BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
SAVUNMA TEKNOLOJİLERİ VE SİSTEMLERİ ANABİLİM DALI
SAVUNMA TEKNOLOJİLERİ VE SİSTEMLERİ DOKTORA PROGRAMI

KARA MUHAREBE ARAÇLARI İÇİN YAPAY ZEKÂ TABANLI HEDEF TESPİT SİSTEMİ

HAZIRLAYAN

REŞAT ALİ TÜTÜNCÜOĞLU

ANKARA – 2025

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
SAVUNMA TEKNOLOJİLERİ VE SİSTEMLERİ ANABİLİM DALI
SAVUNMA TEKNOLOJİLERİ VE SİSTEMLERİ DOKTORA PROGRAMI

KARA MUHAREBE ARAÇLARI İÇİN YAPAY ZEKÂ TABANLI HEDEF TESPİT SİSTEMİ

HAZIRLAYAN

REŞAT ALİ TÜTÜNCÜOĞLU

DOKTORA TEZİ

TEZ DANIŞMANI

DOÇ. DR. SELDA GÜNEY

ANKARA – 2025

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Savunma Teknolojileri ve Sistemleri Anabilim Dalı Savunma Teknolojileri ve Sistemleri Doktora Programı çerçevesinde Reşat Ali TÜTÜNCÜOĞLU tarafından hazırlanan bu çalışma, aşağıdaki jüri tarafından Doktora Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 08 / 05 / 2025

Tez Adı: Kara Muharebe Araçları İçin Yapay Zekâ Tabanlı Hedef Tespit Sistemi

Tez Jüri Üyeleri		**İmza**
Doç.Dr. Selda GÜNEY	Başkent Üniversitesi	
Doç.Dr. Murat ÜÇÜNCÜ	Başkent Üniversitesi	
Prof.Dr. Hamit ERDEM	Başkent Üniversitesi	
Doç.Dr. Durdu Hakan UTKU	THK Üniversitesi	
Prof.Dr. Hasan Şakir BİLGE	Gazi Üniversitesi	

ONAY

Prof. Dr. Dilek ÇÖKELİLER SERDAROĞLU
Fen Bilimleri Enstitüsü Müdürü
Tarih : … / 05 / 2025

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
DOKTORA TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 23 / 05 / 2025

Öğrencinin Adı, Soyadı	: Reşat Ali TÜTÜNCÜOĞLU
Öğrencinin Numarası	: 2109445
Anabilim Dalı	: Savunma Teknolojileri ve Sistemleri
Programı	: Savunma Teknolojileri ve Sistemleri Doktora
Danışmanın Unvanı/Adı, Soyadı	: Doç.Dr. Selda GÜNEY
Tez Başlığı	: Kara Muharebe Araçları İçin Yapay Zekâ Tabanlı Hedef Tespit Sistemi

Yukarıda başlığı belirtilen Doktora tez çalışmamın; Giriş, Ana Bölümler, Sonuç ve Ekler Bölümünden oluşan, toplam 125 sayfalık kısmına ilişkin, 23 / 05 / 2025 tarihinde tez danışmanım tarafından "Turnitin" adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %5'tir.

Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:

ONAY

Tarih: 23 / 05 / 2025
Öğrenci Danışmanı:
Doç.Dr. Selda GÜNEY
……..

TEŞEKKÜR

Saygıdeğer Doç.Dr. Selda GÜNEY,

Doktora tez çalışmam süresince şahsıma olan destek ve katkılarınızdan ötürü teşekkürlerimi sunmak istiyorum. Bilgi ve deneyimlerinizi benimle cömertçe paylaştığınız için müteşekkirim. Yönlendirmeleriniz ve sabrınız, bu zorlu süreçte benim için büyük bir motivasyon kaynağı oldu. Sizin rehberliğiniz ile akademik yolculuğumda önemli bir mesafe kat ettim.

Saygıdeğer Prof.Dr. Faruk ELADI ve Doç.Dr. Murat ÜÇÜNCÜ,

Sizlerin doktora çalışmam süresi boyunca olan desteğiniz ve olumlu yönlendirmeleriniz kapsamında bugünlere ulaştım. Sizlere bu nedenle saygılarımı ve minnetimi sunuyorum.

Sayın Büyüklerim Verda MENGÜ Hanım ve Kadri MENGÜ Bey,

Sizlerin her zaman bana olan güveniniz ve desteğiniz için ayrıca minnetimi sunuyorum.

Sevgili Eşim ve Kızım,

Bu çalışmanın arkasında, her daim yanımda bulunan, eşim Gülda ve kızım Nilüfer Ece'ye özel bir teşekkür etmek istiyorum. Sabrınız, destek ve sevgileriniz sayesinde bu süreci daha kolay ve anlamlı hale getirdiniz. Sizin varlığınız, beni her zaman motive etti ve cesaretlendirdi. Çalışmalarımda gösterdiğiniz ilgi ve desteğin, başarıma ulaşmamda büyük bir payı olduğuna inanıyorum. Minnetle kalacağım bu yolculukta, benimle yürüdüğünüz için teşekkür ederim.

ÖZET

Reşat Ali TÜTÜNCÜOĞLU

KARA MUHAREBE ARAÇLARI İÇİN YAPAY ZEKÂ TABANLI HEDEF TESPİT SİSTEMİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Savunma Teknolojileri ve Sistemleri Anabilimdalı

2025

Modern askeri istihbarat sistemleri sürekli olarak gelişmektedir. Özellikle kara kuvvetlerinin muharebe sahasında görev aldığı ortamda, muharebe araçlarının sınıflandırması çok önemli hale gelmiştir. Farklı tipteki muharebe araçlarının yeteneklerini ve zafiyetlerini belirlemek, etkili taktik ve stratejiler geliştirmek hayati öneme haizdir. Yapay zekâ destekli olarak askeri araçların sınıflandırılmasında; görüntü işleme, kalıp tanıma ve derin öğrenme teknolojilerinin kullanımı geçmişe kıyasla daha uygulanabilir hale gelmiştir. "Sadece Bir Kez Bakarsınız - You Only Look Once" (YOLO) algoritması, nesne tespiti ve görüntü sınıflandırmasında belirgin avantajlar sunmaktadır. YOLO, hedeflerin hızlı bir şekilde tespit edilmesini sağlayarak gerçek zamanlı nesne tahminleri yapma yeteneğine sahiptir. YOLO algoritmasının; yüksek doğruluk oranı ile tespit ve değerlendirme süreçlerine katkıda bulunması, muharebe sahasındaki personele önemli bir destek sunar. Özellikle düşük arka plan hataları ile doğru sonuçlar üretebilmesi ve genel nesne temsilini anlaması, bu yöntemin etkinliğini artırmaktadır. Bu tez çalışmasında, muharebe araçlarının tespiti için YOLO Algoritması kullanılmıştır. Mobil kullanımlara uygunluğu ve performans değerlendirmelerinde gösterdiği başarı nedeniyle, tez çalışmasında YOLOv8m modeli tercih edilmiştir. Model üzerine eklenen Sıkıştırma ve Uyarım Bloğu ile performansı daha da arttırılmıştır. Çalışma, elektro-optik sistemler aracılığıyla elde edilen görüntüleri kullanarak hedeflerin detaylı özelliklerinin tespit edilmesi ve karar destek sürecine odaklanmaktadır. Bu tez çalışmasının amacı, hedef yönetim sistemlerinin temel bileşenlerinin belirlenmesi ve tanımlanmasına katkı sağlayarak karar destek süreçlerini güçlendirmektir. Süreç, özel nitelikli veri toplama, ön işleme, segmentasyon, temel özellik çıkarımı, sınıflandırma, gelişmiş özellik çıkarımı ve karar destek matrisleri kullanılarak kullanıcıya önerilerde bulunulmasını içeren derin öğrenmeye dayalı bir “Hedef Belirleme ve Tanımlama - Target Detection and Identification” (HBT - TDI) sisteminin geliştirilmesini kapsamaktadır. Geliştirilen model, görüntü işleme ve nesne tespitinde önemli avantajlar sağlamaktadır.

Ayrıca, tasarlanan model ile ilave bir donanım gerektirmeden, mevcut elektro-optik cihazlarını kullanarak muharebe araçlarının, tespitinin pasif bir şekilde tanımlanması, gözlemcinin konumunu açığa çıkarmadan mesafe tahmininde bulunarak karar desteğin sağlanması gerçekleştirilmektedir. Model, yüksek doğrulukta (mAP- mean avarege precision) %88,38 gerçek zamanlı tahminler sağlayarak, bilinçli kararlar alınmasına yardımcı olmakta ve riskleri minimize etmektedir.

ANAHTAR KELİMELER: Kara muharebe araçları, görüntü işleme, sınıflandırma, tespit, değerlendirme, yapay zekâ (YZ), derin evrişimsel sinir ağı, karar destek matrisi.

ABSTRACT

Reşat Ali TÜTÜNCÜOĞLU

AI-BASED TARGET DETECTION SYSTEM FOR GROUND COMBAT VEHICLES

Başkent Univercity Institute of Science

Department of Defence Technologies and Systems

2025

Modern military intelligence systems are continuously evolving. Particularly in environments where ground forces operate on the battlefield, the classification of combat vehicles has become critically important. Identifying the capabilities and vulnerabilities of various types of combat vehicles is essential for developing effective tactics and strategies. The use of artificial intelligence in the classification of military vehicles, through image processing, pattern recognition, and deep learning technologies, has become more feasible compared to the past. "You Only Look Once" (YOLO) algorithm provides significant advantages in object detection and image classification. YOLO possesses the ability to rapidly detect targets, enabling real-time object predictions. The algorithm's contribution to detection and evaluation processes with high accuracy offers substantial support to military personnel on the battlefield. Its ability to produce accurate results with low background errors and to understand general object representation further enhances the effectiveness of this method. In this thesis, the YOLO algorithm has been employed for the detection of combat vehicles. The YOLOv8m model has been selected due to its suitability for mobile applications and its demonstrated success in performance evaluations. The performance has been further enhanced with the addition of the squeeze and excitation block to the model. The study focuses on detecting detailed characteristics of targets using images obtained through electro-optical systems and emphasizes the decision support process. The objective of this thesis is to contribute to the identification and definition of the fundamental components of target management, thereby strengthening decision support processes. The process encompasses the development of a deep learning-based "Target Detection and Identification" (TDI) system, which includes specialized data collection, pre-processing, segmentation, basic feature extraction, classification, advanced feature extraction, and the provision of recommendations to users through decision support matrices. The developed

model offers substantial advantages in image processing and object detection. Moreover, with the designed model, the passive identification of combat vehicles can be achieved using existing electro-optical devices, without requiring additional hardware, by estimating distances while keeping the observer's position concealed. The model provides real-time predictions with high accuracy (mAP 83.6%), thereby assisting in informed decision-making and minimizing risks.

KEYWORDS: Combat military vehicles, image processing, classification, detection, assessment, artificial intelligence (AI), deep convolutional neural network, decision support matrix.

İÇİNDEKİLER

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

SİMGELER VE KISALTMALAR LİSTESİ

AA	Armoured Artillery
AI	Artificial Intelligence – Yapay Zekâ
API	Application Programming Interface- Uygulama Programlama Arabirimi
C2	Command Control – Komuta Kontrol
CNN	Convolutional Neural Network- Evrişimsel Sinir Ağı
COLAB	Colaboratory
COCO	Common Objects in Context
DEOC	Dynamic Electro- Optic Configuration
D3A	Decide, Detect, Deliver, Assess- Karar, Tespit, Dağıt ve Değerlendir
DSM	Decision Support Matrix – Karar Destek Matrisi
FC	Full Connected – Tam Bağlı
FPS	Frame Per Second- Saniyede tekrarlanan görüntü/çerçeve
HBT	Hedef Belirleme ve Tanımlama
HT	Heavy Tanks
JDL	Joint Directors of Laboratories- Müşterek Laboratuar Direktörlükleri
JP	Joint Publish- Müşterek Basım
LAW	Light Armoured Wheeled
MAP	Mean Average Precision- Ortalama Kesinlik Yüzdesi
MHAT	Medium/Heavy Armoured Tracked
MHAW	Medium/Heavy Armoured Wheeled
MRAP	Mine Resistant Ambush Protected
MT	Military Trucks
OPENCV	Open Source Computer Vision Library-Açık Kaynak Bilgisayar Görüsü Kütüphanesi
PyTORCH	Makine Öğrenme Kütüphanesi (aplikasyon uygulamaları için)
Px	Pixel -piksel
SE	Squeeze and Excitation- Sıkıştırma ve Uyarım
SSD	Single Shot Multi-Box Detector
TDI	Target Detection and Identification
TSE	Technical Spesification Entry
YOLO	You Look Only Ones- Sadece Bir kere Bakarsınız
YZ	Yapay Zekâ

1. GİRİŞ

Askerî operasyonların karmaşıklığı ve dinamik yapısı, etkili hedef yönetimi ve stratejik planlamayı gerekli kılmaktadır. Bu süreçlerin başarısı, rakibin davranışlarının doğru şekilde yönlendirilmesi ve stratejik hedeflerin başarılmasına bağlıdır. Gelişmiş bilgi teknolojileri ve yapay zekâ uygulamaları, muharebe alanında önemli değişimlere olanak sağlamıştır. Bu tez çalışması, elektro-optik sistemler aracılığıyla elde edilen görüntülerden kara muharebe araçlarının tespiti ve tanımlanması sürecini optimize etmeyi amaçlamaktadır. Hedef yönetim sistemlerinin temel bileşenlerini oluşturan tespit, izleme, teşhis ve müdahale süreçleri, derin öğrenme yöntemleri ile entegre edilmiştir. Çalışmanın odak noktası; askeri operasyonların etkinliğini artırmak, erken uyarı sistemlerini güçlendirmek ve karşı tedbirlerin en doğru şekilde gecikmeksizin planlanmasını sağlamaktır. Bu çerçevede, bu tez çalışması kapsamında geliştirilmiş bir Hedef Belirleme ve Tanımlama (HBT) sistemi sunulmaktadır.

1.1. Amaç

Askeri operasyonlar, bir ulusun siyasi hedeflerini gerçekleştirmek amacıyla dinamik ve belirsizliklerle dolu bir operasyonel ortamda yürütülen faaliyetleri kapsar. Bu operasyonların temel amacı, hasmın davranışlarını değiştirmek ve dost kuvvetlerin iradesini karşı tarafa kabul ettirmektir. Bu doğrultuda, askerî harekâtların başarısı, harekâtın tutarlı ve senkronize bir şekilde planlanmasına bağlıdır.

Askeri operasyonlar, genel olarak "planlama, hazırlık, yürütme ve sürekli değerlendirme" olmak üzere dört temel aşamadan oluşur: Bu aşamalar, birbirleriyle yakından ilişkili olup birbirini destekleyecek şekilde tasarlanmıştır.

Operasyonların etkinliğini artırmak için bu aşamaların entegre bir şekilde uygulanması önemlidir. Bu bağlamda, muharebe sahası ile ilgili istihbarat hazırlığı, risk yönetimi ve hedef yönetimi gibi entegrasyon süreçleri kritik öneme sahiptir [1]. Bu temel süreçler, askeri operasyonların başarısını sağlamak ve dinamik savaş ortamında etkinliği artırmak için kritik yapı taşlarıdır.

Hedef yönetimi süreci, NATO'nun ortak yayını JP 3-60 [2]'da potansiyel bir angajman veya başka bir eylem için hedef olabilecek varlıkların veya nesnelerin belirlenmesi,

önceliklendirilmesi ve uygun bir yanıtla eşleştirilmesi olarak tanımlanır [3]. Bu süreci yönlendiren temel yöntem ise "Karar Ver, Tespit Et, Dağıt ve Değerlendir" (Decide, Detect, Deliver, Assess- D3A) olarak bilinir. Bu yöntem, hedeflerin etkili bir şekilde yönetilmesi için gerekli olan sistematik bir yaklaşım sunmaktadır.

Askeri hedef yönetimi, öncelikli olarak hedeflerin tespit edilmesiyle başlar. Bu süreç, farklı kaynaklardan gelen verilerin analiz edilerek hedeflerin yerinin, hareketlerinin ve özelliklerinin belirlenmesini içerir. Daha sonra, hedeflerin türünün, büyüklüğünün ve gücünün belirlenmesi amacıyla hedef teşhisi gerçekleştirilir. Teşhis aşamasının ardından, hedeflerin hareketlerinin ve konumlarının sürekli olarak izlenmesi için süreçler devreye girer. Son olarak, askeri personelin güvenliğini sağlamak ve operasyonların başarısını artırmak için hedeflerin uygun bir şekilde yönlendirilmesi sağlanır. Hedeflerin etkili bir şekilde tespit edilip yönlendirilmesi, askeri operasyonların başarısında kilit bir role sahiptir [4].

Yeni teknolojilerin hızlı gelişimi, askeri hedef yönetiminde veri birleştirmeyi kritik bir unsur haline getirmiştir [5]. Farklı kaynaklardan elde edilen verilerin birleştirilmesiyle doğru ve kesin bir operasyonel tablo oluşturulabilir. Bu durum, hem askeri personelin üzerindeki yükü azaltır hem de hata oranını ve müdahale sürelerini önemli ölçüde düşürür. Veri birleştirmenin bu avantajları, askeri hedef yönetiminde operasyonel etkinliği artıran önemli bir araç olarak ön plana çıkmaktadır.

Askeri hedef yönetimi, düşman durumuna ek olarak daha yüksek kademedeki karargâhların plan ve emirlerinden gelen vazife, niyet ve görevler gibi unsurlara dayanmaktadır. Bu süreç, sürekli değişen bir savaş alanının dinamiklerine uyum sağlayan bir süreçtir.

Gelişen bilgi teknolojileri ve uygulamaları, modern savaş alanlarında büyük ve kapsamlı değişikliklere neden olmaktadır. Bilgi teknolojilerinin gelişimi, modern savaşın silahlanma, sensörler ve komuta kontrole dayalı birçok yönünü değiştirmiştir. Sensörler, silahlar ve C2 (Command Control) merkezleri, bir üç boyutlu bilgi ağına bağlıdır, böylece farklı platformlar birbirleriyle ağ bağlantısı kurar ve operasyonel kabiliyetler iyileştirilir [6].

Askeri nitelikte olan hedeflerin tespiti, hedef yönetiminde askeri karar verme sürecinin başlangıcıdır. Çok sensörlü veri birleştirme, dağıtılmış çok etmenli sistemlerde, sensör ağlarında, bilgi işlemde ve arıza tespitinde önemli bir araştırma alanıdır [7–9]. Veri

birleştirme birden fazla veri kaynağını benzersiz ve doğru bir şekilde anlamlı hale getirir. Veri birleştirmenin rolü, belirsiz gerçek zamanlı etki alanlarındaki akıllı aracıların, ortamların geliştirilmiş bir anlık görüntüsüne dayalı olarak farkında olmalarını ve harekete geçmeye hazır olmalarını sağlar. Çoklu sensör füzyonu çoğunlukla, etmenlerin algı dizilerinden eylemlere kadar haritalama üzerinde çalıştığı bir ortamı tam olarak izlemek için uygulanır.

Farklı sensörlerden gelen bilgi füzyonu, dağıtılmış askeri hedef istihbarat ortamlarında çok önemli bir bileşen haline gelmiştir. Sensörlerden gelen bilgileri yorumlayan operatörlerin, gözetim yeteneklerini sağlamak için tehdit türlerini tahmin etmesi ve tanımlaması gerekir. Bir etmenin, arama modunda herhangi bir hedefi tanıması durumunda, denetim görevini başarıyla tamamlamak için hedefi izlemeli türünü ve konumunu belirlemelidir.

Sonuç olarak, askeri hedef yönetimi, askeri operasyonların başarısı için kritik bir rol oynar. Bu süreç, askeri hedeflerin tespiti, izlenmesi, teşhisi ve müdahalelerinin yönetimiyle ilgilidir. Sürekli değişen bir savaş alanındaki dinamiklere ayak uydurabilen bir süreçtir ve askeri personelin güvenliği ve operasyonların başarısı için önemlidir.”

