikleri şu şekildedir:

- Yığın Boyutu: 64
- Subdivision: 4
- Genişlik x Yükseklik: 416 x 416
- F1 Skor: %90
- mAP: %94,83

Çalışma sonucunda elde edilen sonuç ve bulgulardan yola çıkılarak gelecek çalışmalar için araştırmacılara şunlar önerilir; Yolov4-tiny-tensorrt algoritmasının yüksek FPS değeri sağlaması, tiny modellerin önemini vurgular. Gelecekteki çalışmalar, bu tiny modellerin geliştirilmesi üzerine odaklanabilir. Bu çalışmada kullanılan algoritmalar arasında yapılan karşılaştırmalar genişletilebilir. Diğer yaygın olarak kullanılan nesne algılama algoritmalarıyla karşılaştırmalar yapmak, alandaki en iyi performansı elde etmek için katkı sağlayabilir.

TEŞEKKÜR (ACKNOWLEDGMENTS)

Bu araştırma, TÜBİTAK 2209-A Üniversite Öğrencileri Araştırma Projeleri Destekleme Programı kapsamında desteklenmiştir. TÜBİTAK'ın sağladığı maddi kaynaklar ve destek, bu araştırmanın gerçekleşmesine katkı sağlamıştır ve bilimsel araştırma deneyimi elde etmemize yardımcı olmuştur. Proje numarası 1919B012203674 olan bu araştırmanın gerçekleştirilmesinde maddi ve manevi desteklerinden dolayı TÜBİTAK'a teşekkür ederiz.

ETİK STANDARTLARIN BEYANI (DECLARATION OF ETHICAL STANDARDS)

Bu makalenin yazarı çalışmalarında kullandıkları materyal ve yöntemlerin etik kurul izni ve/veya yasal-özel bir izin gerektirmediğini beyan ederler.

The author of this article declares that the materials and methods they use in their work do not require ethical committee approval and/or legal-specific permission.

YAZARLARIN KATKILARI (AUTHORS' CONTRIBUTIONS)

***Hediye ORHAN*:** Deneyleri yapmış, sonuçlarını analiz etmiş ve makalenin yazım işlemini gerçekleştirmiştir.

***Ferda Nur ARICI*:** Deneyleri yapmış, sonuçlarını analiz etmiş ve makalenin yazım işlemini gerçekleştirmiştir.

She conducted the experiments, analyzed the results and performed the writing process.

ÇIKAR ÇATIŞMASI (CONFLICT OF INTEREST)

Bu çalışmada herhangi bir çıkar çatışması yoktur.

There is no conflict of interest in this study.

KAYNAKLAR (REFERENCES)

[1] Akyürek, S., M.A. Yılmaz, and M. Taşkıran, İnsansız Hava Araçları: Muhabere Alanında ve Terörle Mücadelede Devrimsel Dönüşüm. Bilge Adamlar Stratejik Araştırma Merkezi, Ankara, 2012.

[2] ANKA İHA. Available from: https://www.tusas.com/urunler/iha/operatif-stratejik-iha-sistemleri/anka.

[3] Vestel Karayel İHA. Available from: https://www.vestel.com.tr/content/karayel.

[4] Ekmekcioglu, A. and M. Yıldız, İnsansız Hava Araçlarının Askeri ve Sivil Alanlarda Kullanımı: ABD ve Türkiye Örnekleri ve Bazı Politika Önerileri. Türk İdare Dergisi: p. 169.

[5] Bayraktar TB2. Available from: https://www.baykartech.com/tr/uav/bayraktar-tb2/.

[6] Bayraktar TB3. Available from: https://baykartech.com/tr/bayraktar-tb3/.

[7] Kayaalp, K. and A.A. Süzen, Derin Öğrenme. Derin Öğrenme ve Türkiye'deki Uygulamaları, Adıyaman, Türkiye: İKSAD Yayınevi, 2018: p. 25-28.

[8] Nvidia Jetson Nano Developer Kit. Available from: https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[9] Keras Library. Available from: https://keras.io/.

[10] LabelImg. Available from: https://github.com/tzutalin/labelImg.