1.2. Kapsam

Bu tez, askeri operasyonel planlama süreçlerine katma değer sağlayarak operasyonel etkinliği artırmayı ve askeri birliklerin sahada karşılaştıkları tehditlere karşı hazırlıklı olmalarını desteklemeyi amaçlamaktadır.

Çalışma kapsamında elektro-optik sistemler ile elde edilen görüntüler analiz edilecek, analiz sonucunda elde edilen bilgiler ilave girdilerle beraber tekrar işleme sokularak mesafe ölçümü yapılacaktır. Elde edilen tüm bilgiler, karar destek matrisleri yardımıyla işlenerek kullanıcıya sunulacaktır. Çalışmada, ön işleme, bölütleme, öznitelik çıkarma ve sınıflandırma gibi adımları içeren derin öğrenme tabanlı bir Hedef Belirleme ve Tanımlama (HBT-TDI) sisteminin geliştirilmesi hedeflenmektedir. Bu sistem sayesinde hedeflerin yorumlanması ve önerilerin sunulması amaçlanmaktadır. Bu hedef doğrultusunda, kara muharebe araçlarının etkin bir şekilde yönetilmesi için sistematik bir yaklaşım ortaya konmuştur.

Bu kapsamda, çalışma aşağıdaki paragraflarda özetlenen alanlara odaklanmaktadır:

Görüntü İşleme ve Analiz: Elektro-optik sistemler aracılığıyla elde edilen görüntülerin işlenmesi, bölütlenmesi ve özniteliklerin çıkarılması. Bu süreçler, hedeflerin doğru bir şekilde sınıflandırılması ve tanımlanmasına olanak sağlayacaktır.

Transfer Öğrenme Algoritması: Modelin tespit ve teşhisi için eğitilmesini müteakip, modelin öğrendiği bilgilerin kullanılan hedefin mesafesinin tahmin edilmesi sağlanacaktır. Bu süreç özellikle yeni bir problem üzerinde daha hızlı sonuçlar almak istediğimizde oldukça faydalıdır.

Derin Öğrenme Tabanlı Algoritmalar: HBT-TDI geliştirilerek, tespit ve sınıflandırma işlemleri derin öğrenme algoritmaları ile yapılacaktır. Bu sayede, hedefler hızlı ve doğru bir şekilde tanımlanabilir.

Karar Destek Sistemleri: İşlenen veriler, karar destek matrisleri aracılığıyla değerlendirilerek, kullanıcıya somut ve eyleme dönük öneriler sunacaktır. Bu sistem, askeri operasyonların dinamik yapısına uygun, esnek ve hızlı karar almayı kolaylaştıran bir yapıda tasarlanacaktır.

Kara Muharebe Araçlarının Yönetimi: Sistem, kara muharebe araçlarının etkin bir şekilde yönetilmesi için stratejik ve taktiksel çözümler sunacaktır. Araçların tespiti, konumlarının belirlenmesi ve uygun karşı tedbirlerin alınması için sistematik bir yaklaşım benimsenmiştir.

1.3. Yöntem

Kara muharebe araçları, silahlı çatışmalarda kullanılan ve özel tasarımlara sahip askeri araçlardır. Bu araçlar, temel işlevleri ve özelliklerine göre farklı şekil ve boyutlarda kategorize edilir. Kara muharebe araçlarının yeteneklerini ve sınırlamalarını anlamak, askeri strateji ve taktiklerin temel bir parçasıdır. Bu bilgi, komutanların her tür araç ve ekipmanın güçlü yönlerinden en iyi şekilde faydalanmasını ve zayıf yönlerini minimize etmesini sağlar.

Özellikle savaş alanında rakip kara araçlarının tanımlanması ve analiz edilmesi, hedeflerin önceliklendirilmesi, taktik karar verme, ateş gücü planlaması, kuvvet koruma, keşif, istihbarat ve durumsal farkındalık gibi kritik unsurlar açısından büyük önem taşır. Bu analizler, modern savaş alanında etkin bir hedef yönetimi süreci için güçlü bir temel oluşturur.

Yapay zekâ ile desteklenen sistem, düşman kara muharebe araçlarının sınıflandırılmasını kolaylaştırarak, hızlı ve doğru bir istihbarat akışı sağlar. Bu, operatörlerin üzerindeki yükü azaltır ve askeri yeteneklerin gelişmesine katkıda bulunarak savaş alanında bir kuvvet çarpanı rolü üstlenir. Sistemin performansını artırmak amacıyla, mevcut literatürdeki başarılı ön işleme ve segmentasyon algoritmaları incelenmiş ve uygulanmıştır. Ayrıca, yapay sinir ağlarındaki dikkat mekanizmalarının hedef sonuçlar üzerindeki etkisi detaylı bir şekilde analiz edilerek, sistemin yüksek sınıflandırma doğruluğuna ulaşması amacıyla derin öğrenme modeli optimize edilmiştir.

Hedef tespiti ve yönetimine yönelik geliştirilmiş HBT sistemi, tespit edilen kara muharebe araçlarının mesafelerini veri tabanlı yöntemlerle ölçmekte ve bu bilgilerin ışığında en uygun karşı tedbir (ateş ve ateş destek vasıtaları) önerilerini sunmaktadır. Bu sistem, askeri planlamalar esnasında hedef izleme ve kaynak tahsis süreçlerine etkin bir destek sağlayacak şekilde tasarlanmıştır.

Sonuç olarak, geliştirilen yöntemler, çağdaş savaş alanlarında rekabet avantajı elde etmek için sağlam bir zemin hazırlamakta ve çalışmanın hedeflerine ulaşılmasında kritik bir rol oynamaktadır. Tezin bütünü göz önünde bulundurulduğunda, yöntem kısmı, teorik çerçevenin pratik uygulamalara dönüştürülmesi ve elde edilen verilerin askeri operasyonlarda stratejik kullanımı açısından kapsamlı bir yaklaşım sunmaktadır.

2. LİTERATÜR ÖZETİ

2.1. Genel Literatür

Hedeflerin belirlenmesi konusunda literatürde genel olarak iki ana çalışma alanı tanımlanmıştır:

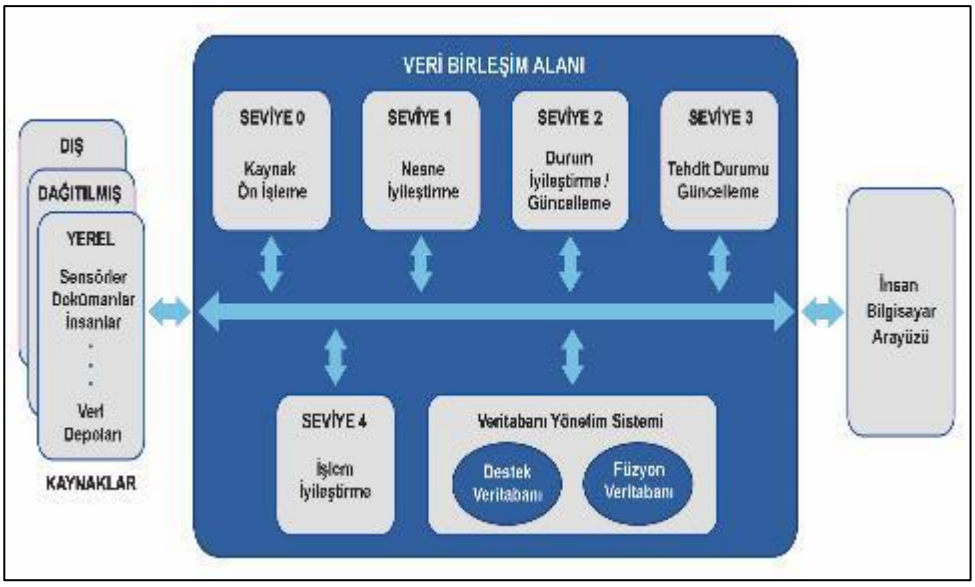
Veri Füzyonu (Data Fusion),

Silah Hedef Belirleme/Görevlendirme (Weapon Target Assignment).

Bu çalışmada, hedef yönetimi bağlamında özellikle kara hedeflerinin tespiti, belirlenmesi, teşhisi ve tanımlanmasıyla ilgili literatür incelemeleri yapılmıştır. Bu kapsamda, modern yapay zekâ teknikleri ve derin öğrenme yöntemlerinin literatüre olan katkıları incelenmiştir.

Evrişimsel Sinir Ağları [Convolutional Neural Network (CNN)] algoritmaları, hedeflerin tespiti, sınıflandırılması ve teşhisinde etkili bir şekilde kullanılabilmektedir. Geleneksel yöntemler, elle öznitelik çıkarma ve makine öğrenmesi tabanlı yaklaşımlar nedeniyle sınırlı bir tanısal doğruluk sunmaktadır. Bu yetersizlikler, araştırmacıları derin öğrenme yöntemlerini benimsemeye yönlendirmiştir. Derin öğrenme, karmaşık ve hiyerarşik öznitelikleri otomatik olarak keşfederek tanımlama sistemlerinin doğruluk ve verimliliğini artırmaktadır. Özellikle CNN ve transfer öğrenme yöntemleri, tanımlama sistemlerinin performansını önemli ölçüde geliştirmiştir. Derin öğrenme yaklaşımlarının sağladığı bu avantajlar, askeri hedef tespiti alanındaki yenilikçi uygulamalara yön vermektedir.

Tezde, görüntülerde hedef tespiti, sınıflandırılması ve teşhisine yönelik farklı CNN ve transfer öğrenme yaklaşımlarının uygulanması ele alınmıştır. Bu kapsamda, sınıflandırma performansını artırmak amacıyla öznitelik seçimi, bölütleme aşamaları ve optimizasyon algoritmalarının entegrasyonu incelenmiştir. Ayrıca, kullanılan derin ağ modelinin hiperparametre optimizasyonu ile en uygun parametre kombinasyonunun elde edilmesine yönelik çalışmalar özetlenmiştir. Bu çalışmalar, hedef tespiti ve sınıflandırma süreçlerinde kullanılan yöntemlerin etkinliğini artırmayı amaçlamaktadır.



Şekil 2.1. JDL Model 4 Veri Füzyonu

Veri füzyonu, farklı sensörlerden gelen verilerin entegre edilerek anlamlı bilgiler haline getirilmesi sürecidir. Bu alanda ABD Ortak Laboratuvarlar Direktörlüğü'nün (JDL) geliştirdiği Şekil 2.1'de gösterilen veri füzyon modeli, yaygın bir şekilde kullanılmaktadır. JDL Model 4 olarak bilinen bu model, beş seviyeden oluşmaktadır.

Seviye 0, sensör verilerinin ön işleme faaliyetleri ile ilişkilidir.

Seviye 1, bir hedefin durumunu tahmin etmeye yönelik nesnel iyileştirme süreçlerini kapsar.

Seviye 2, varlıklar ve durumlar arasındaki ilişkilerin belirlenmesi için durum iyileştirme süreçlerine odaklanır.

Seviye 3, düşman tehditlerinin tanımlanması ve değerlendirilmesine yönelik tehdit analizi süreçlerini içerir.

Seviye 4 ise, tüm veri birleştirme sürecini izleyen ve kaynakları etkin bir şekilde yöneten süreç iyileştirme faaliyetlerini kapsamaktadır [10]. JDL modeli, veri füzyonunun çeşitli aşamalarında kullanılan sistematik bir çerçeve sunarak askeri uygulamalarda yaygın bir kullanım alanı bulmaktadır.

S. K. Das [11], hedef sınıflandırma ve füzyonunu birleştiren bir ara füzyon modeli (Seviye 2 ve 3 Füzyonu) geliştirmiştir. Bu model, durum ve tehdit değerlendirmelerini

iyileştirmeyi amaçlamaktadır. Model kapsamında, sensör verileri analiz edilerek gözlemler gerçek dünyadaki nesnelere (örneğin, uçak, zırhlı araç ve füze fırlatıcı) göre sınıflandırılmıştır. Bu süreçte, Naive Bayes sınıflandırıcısı, Kural Tabanlı Uzman Sistemler, Dempster-Shafer Teorisi ve Bulanık Mantık (Fuzzy Logic) gibi yöntemler kullanılmıştır. Ayrıca, hedef toplama (aggregation) sürecinde, birimler bir dizi alt kategori (örneğin, müfrezeler, bölükler ve filolar) veya belirli bir durum (örneğin, pusu ve geri çekilme) şeklinde temsil edilmiştir. Dinamik operasyonel ortamda, mekânsal ve zamansal küme kalıplarını belirlemek için örnek kümeleme teknikleri geliştirilmiştir. Bu model, özellikle tehdit değerlendirmesi ve operasyonel karar alma süreçlerinde önemli bir fayda sağlamaktadır.

P. Sharma ve arkadaşları [12], nesne algılamanın görüntü analizi ile olan güçlü ilişkisi nedeniyle son yıllarda büyük ilgi gördüğünü belirtmişlerdir. Araştırmacılar, derin öğrenme yöntemlerinin hızla gelişmesiyle birlikte nesne algılama uygulama alanlarının da genişlediğini vurgulamışlardır. Özellikle, görüntü ve videolarda nesne tespiti için regresyon ve sınıflandırma tabanlı modellerin kullanılmasının, derin öğrenme yöntemlerinin performansını artırdığına dair önemli bulgular ortaya konmuştur. Bu tür çalışmalar, derin öğrenme modellerinin operasyonel uygulamalardaki başarısını artırmaya yönelik önemli ipuçları sunmaktadır.

Z. Li ve arkadaşları [13], geniş alan uzaktan algılama sistemlerini kullanarak görüntülerdeki hedefleri tespit etmek ve belirlemek amacıyla bir yöntem geliştirmişlerdir. Bu yöntem, yüksek çözünürlüklü optik uzaktan algılama uydularını kullanarak suüstü hedeflerinin gerçek zamanlı tespitinde başarılı sonuçlar elde etmiştir. Geniş alan uzaktan algılama görüntülerinde hedef çıkarma ve tanımlama süreci, morfolojik eşleştirme ve derin öğrenme teknikleri ile gerçekleştirilmiştir. Araştırmacılar, hava görüntülerindeki nesne tespiti, keşif, hedef gözetleme ve savaş hasarı değerlendirmesi gibi askeri uygulamalar için bu yöntemlerin oldukça etkili olduğunu göstermişlerdir. Ancak, aydınlatma farklılıkları, sahne karmaşıklığı ve platform hareketi gibi zorlukların bu görevlerin etkinliğini azaltabileceğini de belirtmişlerdir. Bu sorunların üstesinden gelmek için, havadaki videolardan nesne tespitine yönelik yeni bir konvolüsyonel sinir ağı (CNN) modeli geliştirilmiştir. Önerilen model, üç seviyeli derin bir CNN yapısına sahip olup nesneleri kaba ve detaylı bir şekilde tahmin edebilmektedir. Deneysel sonuçlar, bu yöntemin üstün performans sergilediğini göstermiştir. Bu tür yöntemler, modern savaş alanlarında gerçek zamanlı hedef tespiti ve gözetlemenin etkinliğini artırmayı hedeflemektedir.

Selbes ve Sert [14], çalışmalarında önceden eğitilmiş ağlarının kullanımına yönelik algoritmaları incelemişlerdir. Çalışma kapsamında, AlexNet ve GoogleNet mimarileri kullanılmıştır. AlexNet, beş konvolüsyon katmanı ve üç tamamen bağlı (FC) katmandan oluşurken, GoogleNet yirmi iki katmana sahiptir. Araştırmacılar, görsel özelliklerin çıkarılmasının yanı sıra ses özelliklerini de analiz ederek bu verileri birleştirmiş ve SVN sınıflandırması ile tipleri belirlemişlerdir. Çalışmanın sonuçları, yalnızca görsel verileri kullanan GoogleNet'in daha iyi bir performans sergilediğini ortaya koymuştur. Bu bulgular, önceden eğitilmiş ağların askeri hedef tespiti ve sınıflandırma alanındaki potansiyelini vurgulamaktadır.

Bir diğer çalışmada, A. Krizhevsky ve arkadaşları [15], 1,2 milyon yüksek çözünürlüklü görüntü verisini sınıflandırmak için büyük ve derin evrişimli sinir ağını (CNN) eğitmişlerdir. Eğitimi hızlandırmak amacıyla doymamış nöronlar ve evrişim işleminin çok verimli bir GPU uygulamasını kullanmışlardır. Ayrıca, tam bağlı (fully connected) katmanlarda aşırı uyumu (overfitting) önlemek için "Drop-out" adı verilen bir düzenleme yöntemi geliştirmişlerdir. Bu yöntem, yüksek doğruluk oranlarına ulaşılmasını sağlamış ve büyük veri kümelerinde rekor kıran doğruluk oranlarını elde edilmiştir. Çalışma, derin CNN mimarilerinin tamamen denetimli öğrenme süreçlerinde üstün başarı sergileyebileceğini kanıtlamıştır. Bu çalışma, derin öğrenme modellerinin büyük veri kümeleri üzerindeki üstün performansını açıkça ortaya koymaktadır.

Suprayitno ve arkadaşları [16], çalışmalarında evrişimli sinir ağlarını (CNN) ve "elektrostatik kayıp" adı verilen yeni bir kayıp fonksiyonunu kullanarak gerçek zamanlı hedef tespiti sağlayabilen bir sistem tasarlamışlardır. Sistem, askeri personelin (savaşan veya savaşmayan) ve sivillerin varlığını tespit etmeye odaklanmıştır. Araştırmacılar, elektrostatik kaybın, derin metrik öğrenme yöntemlerinde alternatif bir kayıp fonksiyonu olarak etkili bir şekilde kullanılabileceğini göstermişlerdir. Bu sistem, gerçek zamanlı hedef tespiti için yenilikçi bir yaklaşım sunmakta ve uygulama alanında önemli bir potansiyel taşımaktadır.

Huang ve arkadaşları [17], insansız hava araçları (İHA) üzerinde çalışan gömülü bir gerçek zamanlı hedef tespit sistemi geliştirmişlerdir. Sistem performansını ve kararlılığını doğrulamak amacıyla araç algılama örneğiyle test edilmiştir. Çalışmanın temel amacı, gömülü teknoloji kullanarak gerçek zamanlı görüntü elde etmek, hedef algılama sonuçlarını iletmek ve kullanıcılar arasındaki etkileşimi desteklemektir. Ayrıca, bu etkileşimlerin birden fazla İHA ve kullanıcı arasında paylaşılabilmesine olanak tanınmıştır. Dinamik hedef tespiti

içeren deneyler sonucunda, %88,2 genel güvenilirlik oranı ve %4,6 kaçırılan algılama oranı elde edilmiştir. Bu çalışma, İHA'lar üzerinde gerçek zamanlı hedef tespiti için yüksek performanslı bir çözüm sunmaktadır.

Vaswani ve arkadaşları [18], yapay sinir ağlarında dikkat (attention) mekanizmasını kullanarak çeviri performansını önemli ölçüde artırmayı başarmışlardır. Dikkat mekanizması, bir sorguyu bir dizi anahtar-değer çifti ile eşleştirerek çıktıyı oluşturur. Çıkışlar, her bir değerin ağırlıklandırılmış toplamı olarak hesaplanır ve bu ağırlıklar bir uyumluluk fonksiyonu aracılığıyla belirlenir. Bu mekanizma, özellikle dil işleme uygulamalarında yüksek performans sağlamıştır. Dikkat mekanizmasının yapay sinir ağlarında uygulanması, hedef tanımlama ve analiz süreçlerinde de etkili çözümler sunabilecek bir potansiyele sahiptir.

2.2. Görüntü Algılama ve Sınıflandırmalarına Yönelik Literatür

Tez çalışmamızın ilk aşamasında gerçekleştirilen genel literatür taramasını takiben, görüntülerde hedef tespiti ve sınıflandırılmasına yönelik çalışmalar detaylı bir şekilde incelenmiş ve bu çalışmaların algoritmik değerlendirmelerine ilişkin kapsamlı bir literatür analizi yapılmıştır. Bu literatür incelemesi, bilgisayarlı görme alanındaki nesne algılama ve sınıflandırma algoritmalarının etkili bir şekilde analiz edilmesine odaklanmıştır.

Bilgisayarlı görme alanında nesne algılama ve sınıflandırma algoritmaları arasında önemli farklar bulunmaktadır. Nesne algılama algoritmaları, ilgilenilen nesnelerin etrafına sınırlayıcı kutular yerleştirerek bir görüntü içindeki nesneleri tanımlamayı ve konumlandırmayı amaçlamaktadır. Bu algoritmaların ana görevi, nesnelerin görüntü çerçevesi içindeki pozisyonlarını doğru bir şekilde belirlemektir. Ancak, algılama sürecinde nesnelerin sayısının ve sınırlayıcı kutuların önceden bilinmemesi, algoritmaların karmaşıklığını artırmakta ve çözüm süreçlerini zorlaştırmaktadır. Bu zorlukların üstesinden gelmek için farklı yaklaşımlar ve algoritmalar geliştirilmiştir.

Bu tür sorunların üstesinden gelmek için Evrişimsel Sinir Ağları (CNN) kullanılarak görüntüdeki farklı ilgi bölgelerinin belirlenmesi ve bu bölgelerde yer alan nesnelerin sınıflandırılması önerilmektedir. Ancak, bu yaklaşımda nesnelerin görüntüde farklı konumlarda ve çeşitli en-boy oranlarında bulunabilmesi, tespit sürecini daha karmaşık hale getirmektedir [19]. Bu karmaşıklık, tek ve iki aşamalı hedef tespit algoritmalarının

geliştirilmesine neden olmuştur.

Araç türlerini tespit etmek ve sınıflandırmak amacıyla araştırmacılar tarafından çeşitli algoritmalar geliştirilmiştir. Bu algoritmalar genel olarak tek aşamalı ve iki aşamalı hedef tespit algoritmaları olarak kategorize edilmektedir. İki aşamalı hedef tespit algoritmalarına örnek olarak Bölge Tabanlı Evrişimsel Sinir Ağları (R-CNN), FAST R-CNN ve FASTER R-CNN algoritmaları verilebilir [20]. Tek aşamalı algoritmalar arasında ise YOLO (You Only Look Once) öne çıkmakta olup, sınıflandırma ve regresyon işlemlerini doğrudan farklı konumlardan gerçekleştirmektedir [21]. Bu algoritmaların yapısal farklılıkları, tespit performanslarını ve kullanım alanlarını etkilemektedir.

Son yıllarda bilgi işlem cihazlarının işlem kapasitesindeki artış ve etiketli görüntü verilerinin çoğalması, Evrişimsel Sinir Ağı (CNN) mimarilerinin başarısını önemli ölçüde artırmıştır. CNN mimarileri, özellikle görsel nesne sınıflandırma görevlerinde yaygın bir şekilde kullanılmaktadır. Bu gelişmeler, CNN tabanlı tespit algoritmalarının performansını artırarak yeni uygulamaların önünü açmıştır.

R-CNN algoritması, çok sayıda bölge seçme problemini çözmek için Ross Girshick ve ekibi [22] tarafından geliştirilmiştir. Algoritma, bir görüntüden yalnızca 2000 bölge önerisi çıkarmak için seçici arama yöntemini kullanmaktadır. Bu yaklaşım sayesinde, tüm bölgeleri sınıflandırmak yerine yalnızca belirlenen 2000 bölge üzerinde çalışılmakta, böylece işlem yükü azaltılmaktadır. R-CNN'in bu özelliği, bölge önerileri aracılığıyla sınıflandırma sürecini hızlandırmaktadır.

R-CNN algoritması, tespit problemini iki alt probleme ayırır. İlk olarak, kategoriden bağımsız bir şekilde nesne konum önerileri oluşturmak için renk ve doku gibi düşük seviyeli ipuçlarını kullanır. İkinci aşamada, bu konumlarda yer alan nesnelerin kategorilerini tanımlamak için CNN tabanlı sınıflandırıcıları kullanır. Bu iki aşamalı yaklaşım, sınırlayıcı kutuların doğruluğunu artırırken modern CNN'lerin güçlü sınıflandırma yeteneklerinden de yararlanır [22]. Bu iki aşamalı yaklaşım, nesne tespit süreçlerinde doğruluk oranlarını artırmıştır, ancak bazı sınırlamalara da sahiptir.

Bu tür algoritmaları eğitmek uzun zaman almaktadır. Bunun sebebi her bir görüntü için 2000 bölge önerisine ihtiyaç duyulmaktadır. Bu nedenle, R-CNN algoritması gerçek zamanlı uygulamalara uygun değildir. Her test görüntüsünün işlenmesi ortalama 47 saniye sürmektedir. Ayrıca, seçici arama algoritması sabit bir yapıya sahiptir ve herhangi bir

öğrenme süreci içermez. Bu durum, kötü aday bölge önerilerinin ortaya çıkmasına neden olabilir [19]. Bu sınırlamaları gidermek için Faster R-CNN ve YOLO gibi gelişmiş algoritmalar geliştirilmiştir.

Faster R-CNN algoritmasında, görüntü önce evrişimli bir ağ tarafından işlenerek bir özellik haritası oluşturulur. Bu algoritma, seçici arama yerine bölge önerilerini tahmin etmek için ayrıştırılmış bir ağ kullanır. Tahmin edilen bölge önerileri daha sonra, görüntünün ilgili bölgede sınıflandırılması ve sınırlayıcı kutuların uzaklık değerlerinin tahmin edilmesi amacıyla bir RoI (Region of Interest) havuzlama katmanı aracılığıyla yeniden şekillendirilir. Faster R-CNN'in bu yenilikçi yaklaşımı, nesne tespit sürecindeki verimliliği artırmıştır.

YOLO, önceki nesne algılama algoritmalarından farklı olarak, nesne lokalizasyonu için görüntünün belirli bölgelerini incelemek yerine görüntünün tamamını tek bir seferde analiz eder. Bu algoritma, tek bir Evrişimsel Sinir Ağı (CNN) kullanarak görüntüdeki sınırlayıcı kutuları oluşturmaktadır. Algoritma, bu kutuların sınıf olasılıklarını tahmin eder. YOLO, nesne tespit sürecini tek bir aşamada gerçekleştirmesiyle bölge tabanlı algoritmalardan ayrılmaktadır. Bu özelliği, hızlı ve etkili bir tespit sağlamasına olanak tanır [23]. YOLO'nun bu yenilikçi yaklaşımı, nesne algılama alanında hız ve doğruluk açısından önemli avantajlar sağlamaktadır.

YOLO'nun çalışma prensibi, görüntüyü bir SxS ızgarasına bölmektir. Her bir ızgara hücresinde, sınırlayıcı kutuların sayısı ve bu kutulara ait sınıf olasılıkları ile ofset değerleri tahmin edilir. Kullanıcı tarafından belirlenen eşik değerinin üzerinde sınıf olasılığına sahip olan sınırlayıcı kutular seçilir ve bu kutular nesnelerin görüntü içindeki konumlarını belirlemek için kullanılır. Bu yapı, YOLO algoritmasının hızını artırırken, küçük nesnelerle ilgili sınırlamaları da beraberinde getirmektedir.

YOLO algoritmaları, diğer nesne algılama algoritmalarına kıyasla çok daha hızlıdır. Ancak, bu algoritmanın temel sınırlamalarından biri, küçük nesnelerin tespiti konusunda yaşadığı zorluklardır. Özellikle bir grup küçük nesnenin algılanması, algoritmanın uzaysal kısıtlamalarından dolayı zorlaşabilmektedir [24]. Bu sınırlamaları aşmak için YOLO algoritmasının çeşitli sürümleri geliştirilmiştir.

YOLO algoritmaları, görüntü algılama ve sınıflandırma alanında geliştirilen son teknoloji ürünü, gerçek zamanlı bir nesne algılama sistemidir [24]. YOLO, kendisinden önceki nesne algılama algoritmaları tarafından kullanılan ve sınıflandırıcıları algılama

yapmak üzere yeniden kullanan stratejinin aksine, sınırlayıcı kutuların ve sınıf olasılıklarının tahminlerini aynı anda sağlayan uçtan uca bir sinir ağının kullanılmasını önermektedir. YOLO, gerçek zamanlı nesne detektörlerine kıyasla hız, sınırlayıcı kutularda birleşime göre daha iyi kesişme ve gelişmiş tahmin doğruluğu gibi doğal bir avantaja sahiptir. YOLO, 45 FPS (Frame per second- Saniyede görüntülenen çerçeve sayısı)'e kadar çalışır. Bu özelliği ile rakiplerinden çok daha hızlı bir algoritma olarak öne çıkmaktadır. YOLO'nun mimarisi, sonunda 2 tamamen bağlı katman olmak üzere toplam 24 evrişimli katmana sahiptir. YOLO ile ilgili karşılaşılan sorunlar ise, küçük nesnelerin gruplar halinde tanımlanması ve yerelleştirme doğruluğudur.

YOLOv5, PyTorch çerçevesini kullanarak daha küçük bir boyut, yüksek performans ve gelişmiş entegrasyon gibi birçok avantaj sunmaktadır [25]. Bu özellikler, YOLOv5'in gömülü cihazlara kolaylıkla entegre edilmesini ve bu cihazlarda etkin bir şekilde çalışmasını sağlamaktadır [26]. Bu avantajlar, YOLOv5'in kullanım alanını genişleterek, nesne algılama sistemlerinde öne çıkmasını sağlamıştır.

Olorunshola ve arkadaşları [27], YOLOv5 ve YOLOv7 modellerinin karşılaştırmalı analizini gerçekleştirmiştir. Deney sonuçları, bu modellerin diğer nesne algılama çalışmalarıyla karşılaştırıldığında önemli katkılar sağladığını ortaya koymuştur. Ayrıca, tespit modellerinin kurulum ve kullanım kolaylıkları ile farklı değerlendirme metrikleri açısından incelendiği bu çalışma, YOLOv5'in YOLOv7'ye kıyasla gerçek zamanlı nesne tespiti açısından daha etkili olduğunu göstermiştir [27]. YOLOv7, daha iyi tamponlama ve hedefleme stratejileri ile birlikte gelirken, YOLOv5 yüksek verimlilik odaklı optimizasyonu ile dikkat çekmektedir. Bu sonuçlar, YOLOv5'in bazı açılardan üstünlüğünü ortaya koyarken, YOLO algoritmalarının sürekli gelişime açık bir yapı sunduğunu göstermektedir.

2.3. Performans Artırımı ve Mesafe Ölçümü Çalışmalarına Yönelik Literatür

Tespit ve teşhis süreçlerinin performansını artırmaya yönelik çeşitli çalışmalar yapılmıştır. Bu çalışmalar, özellikle hedeflerin bakış açısı sorunları ve karmaşık arka plan problemlerine odaklanmıştır. Bu problemleri çözmek için geliştirilen algoritmalar, hedef tespiti ve mesafe ölçümünde doğruluk oranlarını artırmayı amaçlamaktadır.

R. Zhang ve arkadaşları [28], YOLOv5 algoritmasında bir dikkat mekanizması geliştirerek bakış açısı sorunlarını çözmeye yönelik bir yaklaşım sunmuşlardır. Bu yöntem,

hedeflenen bakış açılarının geometrik hesaplamalarla düzeltilmesini ve hedeflerin daha doğru bir şekilde sınıflandırılmasını sağlamıştır. Hedef tespit doğruluğunu artırmak için farklı odak alanlarında da iyileştirmeler yapılmıştır.

C. Cui ve arkadaşları [29], optik uzaktan algılama görüntülerindeki karmaşık arka plan problemlerine çözüm bulmak amacıyla hibrit evrişim kaynaşmış dikkat mekanizmasına dayalı bir YOLOv7 optimizasyon algoritması geliştirmişlerdir. Bu algoritma, küçük hedeflerin yoğun dağılımı, hedef ölçeklerindeki büyük farklılıklar ve karmaşık arka plan gibi sorunlarla etkili bir şekilde başa çıkılmasını sağlamaktadır. Hedef tespitindeki bu iyileştirmeler, mesafe ölçüm algoritmalarına entegre edilerek daha kapsamlı sistemler geliştirilmesine olanak tanımaktadır.

J. Hu ve arkadaşları [30], ağların temsili gücünü artırmak amacıyla "Squeeze-and-Excitation" (SE) bloğu olarak adlandırılan yeni bir mimari birimi önermiştir. Bu blok, kanallar arasındaki bağımlılıkları açık bir şekilde modelleyerek kanal bazında özellik tepkilerini adaptif olarak yeniden ölçeklendirmektedir. Araştırmaları, bu SE bloklarının üst üste konulmasıyla zorlu veri kümeleri üzerinde son derece iyi genelleştiren SENet mimarilerinin oluşturulabileceğini göstermektedir. Özellikle, SE bloklarının mevcut derin mimariler için minimal ek hesaplama maliyeti ile önemli performans iyileştirmeleri sağladığı ifade edilmiştir.

W. Tianyong ve arkadaşları [31], küçük nesnelerin ve farklı ölçeklerdeki nesnelerin tespitine ilişkin sorunları çözmek amacıyla YOLOv8 ağı ile ilgili olarak YOLO-SE algoritmasını önermektedir. Çalışmada önce YOLO-SE mimarisinin genel bir görünümü sunulmakta, ardından bu mimarinin temel bileşenleri tanıtılmaktadır. Dikkat mekanizmasının tanıtılmasıyla gerçekleştirilen konvolüsyon iyileştirmeleri, orijinal YOLOv8 tespit başlığının bir transformatör tahmin başlığı ile değiştirilmesi ve farklı ölçeklerdeki nesneleri ele almak için ek bir tespit başlığının eklenmesi yer almaktadır.

Muharebe sahasında tespit edilen bir hedefin teşhis oranı ve bizim konumumuza göre yerinin (mesafesinin) tanımlanması, karar vericiler açısından son derece önemlidir. Bir tehdidin üzerimizdeki etkisinin ne zaman kritik bir duruma geleceği konusunda, mesafenin belirlenmesi büyük önem taşır. Ayrıca, belirlenen mesafeye göre alınabilecek tedbirler de farklılık gösterecektir. Bu nedenle, tespit ve teşhis algoritmalarının yanı sıra hedefin yerini (mesafesini) de tespit edebilmek önemli hale gelmektedir. Mesafe tahmin algoritmaları,

genel olarak binoküler veya monoküler pasif sensörlerle çalışmaktadır.

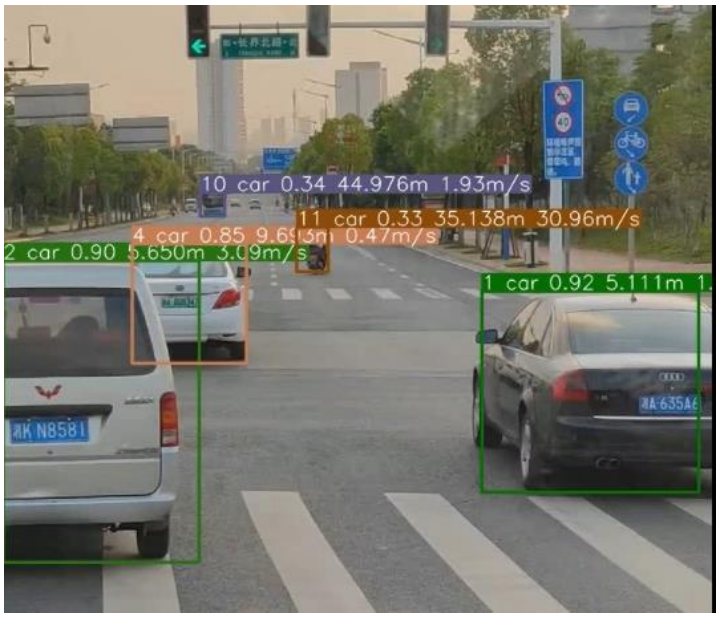
M. Lan ve arkadaşları, YOLOv8 ağ modelini kullanarak bir füzyon duyusal uzaklık modeli geliştirmiştir. Bu model, binoküler uzaklık sistemine entegre edilmiş ve optimize edilmiş verilerle çalışmaktadır. Araştırma sonuçlarına göre, modelin statik testlerde ortalama hata oranı %0,5'in altına düşürülmüş, dinamik testlerde ise hata oranı %1,75'ten %0,86'ya çekilmiştir. Ayrıca, modelin gömülü geliştirme kartlarında yüksek taşınabilirlik sağladığı gösterilmiştir [32]. Bu yenilikçi model, mesafe ölçüm doğruluğunu artırırken, operasyonel alanlarda kullanım kolaylığı sunmaktadır.

Geleneksel binoküler mesafe belirleme yöntemlerinde düşük hedef tespit doğruluğu ve yüksek mesafe hata sorunlarını çözmek amacıyla, B. Wei ve ekibi, YOLOv5 tabanlı bir binoküler görüş menzil algoritması geliştirmiştir. Algoritma, hata oranlarını minimize etmek amacıyla tasarlanmıştır. Mesafe ölçümü işlemi, stereo eşleştirme ve binoküler uzaklık formülü kullanılarak gerçekleştirilir. Ayrıca, hedef merkez noktasının derinlik değeri, 2 boyutlu piksel koordinat sisteminden 3 boyutlu uzay koordinat sistemine dönüştürülerek hesaplanmıştır [33]. Bu tür geliştirmeler, özellikle gerçek zamanlı mesafe ölçüm ihtiyaçları için önemli çözümler sunmaktadır.

Çalışmamızın temel ihtiyacı, monoküler bir elektro-optik sistem kullanarak pasif bir mesafe ölçümü gerçekleştirmektir. Bu yaklaşım, maliyetleri düşürmenin yanı sıra mesafeyi doğruya yakın bir şekilde tespit etmeyi amaçlamaktadır. Ayrıca, bu işlemlerin gerçek zamanlı görüntü işleme üzerinde uygulanabilir olması gerekmektedir. Bu hedefe yönelik olarak literatürde çeşitli çalışmalar yer almaktadır.

Bu çalışmalar arasında özellikle YOLO algoritmalarını temel alan mesafe tahmin yöntemleri dikkat çekmektedir.

G. Joccher [34], YOLOv5 algoritmasını kullanarak nesnelerin kutu oranlarına dayalı mesafe tahmini yapmaya yönelik bir çalışma gerçekleştirmiştir. Bu çalışma, YOLOv5'in güçlü nesne tespit yeteneklerinden faydalanarak görüntüdeki nesnelerin boyut ve oranlarına göre mesafelerini doğru bir şekilde tahmin etmeyi amaçlamaktadır (Şekil 2.2). Bu çalışma, mesafe tahmin algoritmalarının nesne tespit teknolojileri ile entegrasyonunun önemini vurgulamaktadır.



Şekil 2.2. YOLOv5 Algoritması Kullanılarak Yapılan Mesafe Tahmini

M.A. Khan ve arkadaşları [35], sınırlayıcı kutuların ağırlık merkezlerini ve nesnelerin bilinen boyutlarını kullanarak iki nesne arasındaki mesafeyi hesaplamaya yönelik bir yöntem geliştirmiştir. Bu yöntem, sınırlayıcı kutuların ağırlık merkezlerini temel alarak nesneler arasındaki boşluğun doğru bir şekilde ölçülmesini sağlamaktadır. Bu tür yaklaşımlar, mesafe ölçüm algoritmalarında doğruluk ve etkinlik açısından önemli iyileştirmeler sunmaktadır.

Marek Vajgl ve arkadaşları [36], yalnızca monoküler bir kameradan elde edilen bilgileri kullanarak nesnelerin mutlak mesafesini tahmin etmek için YOLO algoritmasını geliştirmiştir. Bu geliştirme, tahmin vektörlerinin genişletilmesi, omurganın ağırlıklarının sınırlayıcı kutu regresörüyle paylaşılması ve orijinal kayıp fonksiyonunun mesafe tahminine yönelik bir bileşenle güncellenmesi gibi yenilikler içermektedir. Sınıf-agnostik bir yaklaşım benimsenerek daha küçük tahmin vektörleri oluşturulmuş ve daha doğru sonuçlar elde edilmiştir. Ayrıca, bu geliştirilmiş yöntemle 45 FPS çıkarım hızı korunmuş ve modifiye edilmemiş YOLO ile aynı hızda çalıştığı gösterilmiştir. Bu sonuçlar, nesne algılama ve mesafe ölçümünün birlikte optimize edilmesinin büyük bir potansiyele sahip olduğunu göstermektedir.