[11] Tensorflow Library. Available from: https://www.tensorflow.org/?hl=tr.

[12] Du, J. Understanding of object detection based on CNN family and YOLO. in Journal of Physics: Conference Series. 2018. IOP Publishing.

[13] Orhan, H. and E. YAVŞAN, Artificial intelligence-assisted detection model for melanoma diagnosis using deep learning techniques. Mathematical Modelling and Numerical Simulation with Applications, 2023. 3(2): p. 159-169.

[14] Aktaş, A., Ö. DEMİR, and B. DOĞAN, Derin öğrenme yöntemleri ile dokunsal parke yüzeyi tespiti. Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 2020. 35(3): p. 1685-1700.

[15] Zhou, D., et al. Iou loss for 2d/3d object detection. in 2019 international conference on 3D vision (3DV). 2019. IEEE.

[16] Deng, J., et al. A review of research on object detection based on deep learning. in Journal of Physics: Conference Series. 2020. IOP Publishing.

[17] Sang, J., et al., An improved YOLOv2 for vehicle detection. Sensors, 2018. 18(12): p. 4272.

[18] Aswini, N. and S. Uma. Custom Based Obstacle Detection Using Yolo v3 for Low Flying Drones. in 2021 International Conference on Circuits, Controls and Communications (CCUBE). 2021. IEEE.

[19] Bochkovskiy, A., C.-Y. Wang, and H.-Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[20] Li, S., et al., Yolo-firi: Improved yolov5 for infrared image object detection. IEEE access, 2021. 9: p. 141861-141875.

[21] Şekil 3. Available from: https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/.

[22] Wang, C.-Y., A. Bochkovskiy, and H.-Y.M. Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023.

[23] Li, C., et al., YOLOv6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976, 2022.

[24] Schmidt-Hieber, J., Nonparametric regression using deep neural networks with ReLU activation function. 2020.

[25] Dubey, A.K. and V. Jain. Comparative study of convolution neural network's relu and leaky-relu activation functions. in Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018. 2019. Springer.

[26] Şekil 6. Available from: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/developing-apps-with-neural-processing-sdk/tuning-optimizing-machine-learning.

[27] Bisong, E. and E. Bisong, Google colaboratory. Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners, 2019: p. 59-64.

[28] Jiang, Z., et al., Real-time object detection method based on improved YOLOv4-tiny. arXiv preprint arXiv:2011.04244, 2020.

[29] BalenaEtcher. Available from: https://github.com/balena-io/etcher.

[30] Opencv. Available from: https://qengineering.eu/install-opencv-on-jetson-nano.html.

[31] TensorRT. Available from: https://developer.nvidia.com/tensorrt.
[32] PYQT5. Available from: https://pypi.org/project/PyQt5/.
[33] Liu, M., et al., Uav-yolo: Small object detection on unmanned aerial vehicle perspective. Sensors, 2020. 20(8): p. 2238.
[34] Hu, Y., et al. Object detection of UAV for anti-UAV based on improved YOLO v3. in 2019 Chinese Control Conference (CCC). 2019. IEEE.
[35] Sahin, O. and S. Ozer. YOLODrone+: improved YOLO architecture for object detection in UAV images. in 2022 30th Signal Processing and Communications Applications Conference (SIU). 2022. IEEE.
[36] ALTINÖRS, A. and S. ÇELİK, YOLOv3 Derin Öğrenme Algoritması ile İHA Görüntülerinden Çevresel Atık Tespiti. International Journal of Innovative Engineering Applications. 7(1): p. 76-85.
[37] Albayrak, E., Derin öğrenme ile İHA görüntülerinden nesne tespitinin yapılması. 2021, Bilecik Şeyh Edebali Üniversitesi, Fen Bilimleri Enstitüsü.
[38] Li, Y., et al., A Modified YOLOv8 Detection Network for UAV Aerial Image Recognition. Drones, 2023. 7(5): p. 304.
[39] Liu, B. and H. Luo, An improved Yolov5 for multi-rotor UAV detection. Electronics, 2022. 11(15): p. 2330.
[40] Shi, Q. and J. Li. Objects detection of UAV for anti-UAV based on YOLOv4. in 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT. 2020. IEEE.