3. ÖZGÜN DEĞER

Kara muharebe araçlarının tespiti, teşhisi ve mesafesinin belirlenmesi, derin öğrenme yöntemleri kullanılarak etkili bir şekilde gerçekleştirilebilir. Derin öğrenme, çok katmanlı yapay sinir ağları aracılığıyla veri örüntülerini öğrenme ve tanımlama kapasitesine sahip bir makine öğrenme yöntemidir. Bu teknik, özellikle kara muharebe araçlarının tespiti ve teşhisi gibi karmaşık görevlerde son derece başarılıdır. Bu tez çalışmasında, kara muharebe araçlarının tespiti ve teşhisi için kullanılan derin öğrenme yöntemleri şu başlıklar altında ele alınmıştır:

3.1. Sinir Ağı Tabanlı Nesne Algılama

Önceden eğitilmiş bir sinir ağı modeli, kara muharebe araçlarının tespiti için etkin bir şekilde kullanılmıştır. Çalışma, özellikle kara muharebe araçlarının tespitine odaklanmıştır. Algılanan hedeflerin doğru bir şekilde sınıflandırılması ve bu hedeflerin konumlarına göre mesafelerinin tahmin edilmesi, geliştirilen sistemin karar destek mekanizmasına katkıda bulunmasını sağlamıştır. Bu yaklaşımla, karar vericilerin operasyonel süreçlerde daha hızlı ve isabetli kararlar alması hedeflenmiştir.

3.2. Veri Seti Hazırlığı

Geliştirilen sistemin, tespit, teşhis, mesafe tahmini ve öneri gibi genelleme yeteneğinin yüksek olması hedeflenmiştir. Bunun için, açık erişimli veri setleri birleştirilmiş, alan uzmanlığı desteği kullanılarak etiketleme yapılmıştır. Bunların yanında yine alan uzmanlığı kullanılarak veri eklemeleri yapılmıştır. Bu sayede büyük ve dengeli bir veri seti oluşturulmuştur. Ayrıca, veri artırma tekniklerinin uygulanmasıyla genelleme performansı yüksek bir veri seti yaratılmıştır.

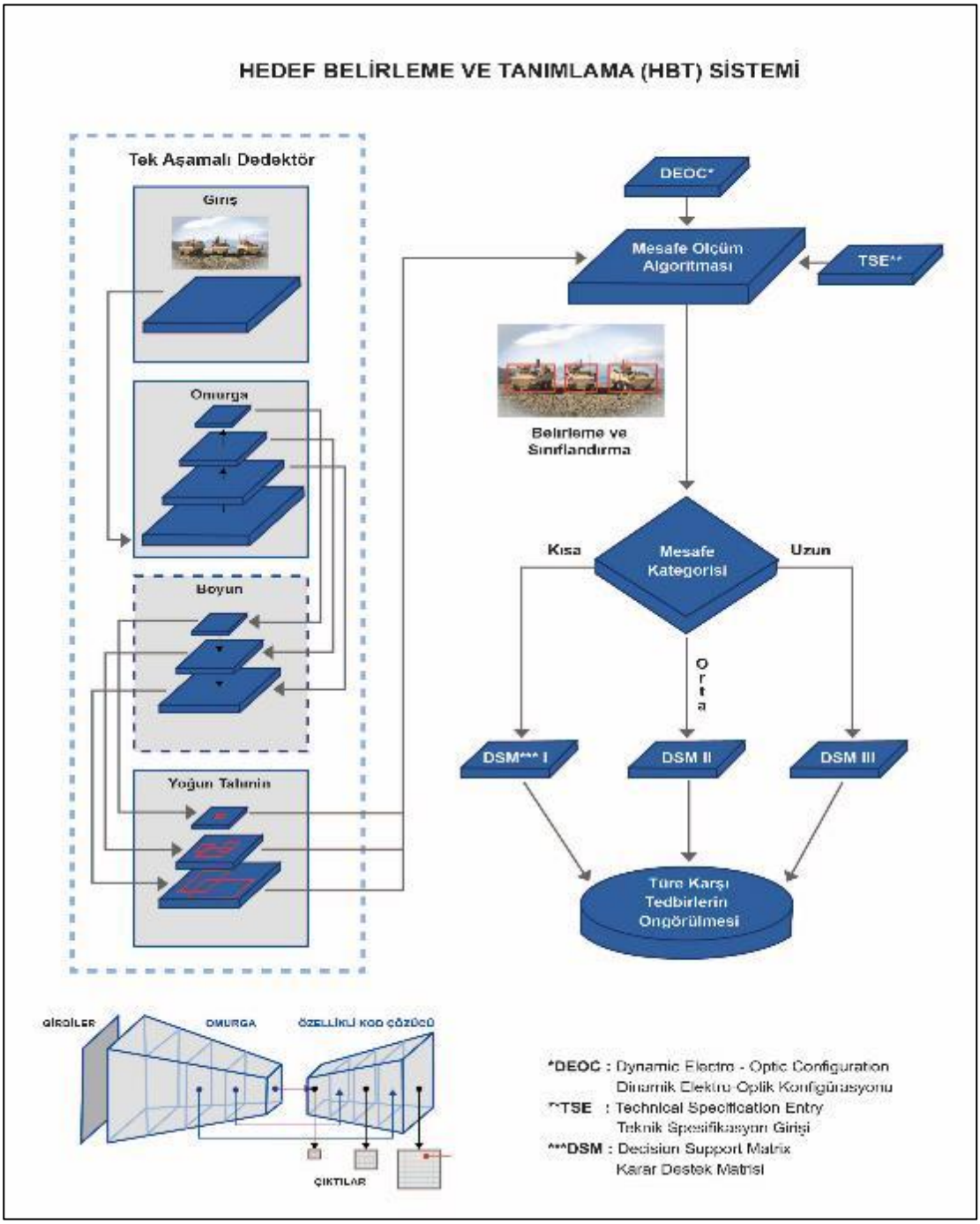
3.3. Model Geliştirme ve Optimizasyonu

Derin öğrenme tabanlı modelin yüksek sınıflandırma başarısına sahip olabilmesi için farklı senaryolar denenerek, en iyi sonucu verecek bir model oluşturulmaya çalışılmıştır. Bu amaçla modelin parametreleri üzerinde optimize edilmiş ve performansı artırılmıştır. Bu maksatla, optimize edici kayıp fonksiyonu ve veri kümesi bileşimi uygun şekilde yapılmaya çalışılmıştır. Optimum performans için ayarlamaların yanında çok miktarda deneme

yapılmıştır. Algoritmanın performansını arttırmak için dikkat mekanizması da eklenmiştir. Bu mekanizmalar, bir modelin hangi özelliklerin veya bilgilerinin daha önemli olduğuna odaklanmasını sağlayarak, genel performansını artırır. Ayrıca, modele pasif mesafe ölçme algoritması eklenerek, tahmin yeteneği performans kaybı olmadan geliştirilmiştir. Elde edilen sonuçlara göre, karar vericiye değerlendirme kritik zamanını azaltan ve karar vermesini kolaylaştıran bir Karar Destek Sistemi ortaya konulmaya çalışılmıştır.

4. YÖNTEMLER VE ANALİZ SONUÇLARI

Bu tez çalışmasında, kara muharebe araçlarının tespiti, teşhisi, mesafe ölçümü ve karşı tedbir önerilerinin geliştirilmesi için kapsamlı bir yöntem (Şekil 4.1) önerilmektedir.



Şekil 4.1. Örnek Yaklaşım

Çalışmanın ana hedefi, askeri operasyonlarda kullanılan araçların hızlı ve doğru bir

şekilde tespit edilmesi, bu bilgilerin analiz edilmesi ve operatif karar alma süreçlerine katkı sağlayacak öneriler geliştirilmesidir. Bu kapsamda, açık erişimli ve özel olarak hazırlanmış veri setlerinden yararlanılarak veri artırma ve ön işleme teknikleri uygulanmış, başarı oranını artıracak model geliştirme adımları tasarlanmıştır.

Çalışma kapsamında, açık erişimli veri setlerinin yanı sıra, askeri gereksinimlere uygun özel veri setleri hazırlanmıştır. Özel veri setleri, kara muharebe araçlarının çeşitli sınıflarını temsil eden görüntülerden oluşturulmuş ve bu veri setlerine veri artırma teknikleri uygulanarak çeşitlilik ve hacim artırılmıştır. Ayrıca, bu görüntülere uygulanan ön işleme ve bölütleme yöntemleri ile öznitelik çıkarımı ve sınıflandırma aşamalarında yüksek doğruluk oranları elde edilmesi hedeflenmiştir. Bu veri setleri ve uygulanan işlemler, sistemin görsel takip ve teşhis süreçlerinde gerçek zamanlı bir performansa ulaşmasını sağlayacak şekilde tasarlanmıştır.

Görsel takip ve teşhis sisteminin hızlı ve gerçek zamanlı bir performans sunması amacıyla, çeşitli şekillendirme ve optimizasyon yöntemleri uygulanmıştır. Hedeflerin hızlı bir şekilde tespit edilmesi için çeşitli algoritmalar denenmiştir. Uyarlama ve denemelerle modelin performansı artırılmıştır. Bu yaklaşımla, analiz süreci sadeleştirilmiş ve hedef tespitine yönelik işlemler daha verimli hale getirilmiştir. Bu adımları desteklemek için, transfer öğrenme yöntemleri ve detaylı sınıflandırıcı ağlar entegre edilmiştir.

Başlangıç aşamasında, temel bir ağ yapısı ve sınıflandırıcı kullanılarak hedeflerin temel bilgileri tanımlanmıştır. Daha sonra, hedeflerin mesafe tahminlerini gerçekleştirmek ve bu bilgilere dayanarak operasyonel karar süreçlerine katkı sağlayacak karşı tedbir önerileri oluşturmak için ek işlemler geliştirilmiştir. Bu süreçte, çerçeveler arası fark algoritması, arka plan modelleme algoritması ve görüntü bölütleme teknikleri etkin bir şekilde kullanılmıştır. Transfer öğrenme yöntemiyle önceden eğitilmiş modeller, mevcut veri setine uyarlanarak sistemin genelleme kabiliyeti artırılmıştır.

Derin öğrenme tabanlı modelin performansını artırmak amacıyla, ağ parametreleri detaylı bir şekilde genetik algoritmalar kullanılarak optimize edilmiştir. Bu optimizasyon süreci, modelin hem yüksek doğruluk oranlarına ulaşmasını hem de işlem yükünü minimize ederek sistem kaynaklarını verimli kullanması sağlanmıştır. Sonuç olarak, hızlı, etkili ve halihazırda kara muharebe araçlarında bulunan mevcut elektro-optik sistemleri zorlamayan bir hedef tespit ve sınıflandırma mekanizması geliştirilmiştir.

Elde edilen bu sistem, askeri operasyonlarda gerçek zamanlı veri işleme ve karar destek mekanizmalarının etkinliğini artıracak şekilde tasarlanmıştır.

4.1. Çalışmada Kullanılacak Veri Setleri ve Analize Uygun Hale Getirilmesi

Askeri hedeflerin tespiti ve tanınmasına yönelik derin öğrenme tabanlı bir sistem, kaliteli ve büyük ölçekli veri setlerine ihtiyaç duymaktadır. Görüntü analizi yapabilmek için kullanılacak veri seti için tam olarak oluşturulmuş ve yayımlanmış bir set bulunmamaktadır. Literatürde askeri hedefler ile ilgili pek çok açık erişimli veri seti bulunmaması da gizlilik mülahazaları dikkate alındığında normal karşılanabilir. Bu doğrultuda tez çalışmasında kullanılması planlanan veri setleri özel çalışma ile oluşturulmuştur. Bu kapsamda;

Veriler, açık kaynaklardaki görüntüler veya videolar gibi farklı biçimlerde toplanmıştır. Veri toplama aşamasında, verilerin yeterli sayıda ve verinin çalışılan kümeyi temsil edecek ettiğinden emin olunmuştur. Verilerin etiketlenmesinde ilgili alan uzmanlarının desteği alınarak, veri setinin eğitim sırasında doğru şekilde kullanılmasına çalışılmıştır. Bu işlem, zaman alıcı ve zahmetlidir, ancak doğru şekilde yapıldığında modelin iyi şekilde eğitilmesini sağlayacağı değerlendirilmektedir. Verilerdeki yanlış veya bozuk verilerin çıkarılması sağlanmakta, bu sayede de veri setinin doğru ve güvenilir olması hedeflenmiştir.

Veri setinin boyutunu arttırmak, modelin daha iyi sonuçlar vermesine yardımcı olmak için, veri arttırma, mevcut verilerin farklı şekillerde ele alınarak yeni veriler oluşturulması yöntemleri uygulanmıştır. Veri arttırma yöntemleri olarak, döndürme, yansıtma, kırpma, renk dönüşümü, gürültü ekleme, kesme ve genişletme yöntemleri uygulanmıştır.

4.2. Uygulanmakta Olan Ön İşleme Adımları

Hedef tanımlama sistemlerinin başarısını belirleyen en kritik adımlardan biri ön işlemedir. Bu süreç, görüntü analiz sistemlerinde, önemli teşhis özelliklerini muhafaza ederek görüntü kalitesini artırmak ve aynı zamanda istenmeyen unsurları ve gürültüyü ortadan kaldırmak amacıyla uygulanır. Ön işleme, sistemin genel performansını doğrudan etkileyen temel bir adımdır.

Görüntü iyileştirme ve gürültü giderme işlemlerine ek olarak, veri büyütme ve normalizasyon gibi adımlar da bu sürecin önemli parçalarını oluşturmaktadır. Ön işleme

aşamasında, görüntü iyileştirme ve gürültü giderme gibi temel tekniklerin yanı sıra, görüntülerin yeniden boyutlandırılması, veri büyütme ve normalizasyon işlemleri gerçekleştirilmiştir. Bu adımlar hem modelin eğitim verimliliğini artırmak hem de farklı çözünürlüklerdeki görüntülerin uyumlu hale getirilmesini sağlamak amacıyla uygulanmıştır. Özellikle veri artırma adımı, derin öğrenme modellerinin aşırı öğrenme eğilimlerini azaltmak ve genel performansını artırmak için kritik bir öneme sahiptir.

Yukarıda anlatılan bilgiler doğrultusunda ön işleme amacıyla aşağıdaki adımlar uygulanmıştır.

Derin öğrenme ağlarında genelleme yeteneğini geliştirmek ve aşırı öğrenmeyi önlemek için büyük ve çeşitli veri kümelerine ihtiyaç duyulmaktadır. Ancak veri toplama ve etiketleme süreçleri, zaman alıcı olmalarının yanı sıra veri gizliliği gibi zorlukları da beraberinde getirir. Bu durumun üstesinden gelmek için, veri artırma teknikleri kullanılarak mevcut veri setinin çeşitlendirilmesi sağlanmıştır. Veri artırma, orijinal veri kümesinden farklı varyasyonlar üreterek modelin genelleme yeteneğini artıran etkili bir yöntemdir. Uygulanan teknikler arasında döndürme, öteleme, ölçekleme ve çevirme gibi işlemler yer almaktadır. Bu yöntemler, modelin eğitim süreci boyunca daha fazla farklılık ve çeşitlilik içeren verilerle karşılaşmasını sağladığı için, modelin genel performansını artırmaktadır.

Deneylerin ilk aşamasında, Roboflow Universe [37] platformu kullanılarak veri setleri hazırlanmıştır. İlk veri seti, toplamda 730 görüntüden oluşmaktadır. Veri büyütme teknikleri ile bu set genişletilmiş; çevirme ve döndürme işlemleri uygulanmıştır. Örneğin, döndürme işlemi -20° ile +20° arasında, sınırlayıcı kutu döndürme ise -15° ile +15° aralığında gerçekleştirilmiştir. Bu adımlar sonucunda, veri seti 2450 görüntüye ulaşmıştır. Sonuçlar incelenmiş ve türler arasında dengenin korunması amacıyla veri sayısı 2400'e düşürülmüştür. Ek olarak, döndürme, kesme ve gri tonlama gibi veri artırma teknikleri ile yeni görüntü çeşitleri (Şekil 4.2) üretilmiş ve veri seti daha da çeşitlendirilmiştir. Uygulanan bu veri artırma teknikleri, modelin genelleme yeteneğini artırarak tespit ve sınıflandırma işlemlerinin başarısını artırmıştır.

Şekil 4.2 Veri Arttırma Teknikleri Uygulanması (Rotate, Shear and Grayscale)

Döndürme metodu (-15° ile +15° arasında), uzatma metodu (-15° ile +15° arasında), uygulayarak ve gri tonlama yaparak (genel setin %25'ine uygulanır), 5984 görüntüden yeni veri seti oluşturulmuştur (Tablo 4.1'deki veri seti bölümüne bakın).

Tablo 4.1. Veri Seti

Eğitim Seti	Geçerleme Seti	Test Seti
%75	%15	%10
4489	897	598

Doğrulama ve test için gerekli görüntüler orijinal veri setinden ayrılmıştır. Bu nedenle yüzde eşdeğerleri düşüktür. Tablo 4.2'de görüleceği üzere, görüntü başına ortalama ek açıklamalar ise görüntü başına 1,2'dir. Ortalama görüntü boyutları 0,41 MP, ortalama görüntü oranları ise 640x640'tır.

Tablo 4.2. Veri Seti Resim Özellikleri

EK Açıklamalar	Ortalama Görsel Büyüklüğü	Medyan Görsel Oranı
5984	0,41 MP	640x640
Şekil Başına 1,2 görsel	-	Kare

4.2.1. Operasyonel Veri Tipleri

Kara muharebe araçlarının sınıflandırılması, ülkelerin ve askeri organizasyonların farklı standartlarına göre değişiklik gösterebilir. Ayrıca, teknolojik gelişmeler, mevcut kategorilerin yeniden tanımlanmasına veya yeni araç kategorilerinin oluşmasına yol açabilir. Bunun yanı sıra, bazı araçlar, yapıları ve görev durumlarına bağlı olarak birden fazla rol üstlenebilmektedir. Bu nedenle, veri setlerinin hazırlanmasında operasyonel tiplerin doğru tanımlanması kritik bir öneme sahiptir. Bu bağlamda, veri seti oluşturulurken belirli göstergeler dikkate alınarak kara muharebe araçlarının türlerine ve görev tanımlarına odaklanılmıştır.

Veri seti türlerini belirlerken hareket kabiliyeti ve askeri koruma sınıfları gibi ana göstergeler dikkate alınmıştır. Kara zırhlı muharebe araçları, hareket kabiliyetlerine göre tekerlekli ve paletli olarak iki ana kategoriye ayrılmıştır. Ayrıca, her araç, askeri yük ve koruma sınıflarına göre ağır, orta ve hafif olarak sınıflandırılmıştır [22]. Keşif faaliyetleri esnasında veya herhangi bir faaliyet esnasında düşmana ait yapılan tespitler için hazırlanan raporlara Gözetleme raporu denilmektedir. Gözetleme raporlarında, tür ve sayı bilgilerinin istihbarat değerlendirmeleri için kritik olduğu belirtilmiştir [23,24]. Bu doğrultuda, muharebe sahasında dost birliklere karşı en etkili olduğu değerlendirilen araçlar, alan uzmanlarının rehberliğinde sınıflandırılmıştır. Bu sınıflandırma, veri seti oluşturma sürecinin temelini oluşturmuş ve çalışma kapsamında odaklanılan araç türleri belirlenmiştir.

Hazırlanan veri seti, kara kuvvetlerinde yaygın olarak kullanılan yedi ana araç türünü içermektedir. Bu araç türleri Tablo 4.3'de sunulmuştur.

Tablo 4.3. Araç Türleri

Sıra No:	Tür (Türkçe)	Tür (İngilizce)	Kısaltma
1	Orta ve Ağır Zırhlı Tekerlekli Taşıyıcılar	Medium/Heavy Armoured Wheeled	MHAW
2	Orta ve Ağır Zırhlı Paletli Taşıyıcılar	Medium/Heavy Armoured Tracked	MHAT
3	Askeri Kamyonlar	Military Trucks	MT
4	Ağır Ana Muharebe Tankları	Heavy Tanks	HT
5	Hafif Zırhlı Tekerlekli Taşıyıcılar	Light Armoured Wheeled	LAW
6	Kundağı Motorlu Zırhlı Topçular	Armoured Artillery	AA
7	Mayına Dayanıklı ve Pusuya Karşı Korumalı Araçlar	Mine Resistant Ambush Protected Vehicles	MRAP

Bu türlerin seçiminde, alan uzmanlarının rehberliğiyle muharebe sahasında karşılaşılan ihtiyaçlar dikkate alınmıştır. Bu araç türlerinin görsel özellikleri, sınıflandırma modelinin eğitim sürecinde temel veri olarak kullanılmıştır.

(a) MHAW (b) MHAT

(c) MT (d) HT

(e) AA (f) LAW

(g) MRAP

Şekil 4.3. Örnek Zırhlı Muharebe Araçları

Veri setinde kullanılan tüm görseller, askeri alan uzmanlarının gözetiminde detaylı bir şekilde örnek resimlerdeki gibi etiketlenmiştir (Şekil 4.3. Örnek Zırhlı Muharebe Araçları). Bu süreç, modelin eğitimi sırasında doğru sınıflandırma yapılmasını sağlamak amacıyla gerçekleştirilmiştir. Ayrıca, test verileri, eğitim veri setinden bağımsız olarak oluşturulmuş ve böylece modelin genelleme yeteneği tarafsız bir şekilde değerlendirilmiştir. Bu şekilde oluşturulan veri seti, sistemin yüksek doğruluk oranlarına ulaşmasını ve muharebe sahasında etkili bir şekilde kullanılmasını sağlayacak sağlam bir temel sunmaktadır.

4.3. Öznitelik Çıkarımı ve Sınıflandırma Çalışmaları

Öznitelik çıkarımı, örüntü tanıma alanında sistemin performansını doğrudan etkileyen kritik bir adımdır. Bu adımda, doğru öznitelik çıkarma tekniklerinin seçilmesi ve sınıflandırıcı algoritmasının bu özellikleri doğru bir şekilde öğrenmesi, bir HBT sisteminin tam kapasiteyle çalışabilmesi için hayati öneme sahiptir. Geleneksel HBT sistemleri, görüntü işleme teknikleriyle elle çıkarılan özniteliklere dayanır. Ancak bu yaklaşımlar, özniteliklerin manuel çıkarılması nedeniyle sübjektif ve sınırlı kalmaktadır. Günümüzde, araştırmacılar geleneksel yöntemlerin yerini, görüntülerden otomatik ve objektif öznitelikler çıkarabilen derin öğrenme tabanlı yöntemlerle değiştirmiştir. Derin öğrenme algoritmaları, ham verileri işleyerek manuel öznitelik çıkarımına gerek duymadan sınıflandırma problemlerinde daha güvenilir sonuçlar elde edilmesini sağlar. Bu bağlamda, derin öğrenme algoritmaları özellikle CNN gibi yöntemler, öznitelik çıkarımı ve sınıflandırma süreçlerinde büyük avantajlar sunmaktadır.

CNN, derin öğrenme ve görüntü analizi alanlarında en yaygın kullanılan sinir ağı türlerinden biridir. CNN, öznitelik çıkarma ve sınıflandırma olmak üzere iki ana katmandan oluşur. Öznitelik çıkarma katmanları, evrişim ve havuzlama işlemleri yoluyla yüksek boyutlu verilerden düşük boyutlu öznitelikler elde eder. Evrişim katmanı, çekirdek (filtre) ve girdi (örneğin bir görüntü) arasındaki evrişim işlemini gerçekleştirerek öznitelik haritaları oluşturur. Bu katmandan çıkan sonuç, filtre boyutu, filtre sayısı, adım ve dolgu gibi parametrelerden etkilenir. Evrişim işleminden sonra uygulanan aktivasyon fonksiyonu, ağı doğrusal olmayan bir yapıya dönüştürerek öğrenme sürecini hızlandırır. Havuzlama katmanı, her bir öznitelik haritasının boyutunu küçültür ve aşırı öğrenmeyi önlerken sonraki işlemler için hesaplama yükünü azaltır. CNN yapısının sonunda yer alan düzleştirme katmanı, öznitelik haritalarını bir boyutlu bir diziye dönüştürerek tam bağlantılı katmanlara aktarır. Tam bağlantılı katmanlar ise, çıkarılan öznitelikleri kullanarak sınıflandırma sonuçlarını üretir. Bu yapı, CNN'in hem öznitelik çıkarımı hem de sınıflandırma süreçlerinde sağladığı avantajlarla modelin genel performansını artırmasını sağlar.

Görüntülerin hedef kategorilere göre sınıflandırılması sürecinde, her bir adımda görüntülerin daha yüksek doğrulukla iyileştirilmesi amaçlanmıştır. Bu çalışmada, önceden eğitilmiş CNN mimarileri doğrudan kullanılmış ve bu mimarilerin mevcut katmanlarında herhangi bir değişiklik yapılmamıştır. Ek olarak, sınırlayıcı katman ve algoritmalar ilave edilerek hedeflere daha hassas şekilde ulaşılmıştır. Analiz sonuçları, yüksek doğruluk

oranlarının elde edildiğini ve modelin validasyon süreçlerinde tutarlı performans gösterdiğini ortaya koymuştur.

Algoritmanın ince ayarlamaları hiperparametreler değiştirilerek yapılmıştır. Hiperparametreler seçilecek algoritma için yüksek seviyeli, yapısal ayarlardır. Eğitim aşamasından önce ayarlanırlar ve eğitim sırasında sabit kalırlar. Eğitim esnasında öğrenme oranı (minimuma doğru hareket ederken her iterasyonda adım boyutunu belirler), parti büyüklüğü (batch: bir ileri geçişte aynı anda işlenen görüntü sayısı), devir/çağ sayısı (epochs: tüm eğitim örneklerinin bir tam ileri ve geri geçişidir) ve tabii ki algoritmanın temel mimari özellikleri (kanal sayıları, katman sayısı, aktivasyon fonksiyonlarının türleri vb.).

Ayrıca, eğitim sürecinde kullanılan ham ve iyileştirilmiş görüntüler üzerinde yapılan analizler, modelin aşırı öğrenmeden (overfitting) kaçınmasını sağlamış ve genelleme kapasitesini artırmıştır.

Bu sonuçlar, modelin hem teorik hem de pratik uygulamalarda güçlü performans sergilediğini ve kara muharebe araçlarının tespiti ve teşhisi için etkili bir sistem olduğunu göstermektedir.

4.4. Temel Algoritma Seçimi

Nesne algılama alanında birçok farklı algoritma bulunmaktadır. R-CNN (Regions with CNN features), görüntüdeki mantıksal nesne bölgelerini tanımlamak için öneri kutucukları oluşturan ilk başarılı yöntemlerden biridir. R-CNN, her bir öneri kutusunu ayrı ayrı işlediği için yüksek doğruluk sağlar; ancak işlemler sırasında yüksek hesaplama maliyetleri ve düşük hız gibi dezavantajları vardır. Bu zorlukları aşmak için daha hızlı sürümleri geliştirilmiştir, örneğin Fast R-CNN ve Faster R-CNN; ancak bu modeller de çoğunlukla karmaşıklıkları nedeniyle gerçek zamanlı uygulamalarda sınırlı kalmıştır.

SSD (Single Shot Multi-Box Detector), nesne tespitini tek bir adıma indirger ve bu sayede hızlı bir şekilde tahmin yapar. SSD, çok ölçekli nesneleri tespit etme yeteneği ile dikkat çekerken, bazen sınırlı doğruluk sunabilmektedir. Benzer şekilde, RetinaNet algoritması, Focal Loss işlevini kullanarak zorlayıcı sınıflara odaklanmayı hedefler ve tahmin doğruluğunu artırır. Ancak, bu algoritmalar da genellikle işlem süresinin uzunluğu veya karmaşıklıklarıyla sınırlıdır.

Tüm bu yöntemler arasında, YOLO algoritması, hızlı işlem süresi ve yüksek doğruluk oranı sayesinde nesne algılama için ideal bir seçimdir. YOLO, görüntüyü tek bir aşamada analiz ederek, gerçek zamanlı nesne tespitinde rakipsiz bir performans sergilemektedir. Ayrıca, mimari yapısının getirdiği kullanım kolaylığı ve geliştirme esnekliği, mevcut projeler için de büyük avantajlar sunmaktadır. Bu nedenle, YOLO algoritmasını tercih edilmiştir.

4.4.1. YOLO Algoritmasının Genel Özellikleri

YOLO algoritması, nesne tespiti için en popüler ve etkili derin öğrenme yaklaşımlarından biridir. Bu algoritma, bir görüntüyü tek bir sinir ağı yardımıyla analiz ederek nesneleri tespit etme ve sınıflandırma işlemlerini aynı anda gerçekleştirir. Bu özelliği sayesinde gerçek zamanlı uygulamalarda kullanımı oldukça yaygındır.

Hız: YOLO, tüm görüntüyü tek bir seferde değerlendirerek diğer yöntemlere kıyasla çok daha hızlı çalışır ve gerçek zamanlı uygulamalar için ideal bir çözüm sunar [25].

Genel Görüş: YOLO, bir görüntüyü parça parça analiz etmek yerine tamamını değerlendirir. Bu yaklaşım, nesnelerin bağlamını ve çevredeki diğer unsurlarla ilişkilerini daha iyi anlamasına olanak tanır [25].

Azalan Yanlış Pozitifler: Algoritma, tüm resmi analiz ederek yanlış pozitif oranını önemli ölçüde azaltır [25].

Esneklik ve Uyarlanabilirlik: YOLO, farklı tespit senaryolarına ve makine öğrenimi görevlerine kolayca adapte edilebilir [26].

Basit Mimari: YOLO'nun mimarisi hem uygulanabilirliği hem de özelleştirme işlemlerini kolaylaştıran basit bir yapıya sahiptir [25].

Bu özellikler doğrultusunda temel algoritma olarak YOLO seçilmiş, farklı versiyonlar denenerek en iyi performansı sağlayan YOLOv8 üzerinden çalışmalar sürdürülmüştür.

4.4.2. YOLOv8'in Gelişmiş Özellikleri

YOLOv8, YOLO algoritmasının en son geliştirilmiş versiyonlarından biridir. Ultralytics firması tarafından tanıtılmıştır. Gelişmiş özellikleri ve verimliliği sayesinde

nesne tespiti uygulamalarında yeni bir standart oluşturmuştur. Bu standartın bazı temel yapı taşları aşağıdaki paragraflarda özetlenmiştir.

Gelişmiş Omurga ve Boyun Mimarileri: Bu yapı, özellik çıkarma işleminde performansı artırır.

Çapasız Bölünmüş Ultralytics Başlığı: Çapa tabanlı yöntemlere kıyasla doğruluğu ve verimliliği artırır.

Optimize Edilmiş Doğruluk-Hız Dengesi: Gerçek zamanlı nesne tespiti görevleri için hem doğruluk hem de hız açısından dengeli bir yapı sunar.

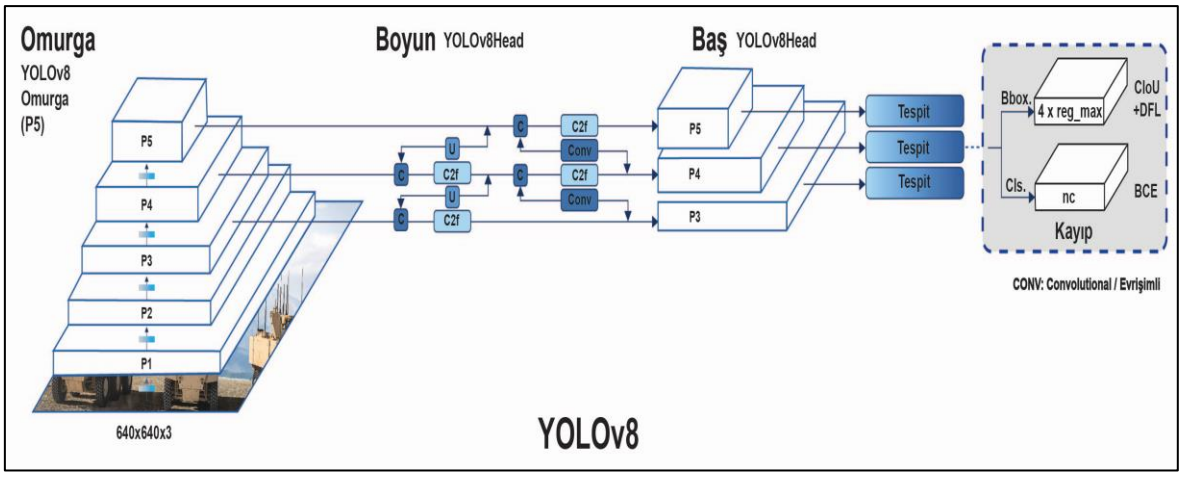
Mozaik Veri Artırma: Eğitim sırasında verilerin çeşitlendirilmesini sağlayarak modelin sağlamlığını artırır.

Ayrışmış Başlık: Sınıflandırma ve regresyon görevlerini ayırarak daha yüksek performans sağlar.

Değiştirilmiş Kayıp Fonksiyonu: Modelin daha iyi eğitilmesini ve doğruluk sağlamasını destekleyen bir kayıp fonksiyonu içerir [38].

4.4.4. Model Yapısı ve Mimari Detaylar

Bu özellikleriyle YOLOv8, bu tez kapsamında kullanılmak üzere en uygun algoritma olarak belirlenmiş ve detaylı bir şekilde incelenmiştir. YOLO mimarisi, nesne algılama ve sınıflandırmada yüksek performans gösteren bir sistemdir ve üç ana bileşenden oluşur: Omurga (Backbone), Boyun (Neck) ve Başlık (Head). Bu bileşenler, veri işlemeyi, özellik çıkarımını ve tespit işlemlerini etkin bir şekilde koordine ederek algoritmanın başarısını sağlar. Her bir bileşen, algoritmanın farklı aşamalarında kritik roller üstlenmektedir. Çalışılmakta olan Model Yapısı detayları Şekil 4.4'te verilmiştir[40].



Şekil 4.4. YOLOv8 Mimarisi

Omurga (Backbone), YOLO algoritmasının temel bileşenlerinden biridir ve görüntüyü işleyerek özellik haritaları oluşturur. Genellikle ResNet veya Darknet gibi önceden eğitilmiş derin öğrenme modelleri bu bölümde kullanılır. Omurganın iki ana bileşeni bulunmaktadır: Konvolüsyonel Katmanlar: Görüntüdeki örüntüleri ve özellikleri çıkararak temel bilgi sağlar. Havuzlama Katmanları: Görüntünün boyutunu küçültüp hesaplama yükünü azaltarak algoritmanın daha verimli çalışmasına olanak tanır. Omurgadan elde edilen bu özellik haritaları, boyun (Neck) bileşeninde daha ileri seviyede işlenir.

Boyun (Neck), omurgadan gelen özellik haritalarını birleştirerek tespit bloklarına aktarılmasını sağlar. Bu bileşen, özellikle tespit doğruluğunu artırmak için önemlidir. Boynun temel işlevleri şu şekilde özetlenebilir: Yükseltme Katmanı: Özellik haritalarının çözünürlüğünü artırarak detaylı bilgi sağlar. C2f Blokları: Optimize edilmiş kısayol bağlantıları ve derinlik katmanlarıyla tespit işlemlerini iyileştirir. Boyunda işlenen bu veriler, başlık (Head) bileşeni aracılığıyla nihai tespit ve sınıflandırma işlemlerine yönlendirilir.

Başlık (Head), farklı boyutlardaki nesnelerin algılanmasını sağlayan özel bir bölümdür. Bu bileşen, tespit bloklarından gelen verileri işleyerek sınırlayıcı kutuların yerlerini ve sınıf tahminlerini gerçekleştirir: C2f Blokları: Nesne algılama doğruluğunu artıran optimize edilmiş konvolüsyonel bloklar. Konvolüsyonel Blokları: Nesnelerin sınıf olasılıklarını ve konumlarını yüksek doğrulukla tahmin eder. YOLO'nun bu mimari bileşenleri, nesne algılama ve sınıflandırma süreçlerini etkili bir şekilde bir araya getirir.

YOLOv8, modelin aşırı uyumu genelleme ve azaltma yeteneğini geliştirmek için

çeşitli veri büyütme teknikleri kullanır. Bu teknikler şunları içerir: Mozaik Büyütme ve Kopyala-Yapıştır Büyütme. Bu özellikler YOLOv5 algoritmasından taşınan özelliklerdir [41].

YOLOv8, modelin performansını artırmak için çeşitli karmaşık eğitim stratejileri uygular. Bunlar şunları içerir: Çok Ölçekli Eğitim, Otoanşör (AutoAnchor), Isınma ve Kosinüs LR Zamanlayıcısı (Cosine LR Scheduler), Üstel Hareketli Ortalama (EMA), Karma Hassasiyet Eğitimi, Hiperparametre Evrimi.

YOLOv8 önceden eğitilmiş modeller sunmaktadır. Model genel performans (Tablo 4.4) bilgileri halihazırda algoritmanın ortalama doğruluk ve uygulama hızları açısından değerlendirildiğinde başarı seviyelerinin iyi olduğunu göstermektedir [37].

Tablo 4.4. YOLOv8 Performans Bilgileri

Model	**Boyut**	**Ortalama Hassasiyet (mAP)**	**Hız**
YOLOv8n	640	37,3	80,4
YOLOv8s	640	44,9	128,4
YOLOv8m	640	50,2	234,7
YOLOv8l	640	52,9	375,2
YOLOv8x	640	53,9	479,1

YOLOv8I ve YOLOv8x gibi daha büyük modeller neredeyse her zaman daha iyi sonuçlar verecektir, ancak daha fazla parametre içerirler, eğitmek için daha fazla CUDA (Compute Unified Architecture) belleğine ihtiyaç duyarlar ve daha yavaş çalışırlar [39].

YOLO algoritmalarında kayıp üç bölümden oluşur:

Sınıf Kaybı [Classes Loss (BCE Loss)]: İkili Çapraz Entropi kaybı, modelin her sınıf için tahmin ettiği olasılıkları gerçek etiketlerle karşılaştırarak, sınıflandırma görevindeki hatayı ölçer.

Nesnellik Kaybı [Objectness Loss (BCE Loss)]: Başka bir ikili Çapraz Entropi kaybı, belirli bir ızgara hücresinde bir nesne olup olmadığını tespit etmedeki hatayı ölçer. Model, her ızgara hücresi için bir nesne bulunup bulunmadığını tahmin eder.

Konum Kaybı [(Location Loss (CIoU Loss)]: onum kaybı, tahmin edilen kutuların gerçek nesne kutularıyla ne kadar örtüştüğünü ölçerek, nesneyi ızgara hücresinde doğru bir şekilde yerleştirmedeki hatayı değerlendirir.

Bu kayıpların toplamı aşağıdaki gibi hesaplanır (4.1):

$$Loss = \lambda 1 Lcls + \lambda 2 Lobj + \lambda 3 Lloc \quad (4.1)$$

Loss : Toplam kayıp
$\lambda 1 Lcls$: Sınıf kaybı
$\lambda 2 Lobj$: Nesnellik kaybı
$\lambda 3 Lloc$: Konum kaybı
λi : Kayıpların ağırlıklandırma (dengeleme katsayısı) parametreleridir.

Üç tahmin katmanının (P3, P4, P5) nesnellik kayıpları farklı ağırlıklandırılmıştır. Denge ağırlıkları sırasıyla [4.0, 1.0, 0.4] 'dür. Üç tahmin katmanından gelen nesnellik kayıpları, aşağıdaki gibi farklı ağırlıklarla birleştirilir (4.2):

$$Lobj = 4.0 \cdot Lobj\ small + 1.0 \cdot Lobj\ medium + 0.4 \cdot Lobj\ large \quad (4.2)$$

$4.0 \cdot Lobj\ \ small$: Küçük nesneler için nesnellik kaybı
$1.0 \cdot Lobj\ \ medium$: Orta boy nesneler için nesnellik kaybı
$0.4 \cdot Lobj\ large$: Büyük nesneler için nesnellik kaybı

YOLO için kutu koordinatlarını tahmin etme formülü, ızgara hassasiyetini azaltmak ve modelin sınırsız kutu boyutlarını tahmin etmesini önlemek için güncellenmiştir. Aşağıdaki denklemler, tahmin edilen kutu koordinatlarının nasıl hesaplandığını gösterir (4.3, 6). Bu denklemlerde görünen terimlerden “bx, by, bw, bh” tahmin edilen sınırlayıcı kutunun merkezi, genişliği ve yüksekliğidir. “tx, ty, tw ve th”, sinir ağlarının çıktılarıdır. “cx ve cy”, hücrenin bağlantı kutusunun sol üst köşesidir. “pw ve ph”, bağlantı kutusunun genişliği ve yüksekliğidir.

X Koordinatı Tahmini:

$$bx = (2 \cdot \sigma(tx) - 0.5) + Cx \quad (4.3)$$

Y Koordinatı Tahmini:

$$by = (2 \cdot \sigma(ty) - 0.5) + Cy \quad (4.4)$$

Genişlik Tahmini:

$$bw = pw \cdot (2 \cdot \sigma(tw))\ 2 \quad (4.5)$$

Yükseklik Tahmini:

$$bh = ph \cdot (2 \cdot \sigma(th))\ 2 \quad (4.6)$$

σ : Sigmoid fonksiyon

YOLO'da, hedef oluşturma süreci nesne tespitinde büyük bir role sahiptir ve modelin eğitim verimliliği ile doğruluğunu artırmak için kritik öneme sahiptir. Bu süreç, çıktı haritasındaki uygun ızgara hücrelerine temel doğruluk kutularını atamak için bir dizi adım izler. Bu adımlar, zemin doğruluk kutuları ile bağlantı kutuları arasında doğru eşleşmeler sağlamayı amaçlar. Bu işlem denklem (4.7) – (4.12) arasındaki adımları ı takip eder:

İlk olarak, zemin doğruluk kutusu boyutlarının ve her bir bağlantı şablonunun boyutlarının oranları hesaplanır. Zemin doğruluk kutusunun genişlik ve yükseklik değerleri (4-7) numaralı fonksiyon ile hesaplanır.

$$rw = wgt/wat \quad (4.7)$$

Bağlantı şablonunun (ankraj kutusu) genişlik ve yükseklik değerleri (4.8) numaralı fonksiyon ile hesaplanır.

$$rh = hgt/hat \quad (4.8)$$

Oranların maksimize edilmesi (4.9 ve 4.10) numaralı fonksiyonlar ile yapılır. Bu adım, oranların simetrik olarak değerlendirilmesi için gereklidir.

$$rw\ max = \max(rw, 1\ rw) \quad (4.9)$$

$$rh\ max = \max(rh, 1\ rh) \quad (4.10)$$

Oranların birleştirilmesi (4.11) numaralı fonksiyonlar ile yapılır. Maksimum oranların birleştirilmesiyle nihai oran hesaplanır.

$$r\ max = \max\ (rw\ max, rh\ max) \tag{4.11}$$

Doğruluk kutusunun eşleştirilmesi (4.12) numaralı fonksiyon ile gerçekleştirilir. Eğer hesaplanan oran belirli bir eşikle karşılaştırıldığında eşit veya küçükse, zemin doğruluk kutusu uygun bağlantı kutusuyla eşleştirilir

$$r\ max = anchort \tag{4.12}$$

Burada, "wgt, hgt, wat ve hat" sırasıyla zemin gerçeği kutularının ve ankraj şablonlarının genişliğini ve yüksekliğini temsil eder. Bu nedenle, genişliği ve yüksekliği zemin gerçeği kutusunun genişliğinin ve yüksekliğinin her biri 4 katından fazla ve 0,25'inden az değilse bir ankraj şablonu zemin gerçeği kutusuyla eşleşir.

Gözden geçirilmiş merkez noktası ofseti nedeniyle, bir zemin doğruluk kutusu birden fazla ankraja atanabilir. Bu nedenle, eşleşen ankrajları uygun ızgara hücrelerine atamak önemlidir. Merkez nokta uzaklık aralıkları, (0, 1) ile (-0.5, 1.5) arasında ayarlandığı için bu durum mümkündür. Bu sayede, bir zemin doğruluk kutusu birden fazla bağlantı kutusuna atanabilir.

Bu şekilde, hedef oluşturma süreci, eğitim sırasında her bir zemin doğruluk kutusunun uygun bir şekilde belirlenen ankrajlarla eşleştirilmesini sağlar. Sonuç olarak, bu işlem algoritmanın nesne algılama görevini daha etkili bir şekilde öğrenmesine katkıda bulunur.

4.4.3. Görüntü İşleme Çalışmalarındaki Uygulama Avantajları

YOLOv8, geniş bir uygulama yelpazesinde kullanılmak üzere tasarlanmıştır ve aşağıdaki avantajları sunar:

Gerçek Zamanlı Algılama: YOLO, görüntülerdeki veya video karelerindeki nesneleri neredeyse gerçek zamanlı olarak tespit edebilir, bu da onu gözetleme, otonom araçlar ve benzeri uygulamalar için ideal hale getirir.

Eğitim Verileri: Çok çeşitli nesne sınıflarını içeren COCO (Common Objects in Context) gibi büyük veri kümeleri üzerinde eğitilen YOLOv8, özel projelerdeki veri

setleriyle de kolayca entegre edilebilir.

Performans: YOLOv8 modelleri, kıyaslama veri kümelerinde en iyi sonuçlara ulaşmıştır. Örneğin, YOLOv8n modeli, COCO veri kümesinde 37,3 mAP (ortalama hassasiyet) ve 0,99 ms hızla çalışabilmektedir [39].

4.4.5. Dikkat Mekanizmaları ve YOLOv8 Sıkıştırma ve Uyarım Modülü (Squeeze and Excitation-SE) Uygulaması

4.4.5.1. Dikkat Mekanizmaları

Dikkat mekanizmaları, derin öğrenme ve bilgisayarla görme alanlarında önemli bir yer tutmaktadır. Bu mekanizmalar, bir modelin hangi özelliklerin veya bilgilerinin daha önemli olduğuna odaklanmasını sağlayarak, genel performansını artırır. Dikkat mekanizmaları uygulamalarında, sıkıştırma ve uyarım (SE) modülü, konvolüsyonel blok dikkat modülü (CBAM), yerel olmayan ağlar (NLN) ve transformerlardaki dikkat mekanizması (AMT) gibi metotları kullanmaktadır.

SE Modülü; her bir kanalın önemini öğrenmek için iki aşamalı bir süreç kullanır: "squeeze" ve "excitation". Bu modül, belirli bir kanalın önemli olup olmadığını belirleyerek, kanal özelliklerine yeniden ölçeklendirme yapar. Yalnızca kanal seviyesinde dikkat sağlar, bu sayede modelin karmaşıklığını artırmadan performansı iyileştirir. Düşük ek hesaplama maliyeti ile yüksek performans sunar [30].

CBAM Modülü hem uzaysal hem de kanal dikkati mekanizmalarını bir arada kullanarak, daha ayrıntılı dikkat sağlar. Bu modül, her bir özelliğin hem konumuna hem de önemine odaklanır. Hem kanal hem de uzaysal dikkat sağladığı için, daha karmaşık nesne tespit senaryolarında yüksek performans sunar. Görüntü işleme uygulamalarında, modelin genel doğruluğunu artırabilir [42].

NLN; her bir pikselin, diğer piksellerin dikkatini toplamak için kullanılır. Bu sayede uzun mesafeli bağıntıları modelleyerek nesnelerin daha doğru bir şekilde tespit edilmesini sağlar. Görüntü içindeki nesneler arasındaki bağıntıları dikkate alarak genel bağlamı anlamaya yardımcı olur. Özellikle karmaşık sahnelerde detaylı temsil sağlar [43].

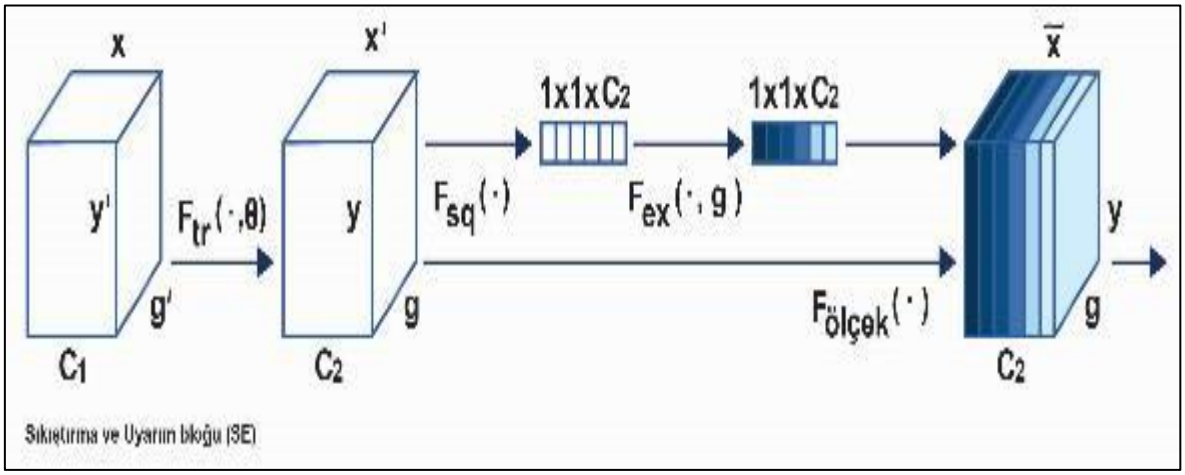
AMT; Transformer mimarisi, dikkat mekanizmalarını modelin belirli kısımlarına

odaklanması için kullanır. Bu yapı, konvolüsyonel ağlarla karşılaştırıldığında daha uzun bağıntıları daha etkili bir şekilde öğrenebilir. Esnekliği ve ölçeklenebilirliği ile dikkat çekici sonuçlar elde edilir. Çoklu bilgi akışlarıyla daha zengin temsiller oluşturur [44].

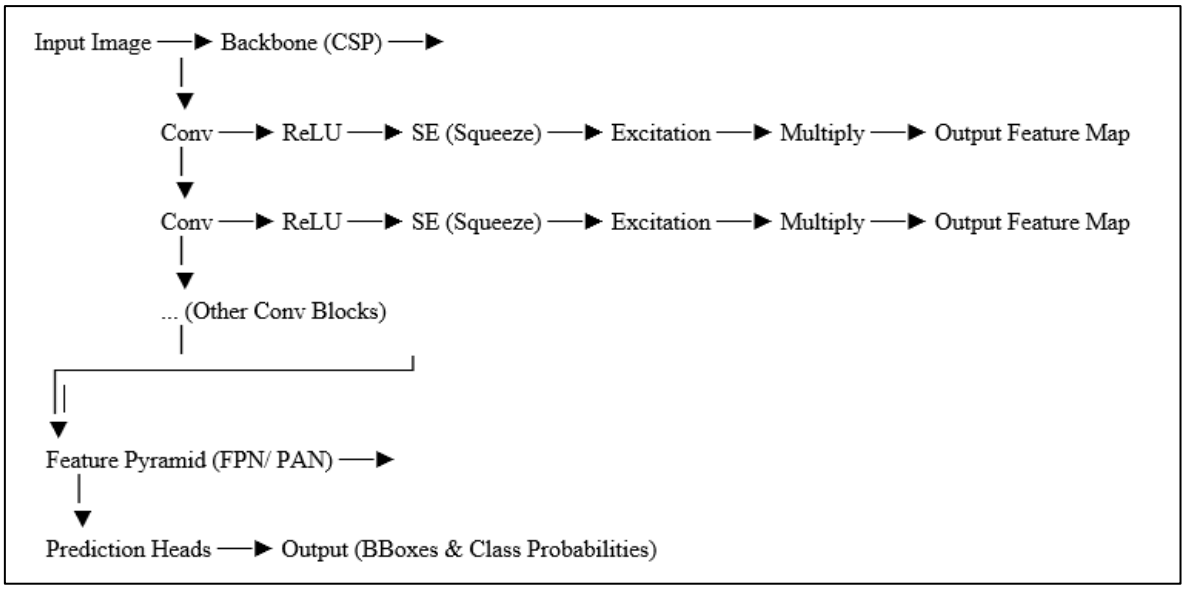
Eğer daha hafif bir yapıya ve daha hızlı bir hesaplamaya ihtiyaç duyuluyor ise, özellikle sınırlı kaynaklı ortamlarda, SE Modülü tercih edilebilir. Bu, genellikle daha az hesaplama gücü gerektirir ve daha hızlı çalışır.

4.4.5.2. Sıkıştırma ve Uyarım Modülü

SE modülü, derin öğrenme ve özellikle konvolüsyonel sinir ağları (CNN) alanında, kanal dikkatini artırmak ve modelin temsili gücünü iyileştirmek için geliştirilmiş bir yapıdır. SE modülü, özellikle nesne tanımada ve görüntü sınıflandırmada kullanılır. Diğer dikkat mekanizmalarına göre Şekil 4.5'de görüleceği üzere daha basit bir yapıya sahiptir. Bu tasarım çalışmamız kapsamında ana mimarimize Şekil 4.6'da belirtildiği üzere her konvolüsyon (Conv) bloğuna eklenmiştir. Bu sayede ağ, daha kritik kanalları ön plana çıkartır, gereksiz kanalların katkısını düşürür ve daha iyi ayırt edici özellikler edinir. Askeri araçlar gibi birbirine benzeyen alt türlerde, SE katmanının modelin başarısını olumlu etkileyeceği değerlendirilmiştir.



Şekil 4.5. Sıkıştırma ve Uyarım Modülü Mimarisi



Şekil 4.6. YOLOv8 Algoritmasında SE Modülünün Eklenmesi

Kanal Dikkati: SE modülü, her bir kanalın önemini öğrenir ve bu kanallara ağırlık verecek şekilde yeniden ölçeklenmesini sağlar. Bu süreç, modelin hangi özelliklerin önemli olduğunu anlamasına ve daha etkili bir temsil oluşturmasına yardımcı olur.

SE Modülünün Yapısı, üç aşamadan oluşur:

Squeeze (Sıkıştırma) Aşaması: Giriş özellik haritası yani yükseklik, genişlik ve kanal sayısı olan bir tensördür (4.13).

$$F_{tr} \ : X - X', \ X \in R \ y' \times g' \times C1 \qquad U \in R \, y \times g \times C2$$

$$X' = v_c * X = \sum_{s=1}^{C1} V_c^s * x^s \tag{4.13}$$

İlk olarak, her bir kanal için global ortalama havuzlama işlemi uygulanarak sıkıştırılır. Bu aşama, her kanalın genel özelliklerini temsil eden bir "özellik özeti" oluşturur (4.14). Burada (z_c), ilgili kanala ait sıkıştırılmış değeri ifade eder.

$$z_c = F_{sq}(X'_c) = \frac{1}{YxG} \sum_{i=1}^{Y} \sum_{j=1}^{G} X'_c \ (i,j) \tag{4.14}$$

Excitation (Uyarım) Aşaması: Sıkıştırılan özellik vektörü iki tam bağlı (dense) katman aracılığıyla yeniden ölçeklendirilir. İlk katmanda ReLU aktivasyon fonksiyonu, ikinci katmanda ise Sigmoid aktivasyon fonksiyonu kullanılır. Bu aşamada, her bir kanalın önem derecesini gösteren ağırlıklar hesaplanır (4.15).

$$s = F_{ex}(.,G) = \sigma(g(.,G) = \sigma\big(G_2\delta(G_1.)\big) \quad (4.15)$$

Burada (G_1) ve (G_2) ağırlık matrisleridir ve (σ) sigmoid fonksiyonunu temsil eder.

Skalasyon (Yeniden Ölçeklendirme) Aşaması: Hesaplanan kanal önem ağırlıkları, orijinal özellik haritası ile çarpılarak yeniden ölçeklendirilir. Bu işlem, her bir kanalı kendi önem değerine göre ayarlamaktadır (4.16).

$$\tilde{X}_c = F_{\text{ölçek}}(X'_c, S_c) = S_c \,.\, X'_c \quad (4.16)$$

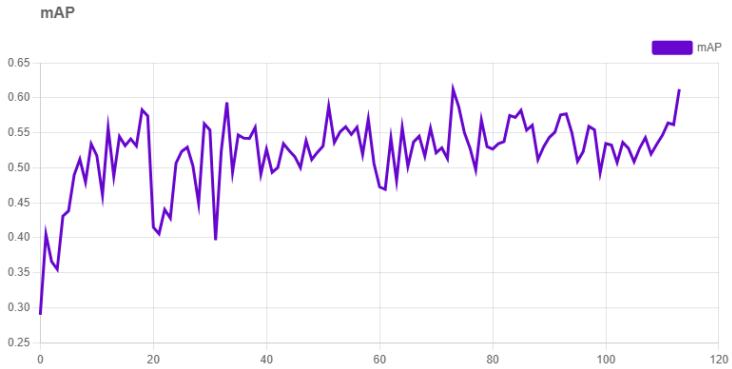
SE Modülü, kanal bazında vurgulamalar yaparak, modelin daha iyi ve anlamlı özellikler öğrenmesini sağlar. Ekstra hesaplama yükü olmaksızın modelin performansını artırır. Genellikle, sınırlı kaynaklara sahip sistemlerde bile iyi çalışır. SE modülünün kullanımı sayesinde ağların genelleme yeteneği artırılır, bu da halihazırda çalışılan zor veri kümelerinde daha iyi performans sağlayacağı öngörülmüştür.

Squeeze-and-Excitation (SE) modülü, derin öğrenme mimarilerinde kullanılabilecek etkili bir dikkat mekanizmasıdır. Özellikle konvolüsyonel sinir ağlarındaki kanal dikkatini artırarak modelin performansını önemli ölçüde iyileştirmektedir. SE modülünün uygulama alanları geniştir ve birçok modern derin öğrenme modelinde önemli bir bileşen olarak yer almaktadır.

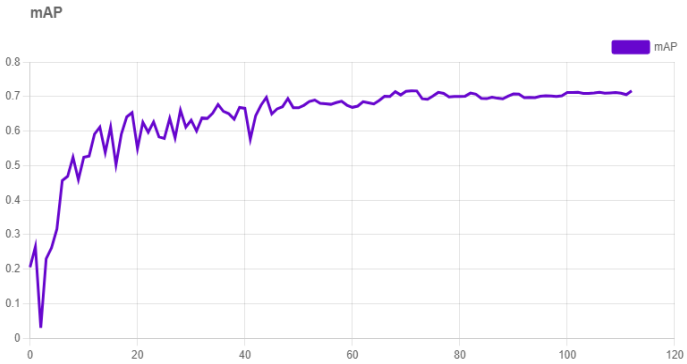
4.5. Hedef Tespit ve Teşhis Çalışmaları

4.5.1. Deneme sonuçları ve değerlendirmeler

Hedef tespiti ve teşhisi için YOLO V8M modeli, eğitim ve test süreçlerinde kullanılmıştır. Eğitim sırasında, veri kümesinin yollarını ve eğitilecek sınıf sayısını tanımlamak amacıyla bir YAML dosyası yapılandırılmıştır. Model, parti büyüklüğü 32'dir. Algoritma 241 dönem boyunca eğitilmiş ve süreç boyunca veriler, "Roboflow" platformu üzerinden gerçek zamanlı olarak görselleştirilip izlenmiştir. Hedef tespit ve teşhisi için %70 ve üzeri olasılıkla yapılan tahminler doğru olarak kabul edilmiştir.

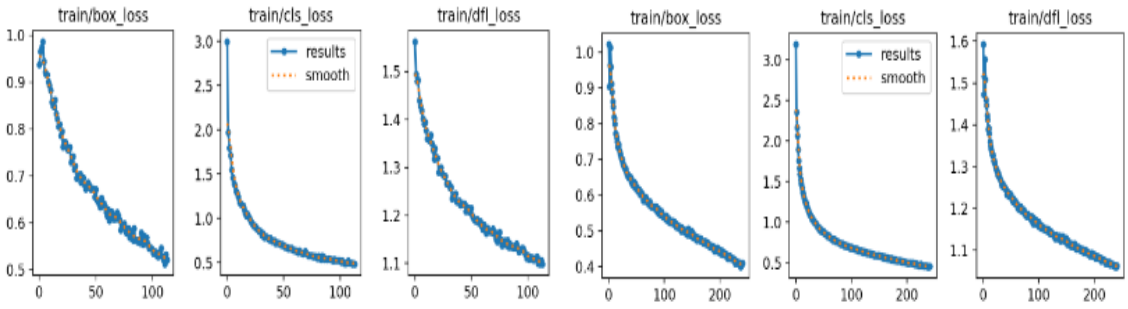


Şekil 4.7. İlk Veri Kümesi Sonuçları

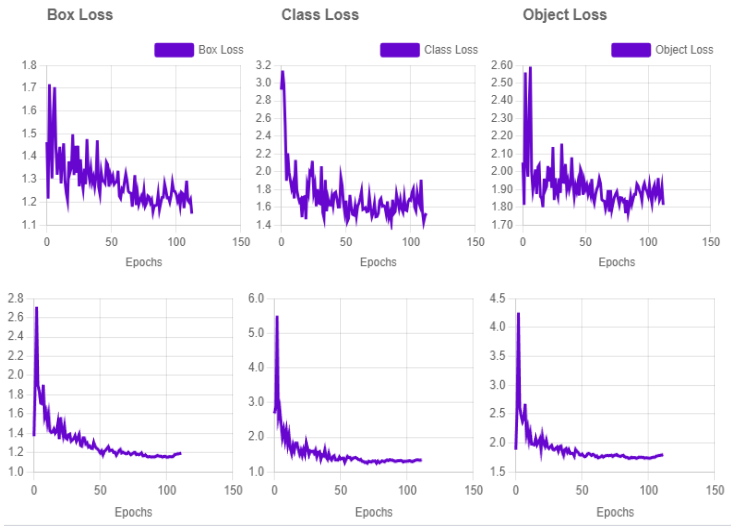


Şekil 4.8. İyileştirilmiş Veri Kümesi Sonuçları

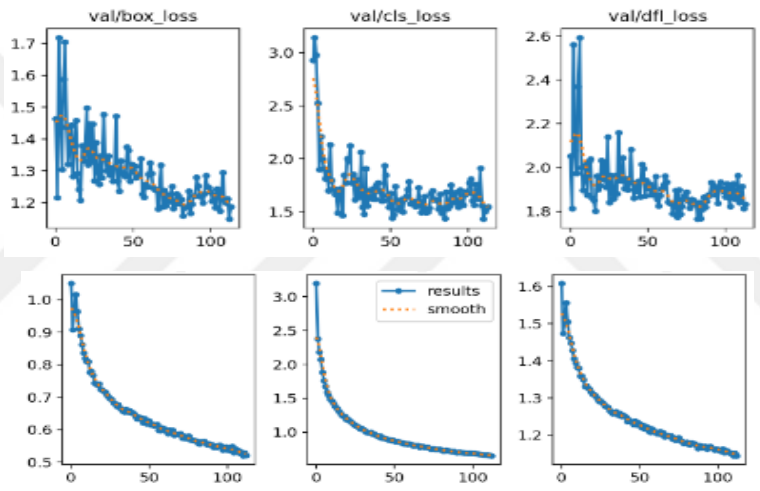
Bu süreçte, farklı veri (Şekil 4.7 İlk Veri Kümesi ve Şekil 4.8 İkinci Veri Kümesi) kümeleriyle yapılan karşılaştırmalar sonuçların doğruluğunu ve modelin performansını değerlendirmek açısından kritik öneme sahiptir. Sonuçların Şekil 4.9, Şekil 4.10 ve Şekil 4.11'e bakılarak karşılaştırılması sonucunda, veri kümesi hazırlığının model performansı üzerindeki belirleyici etkisini açıkça ortaya koymaktadır.



Şekil 4.9. Başlangıç ve İyileştirilmiş Veri Seti Karşılaştırması (Train-Box/Cls/Dfl Loss)



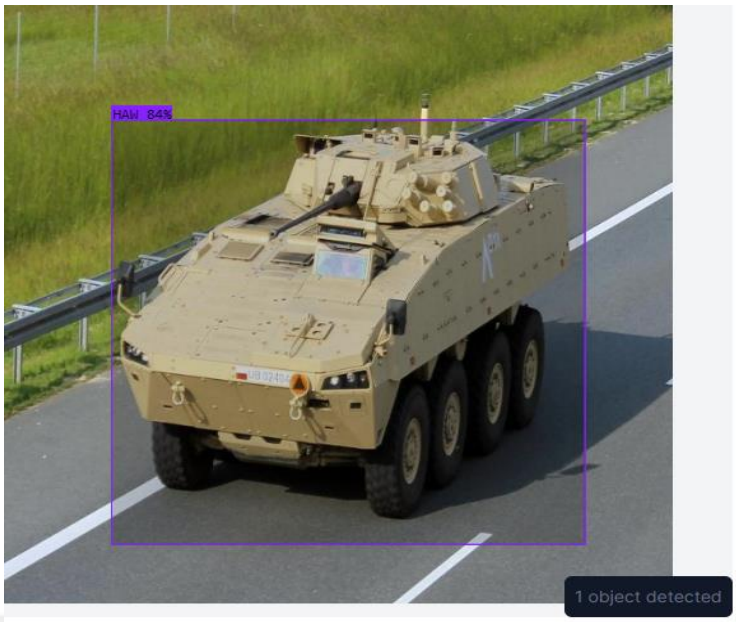
Şekil 4.10. Başlangıç ve İyileştirilmiş Veri Seti Karşılaştırması (Box-Class and Objects Loss)



Şekil 4.11. Başlangıç ve İyileştirilmiş Veri Seti Doğrulaması (Box/Cls/Dfl Loss)

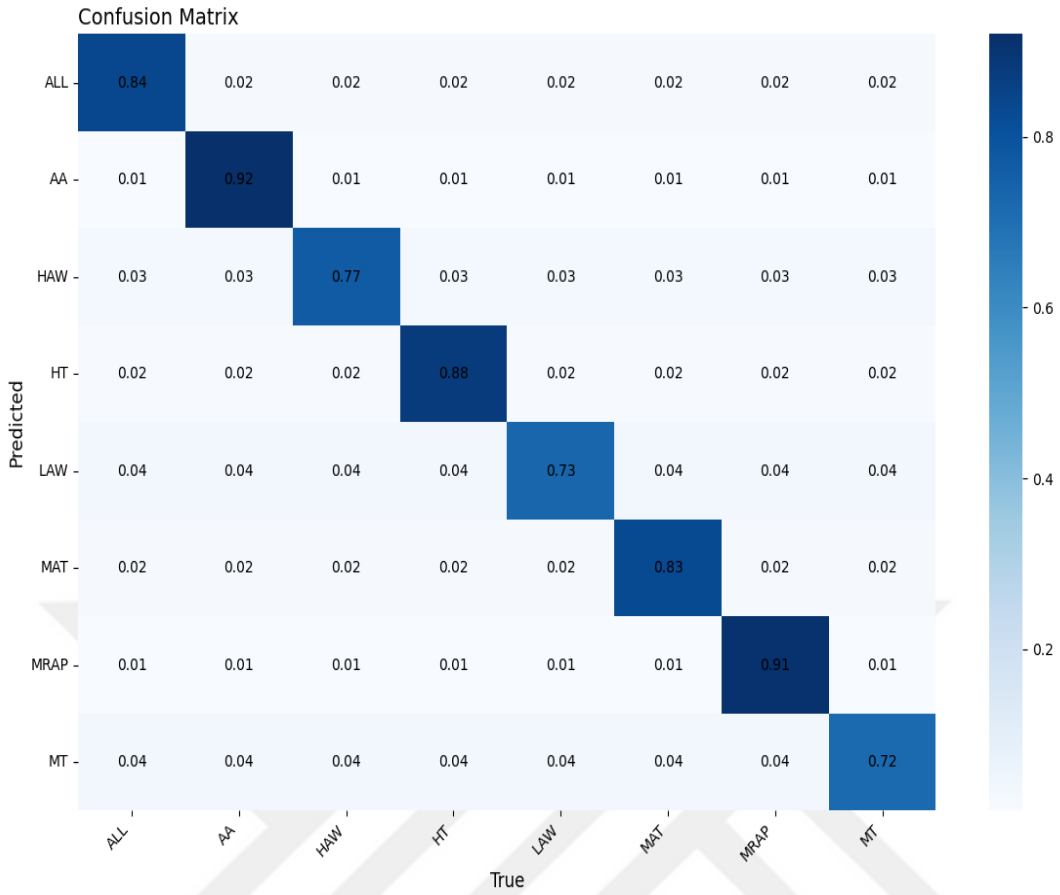
İki veri kümesinin aynı algoritma yapısı içinde incelenmesi kapsamında, kayıp fonksiyonlarında meydana gelen iyileşmeler fonksiyonların daha stabil hale gelmesinden anlaşılmaktadır. Bu bulgular, eğitim döngü sayısının artırılması ve veri kümesi kalitesinin iyileştirilmesinin kayıpları azaltmada etkili olduğunu göstermektedir. Eğitim döngü sayısının artmasıyla birlikte, modeldeki kayıp oranlarının azaldığı gözlemlenmiştir. Sınırlayıcı kutu sınıfı ve nesne gerilemesi bağlamında daha düşük kayıp değerleri, tahmin edilen sınırlayıcı kutuların temel gerçek değerlere yaklaştığını göstermektedir.

Ayrıca, veri kümesinin kalitesinin ve çeşitliliğinin eğitim süreci üzerindeki etkisi de belirgin bir şekilde görülmektedir. Eğitim sonuçları, kullanılan veri setlerinin önemini ve model performansını optimize etmenin yollarını vurgulamaktadır.



Şekil 4.12. Değerlendirme Kutusu ve Sınıf Tahmini

Ortaya çıkan model hedef aracı değerlendirme kutusu içine almakta ve sınıf tahminini Şekil 4.12'te görüleceği üzere başarı ile gerçekleştirmektedir. Oluşturulan hata matrisi (Şekil 4.13) sınıflandırma problemlerinde modelin performansını değerlendirmek için kullanılmıştır. Bu matris, modelin hangi sınıfları doğru tahmin edip edemediğini ve hangi sınıflarda hatalar yaptığını bize göstermektedir. Hata matrisine göre en düşük tespit oranları MHAW ve LAW tipindeki araçlarda gerçekleşmektedir.



Şekil 4.13. YOLO V8 Hata Matrisi

Şekil 4.14. MT, MHAW ve MRAP Örnekleri

MT, MHAW ve MRAP tipindeki (Şekil 4.14) araçlar karşılaştırıldığında, temel farkların araçların boyutları, yerden yükseklikleri, toplam yükseklikleri ve tekerlek sayıları olduğu tespit edilmiştir. Bu farklılıklar, araç tiplerini doğru sınıflandırmada dikkat edilmesi

gereken temel unsurlar olarak öne çıkmaktadır. Çalışma kapsamında çok farklı tipte örnekler seçilmiştir.

Şekil 4.15. MHAT ve MT

Şekil 4.15'de gösterilen MHAT ve MT'yi karşılaştırdığımızda, üst yapılarının hemen hemen aynı olduğu açıkça görülmektedir. Özellikle aynı ülkeler veya firmalar tarafından geliştirilen ürünlerde bu göze çarpmaktadır. Örneğin, Nissan Qashqai ve X-trail modelleri (Şekil 4.16) yapısal olarak birbirlerine çok benzemektedir. Aynı şekilde, bazı tür MHAT'ların HT'lere benzeyen kuleleri vardır. Genellikle tank kulelerinin çapı, MHAT ve kulelerinin çapından daha büyüktür.

Şekil 4.16. Nissan Qashqai ve X-trail modelleri

4.5.2. Operasyonel deneme ve sonuçları

Test veri setleriyle eğitilen algoritma, Panasonic Toughbook CF33 gibi askeri tipte bir bilgisayara yüklenmiştir. Yüklenen algoritma, tablet kamerası ve araç kamerası kullanılarak muharebe araçlarının tespiti ve sınıflandırılması performansını test etmek amacıyla kullanılmıştır. Bu bilgisayarın 1.6GHz hızı, 16GB belleği ve 8MP otofokus arka kamerası bulunmaktadır. Bu bilgisayar ve üzerindeki elektro optik ile yapılan denemelerde sistemin tespit süresinin 0,8 sn.yi aşmadığı görülmektedir. Bu süre operasyonel bakış açısına göre

son derece hızlı kabul edilmektedir. Bu test, algoritmanın operasyonel ortamda gerçek nesnelerle çalışma kapasitesini değerlendirmek için önemli bir adım olarak değerlendirilmiştir.

Yurtdışında bulunan atış ve tatbikat eğitim alanında 10 adet operasyonel MHAW aracı ile arazi testleri gerçekleştirilmiştir. Tespit başarısı %100, birincil sınıflandırma kesinliği ise %80, yanlış teşhis oranı ise %20'dir. Yanlış sınıflandırmalar 1 (bir) LAW ve 1 (bir) MRAP tipindedir. İkincil sınıflandırma oranlarında MHAW en yüksek oranda çıkmaktadır. Bu sonuçlar, algoritmanın tespit yeteneklerinin güçlü olduğunu, ancak sınıflandırma performansında iyileştirmeye ihtiyaç duyduğunu göstermektedir. Bu nedenle 4.5.4 bölümünde belirtileceği üzere ilave iyileştirme çalışmaları yapılmıştır.

Gerçek denemelerde mesafe hataları ilk ölçümlerde %3 olarak belirlenmiş, sisteme eklenen odak mesafe kalibrasyonu yapılmasına müteakiben de hata oranı %1'in altına düşmüştür. Ateş destek vasıtaları ile yapılan atışlarda silah sisteminin özelliğine göre değişse de +- 50 m.ye kadar olan mesafe tahmin hataları normal sınırda kabul edilmektedir [23].

Ayrıca her sınıfa ait araçların basılı resimlerini (her sınıfa ait 10 adet) hazırlayıp tablet kamerasıyla test ettik. Operasyonel test sonuçları aşağıdaki tabloda görüleceği üzere eğitim sırasında elde edilen performansa %5 yakınlık (negatif) göstermiştir. Ancak, sınıflandırma başarısı eğitim sonuçlarına kıyasla

'deki sonuçlar ile karşılaştırıldığında yaklaşık %12 daha düşük bulunmuştur. Basılı resim kullanılması ve kullanılan elektro-optik kameranın performansının düşük olmasının bu sonuçları ortaya çıkardığı değerlendirilmektedir.

Tablo 4.5. Operasyonel Test Sonuçları

Sınıf/D&Y	**LAW**	**MHAW**	**MHAT**	**MT**	**HT**	**AA**	**MRAP**
Doğru	7	8	7	6	8	6	8
Yanlış	3	2	3	4	2	4	2

Operasyonel deneme ve çalışmalarda ortaya çıkacak ikinci bir problem sahası da araçlara eklenebilecek kamuflaj uygulamalarıdır. Kamuflaj uygulamaları sonrasında araçlarda herhangi bir şekil veya görüntü değişikliğinin de teşhis yeteneğini olumsuz yönde azaltacağı değerlendirilmektedir. Bu durumda askeri araçların tespiti için yine elektro optik kameraların termal yeteneklerinin kullanılabileceği değerlendirilmiştir. Ancak teşhis doğruluğu için araçların termal görüntülerine yönelik görüntü çalışmaları yapılmalı ve algoritma tekrar eğitilmelidir.

4.5.3. Farklı YOLO versiyonlarında yapılan denemeler ve sonuçları

Olorunshola ve arkadaşları [27], YOLOv5 ve YOLOv7 modelleri arasında geniş çaplı bir karşılaştırmalı analiz gerçekleştirmiştir. Deney ve test sonuçları, YOLOv5 modelinin YOLOv7'ye göre daha etkili olduğunu ortaya koymuştur.

Bu tez çalışması kapsamında, YOLOv8m'in performans sonuçları, diğer YOLO versiyonları ile karşılaştırılmıştır. Karşılaştırma sonuçları EK-1'de detaylı olarak verilmiştir. Görüleceği üzere YOLOv8'in diğer versiyonlara kıyasla sağladığı %83,6'lık ortalama kesinlik yüzdesi ve 117 FPS'lik tespit hızı, doğruluk ve hız açısından belirgin üstünlükler sağladığı ortaya konulmuştur.

Yapılan deneme ve testlerde YOLO v4, v5, v7, v10 ve v11 algoritmaları, hazırlanan veri kümesi kullanılarak test edilmiştir. YOLOv8'in, YOLOv4, v5, v7, v10 ve v11 ile aynı veri seti ve koşullar altında karşılaştırılması, modelin tutarlı bir biçimde daha iyi sonuçlar verdiğini ve operasyonel etkinlik sağladığını göstermektedir.

YOLOv8'in mimari avantajları, gelişmiş omurga ve boyun yapıları, mozaik veri artırma gibi özellikleri hem tahmin doğruluğunu hem de işlem hızını artırmakta etkili olmuştur.

Verilerle desteklenmiş performans analizleri (5984 görüntüyü içeren veri setinde elde edilen sonuçlar) ile YOLOv8'in üstün performansı, genel sonuçların tutarlılık ve tekrarlanabilirliğini müteakip tabloda göstermektedir. Karşılaştırılan modellere ait detay açıklamalar, şekil ve tablolar EK 1'dedir.

Tablo 4.6. YOLOV8 Algoritmasının Diğer YOLO Serileri Algoritmaları ile Karşılaştırılması

Sınıf	Görüntü	Ortalama Doğruluk Oranı					
		YOLO4m	YOLO5m	YOLO7m	YOLO8m	YOLO10m	YOLO11m
7	5984	60,8%	62,3%	70,7%	83,6%	68,7%	70,5%
Tespit Hızı		76 FPS	79 FPS	84 FPS	117 FPS	80 FPS	100 FPS

4.5.4. YOLOv8 SE uygulama sonuçları

YOLOv8 algoritmasının performansını çalışılan hedef doğrultusunda arttırmak için SE modülü eklenmiştir.

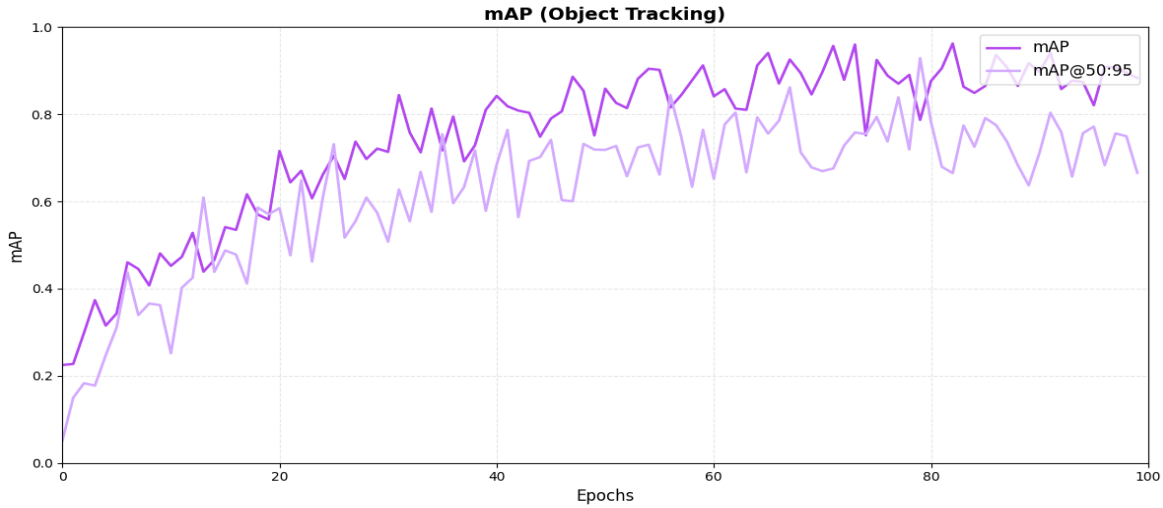
Eğitim Süreci esnasında proje, Google Colab GPU (NVIDIA A100) üzerinde çalıştırılmıştır. Tablo 4.7'de YOLOv8 SE Eğitim Parametreleri sunulmuştur.

Tablo 4.7. YOLOv8 SE Eğitim Parametreleri

Eğitim Parametreleri	
Epoch	50
Batch Size	64
Görüntü Boyutu	640x640
Optimizer	AdamW (lr=0.01, weight_decay=0.0005)
Augmentations	Mosaic=1.0, flip, randaugment vb.

Her konvolüsyon bloğu sonuna eklenen Squeeze-and-Excitation (SE) katmanları ve yapılan ince ayarlamalar ile oluşturulan YOLOv8 SE'nin mevcut veri kümesi ile yapılan eğitimler sonucunda Şekil 4.17'de görülebileceği üzere %88,38 üzerinde ortalama doğruluk

oranına ve %93 kesinlik değerlerine ulaştığı tespit edilmiştir.



Şekil 4.17. Çalışma Veri Kümesinin YOLOv8 SE Algoritması Sonuçları

Kesinlik ve Geri Çağırma kriterleri açısından hata oranlarında iyi sonuçlar elde edilmiştir. Kesinlik, tüm pozitif tahminler arasında doğru pozitiflerin oranını ölçerek modelin yanlış pozitiflerden kaçınma kabiliyetini değerlendirir. Öte yandan, Geri Çağırma, tüm gerçek pozitifler arasındaki doğru pozitiflerin oranını hesaplar ve modelin bir sınıfın tüm örneklerini tespit etme yeteneğini ölçer. YOLOv8 SE sonuçlarına bakıldığında kesinliğin %0.9301'e, geri çağırma/hatırlama oranının ise %0.7980'e ulaştığı görülmektedir.

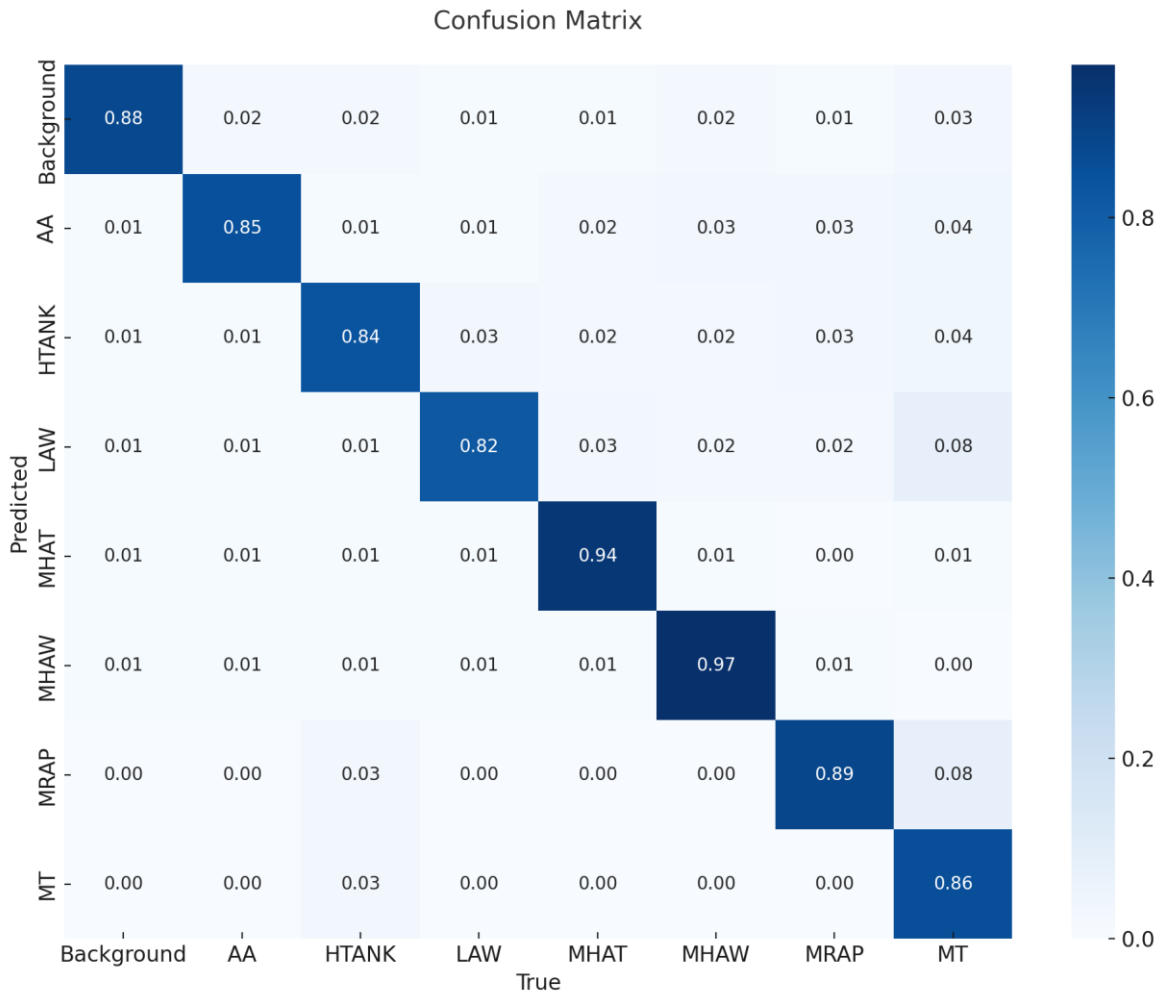
YOLOv8 SE'nin uygulama sonuçlarına dair genel bir değerlendirme ile birlikte, klasik YOLOv8 sonuçları ile karşılaştırması Tablo 4.8'de yapılmıştır. YOLOv8 SE'nin, YOLOv8'e göre ortalama doğruluk oranı %5,7 artmıştır. Bu, özellikle küçük nesneler ve zorlu sahneler üzerinde önemli bir iyileşme sağlamaktadır.

Çalışmamızda, SE modülleri ek bir maliyet yaratmamıştır. YOLOv8 SE'nin hızı, klasik YOLOv8'e göre yakın olmakla beraber, %17'lik hız artışı da yapılan ince ayarlar sonunda elde edilmiştir.

Tablo 4.8. YOLO v8 ve YOLO v8 SE Algoritmalarının Veri Setimize Göre Karşılaştırılması

Sınıf	**Görüntü**	**Ortalama Doğruluk Oranı**		
		YOLO8m	**YOLO8m SE**	**Değişim**
Hepsi	**5984**	83,6%	88,38%	+%5,7
Tespit Hızı		117 FPS	137 FPS	+%17

YOLOv8 SE’nin, daha iyi kanal dikkat mekanizmaları sağlayarak düşük arka planda hatalarla daha doğru sonuçlar ürettiği, aşağıdaki hata matrisinde (Şekil 4.18) görülmektedir.



Şekil 4.18. YOLOv8 SE Hata Matrisi

Ayrıca, YOLOv8'in tür bazında en düşük sonuçlara sahip olan “LAW, MHAW ve MT” serisi araçların, YOLOv8 SE kapsamında yapılan geliştirme sonucunda yapılan karşılaştırmalarında sırasıyla %19, %20 ve %14’lük pozitif yönde performans değerlerinin arttığı görülmektedir.

YOLOv8 SE’ye ait yazılım metodolojisi ve detayları EK 2’dedir.

4.6. Muharebe Araçlarının Bulunduğumuz Yere Göre Mesafelerinin Tespiti

Tespit ve teşhis edilen bir hedefin bulunduğu yere olan mesafenin doğru bir şekilde tahmin edilmesi, askeri karar verme süreçleri ve alınması gereken tedbirler açısından kritik bir öneme sahiptir. Bu süreç, askeri operasyonların etkinliğini artırırken, stratejik ve taktiksel kararların doğruluğunu da güçlendirmektedir.

Geleneksel sistemlerde mesafe tahminleri, genellikle araçlarda bulunan optik dürbünler kullanılarak yapılmaktadır. Bu yöntem, dürbün üzerindeki ölçeklerin kullanılmasıyla gerçekleştirilen basit bir hesaplama sürecine dayanmaktadır ve çoğu zaman tatmin edici doğruluk oranları sağlamaktadır. Bu yöntemlerin temel matematiksel prensipleri, askeri kullanımda kolaylık ve pratiklik sağlamaktadır. Hesaplama işleminin temel prensibi, trigonometrideki benzer üçgenlerin birbirlerine oranlarının eşit olmasından kaynaklanmaktadır.

Doğru mesafe tahminleri, askeri operasyonların etkinliğini artırmanın yanı sıra komutanların stratejik karar verme süreçlerini de desteklemektedir. Bu nedenle, mesafe tahminlerinin güvenilir ve doğru bir şekilde yapılması, taktik seviyede başarı için hayati bir öneme sahiptir. Bu süreçte kullanılan matematiksel hesaplama yöntemleri, doğru mesafe ölçümleri için temel teşkil etmektedir.

Standart bir tank 8,5 metre uzunluğunda ve 2,72 metre yüksekliğindedir. Bir gözetleyicinin; dürbün taksimatı üzerinde okuduğu uzunluk 10 yükseklik 3 milyem ise, hedefin mesafesi denklem (4.17) ile hesaplanmaktadır.

$$\text{Hedefin Mesafesi} = \frac{\text{Hedefin Bilinen Yüksekliği veya Uzunluğu}}{\text{Hedefin Milyem Skalası Olarak Derecesi}} \tag{4.17}$$

1000 katsayısı bir çemberin 6400 milyem olması ve 1000 m mesafede 1 milyem açı değerine denk gelen çember yayının 1 m.ye eşit olmasından kaynaklanmaktadır.

Bir görüntü üzerinde hedefin lokalize edildiği sınırlayıcı kutular (Bounding Box) yardımıyla mesafe tahmini yapılabilmektedir. Sınırlayıcı kutular, x0 ve y0 koordinatları ile genişlik ve yükseklik değerlerini içerir. Bu değerler, nesnenin kameraya olan uzaklığını hesaplamak için kullanılır. Burada x0, y0 sınırlayıcı kutuyu döşemek veya ayarlamak için kullanılır. Genişlik ve Yükseklik değişkenleri ise, nesneyi ölçme ve tespit edilen

nesne/nesnelerin detayını açıklamakta kullanılır. Genişlik ve Yükseklik, nesnenin kameraya olan uzaklığına bağlı olarak değişecektir. Sınırlayıcı kutulardan elde edilen bu parametreler, nesnenin pozisyonunu ve mesafesini değerlendirmek için temel veri sağlar.

Eğer elektro optik sistemde dört temel değişken bulunmaktadır. Bu değişkenler

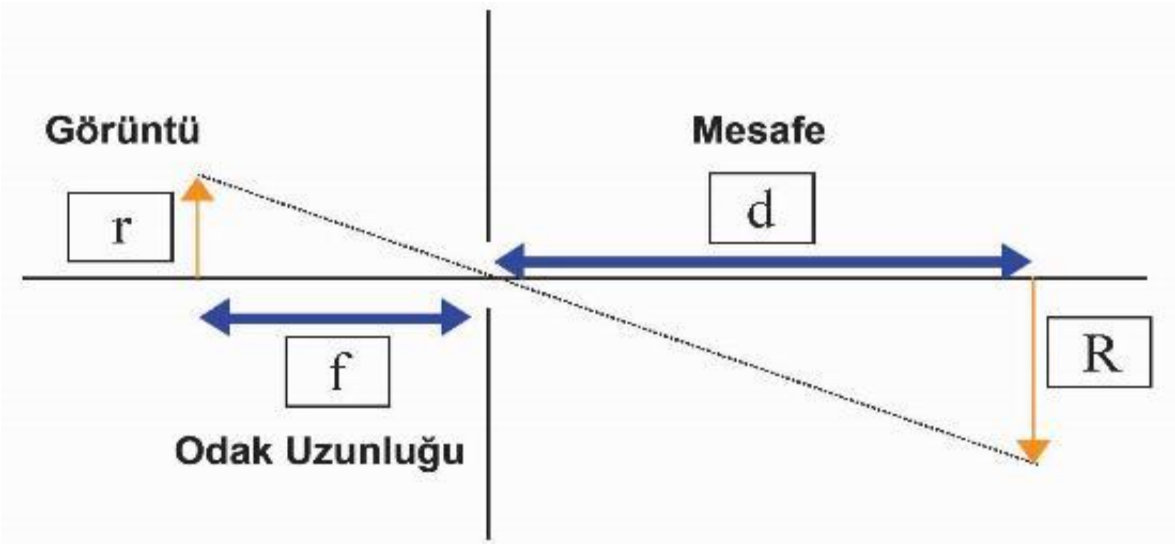
d : Nesnenin mercekten uzaklığı

f : Algılanan görüntünün odak uzaklığı veya odak mesafesi

r : Kırılan görüntünün dışbükey mercekten uzaklığı

R : Görüntünün yüksekliği/genişliği olarak sıralanabilir.

Bu değişkenler aşağıdaki Şekil 4.19'da gösterilen benzer üçgen modelinde gösterilmektedir.



Şekil 4.19. Benzer Üçgen Prensibi

Benzer üçgen modeli kullanılarak oluşturulan bağıntılar (4.18,19,20) kolaylıkla yazılabilir.

$$\frac{f}{d} = \frac{r}{R} \tag{4.18}$$

$$f = d \times \frac{r}{R} \text{ pixels} \tag{4.19}$$

$$d = f \times \frac{R}{r}\ cm \tag{4.20}$$

Şekil 3'de de görüleceği üzere, şeklin sağ tarafında zıt açıların eşit olduğu ve her iki üçgenin bir açısının dik açılı (90°) olduğu iki üçgeni karşılaştırdığımızda denklem (4.21) kolaylıkla yazılabilir.

$$\frac{\text{Açısal Büyüklük}}{360} = \frac{\text{Gerçek Büyüklük}}{2nD} \tag{4.21}$$

Bu hesaplama yöntemi, mevcut YOLO algoritmasına entegre edilmek suretiyle EK-2'de detayları verilen "tespit edilen hedefin türüne göre mesafe tahmini" yapılmasını sağlanabilmektedir. Burada, en çok kullanılan araç tiplerinin en ve boy matrisleri oluşturulmuştur. Araçların boyu kullanıldığında, duruş pozisyonlarından kaynaklı yanlış tahmin oranı artmakta; ancak yükseklik genellikle değişmemektedir. Aşağıdaki Şekil 4.20'de görüleceği üzere bakış açısına göre genişlik algısı ve ölçümü de değişecektir. Bu nedenle, araç tipleri için ortalama yükseklik değerleri alınmıştır. Eğer bu değerler, kullanılacak bölgelere göre revize edilirse, tahmin hata oranının da azalacağı öngörülmektedir.

Şekil 4.20. Bakış Açısına Göre Algının Değişmesi

Yapılan ilk hesaplamalarda, tespit edilen sınırlayıcı kutuların hedefin belli bir oranda dışından oluşturulduğu gözlemlenmiştir. Bu durum, mesafe tahmini üzerinde bir düzeltme katsayısının kullanılmasını gerektirmektedir. Tespit ve teşhis edilen muharebe araçlarında bu oran, yaklaşık olarak %12'ye denk gelmektedir.

Bu yöntem, askeri karar süreçlerinde daha kesin ve güvenilir sonuçlar elde edilmesine katkı sunmaktadır. Dolayısıyla, mesafe tahminine yönelik bu düzeltmeler, operasyonel stratejilerin geliştirilmesine önemli bir destek sunmaktadır.

4.7. Muharebe Araçlarına Karşı Alınacak Tedbir Önerileri

Muharebe araçlarının tespit ve teşhisi sonrasında mesafelerinin doğru bir şekilde tahmin edilmesi, karar vericilerin uygun tedbir almasını veya hedefi etkisiz hale getirecek silah sistemini seçmesini mümkün kılar. Silah sistemleri genel olarak hedefin tipi, korunma derecesi (zırh tipi) ve mesafesine göre "Kısa, Orta ve Uzun Menzilli" veya "Hafif, Orta ve Ağır Sınıf" kategorilerine ayrılmaktadır [22]. Bu çalışmada, hedeflerin mesafesine bağlı olarak "Kısa", "Orta" ve "Uzak" tanımlamaları yapılmıştır. Bu sınıflandırmalar doğrultusunda, hedeflere yönelik karşı tedbirlerin belirlenmesi için bir karar matrisi geliştirilmiştir.

Teşhis edilen muharebe aracı tipi ve bu araçlarda bulunması muhtemel zırh tipine göre kullanılabilecek silah sistemleri ve mühimmat türlerini belirlemek amacıyla bir "Karar Matrisi" geliştirilmiştir. Bu matris, karar seçenekleri ve değerlendirme kıstaslarının belirlenmesiyle oluşturulan "n x m" boyutunda bir yapıdadır ve karar vericilere sistematik bir rehberlik sunar. Karar matrisi, hedefin mesafesi ve korunma durumuna göre en uygun tedbiri otomatik olarak önermektedir.

Müteakip tabloda ve detaylı olarak EK 4'te açıklandığı üzere karşı tedbir önerileri için karar matrisleri matrisi oluşturulmaktadır. Geliştirilen Karar Matrisi, teşhis edilen aracı tipi ve muhtemel zırh türüne göre hangi silah sistemlerinin ve mühimmat türlerinin uygun olduğunu belirlemekte sistematik bir rehberlik sunar. Tablo, hedef mesafesi ve korunma durumuna göre en uygun öneriyi otomatik olarak sağlar ve karar alma süreçlerini yapılandırılmış bir biçimde destekler.

Bu yaklaşım, karar verme süreçlerini sistematik bir şekilde destekleyerek muharebe alanında daha etkili ve güvenilir sonuçların elde edilmesini sağlar. Aynı zamanda, askeri stratejilerin geliştirilmesi için sağlam bir temel oluşturur.

Tablo 4.9. Karşı Tedbir Önerisi için Karar Matrisi (Kısa Mesafe)

Muharebe Araç Türleri	**Minimum Mühimmat/Silah Sistemi Türü**
Kısa Mesafe	**Tavsiye**
Hafif Zırhlı Tekerlekli (LAW) Araçlar	Orta Zırh Delici Mühimmat, RPG
Orta ve Ağır Zırhlı Tekerlekli (MHAW) Araçlar	Kısa Mesafe Silah Sistemi ile A/T Mühimmat

Orta ve Ağır Zırhlı Paletli (MHAT) Araçlar	Kısa Mesafe Silah Sistemi ile A/T Mühimmat
Askeri Kamyonlar (MT)	Kısa Mesafe Silah Sistemi ile A/T Mühimmat
Ağır (Ana Savaş) Tanklar (HT)	RPG
Kundağı Motorlu Topçu (AA)	RPG
Mayın Dayanıklı Pusuya Karşı Korumalı Araçları (MRAP)	RPG

Hedef Belirleme ve Tespit Sisteminin uygulanması, muharebe sahasında karar alma süreçlerine önemli bir katkı sunmaktadır. HBT sisteminin uygulamada, tespit ve teşhis bilgisini "Bu bir ağır sınıf taktik tekerlekli zırhlı araçtır" olarak belirlemesi ve kaynak tablolar yardımıyla hedefin mesafesini "Hedefin mesafesi yaklaşık 1200 m'dir" şeklinde tahmin etmesi sağlanmıştır. Bu bilgiler, muharebe sahasındaki nesneleri tanıyabilen ve uygun karşı tedbir önerileri oluşturabilen bir sistemin geliştirilmesine olanak tanımaktadır. Bu tür bir sistem, askeri operasyonlarda hem etkinlik hem de güvenlik açısından büyük avantajlar sağlamaktadır.

Bu noktada, mesafenin tahmininin aktif sensör sistemleri (örneğin lazer mesafe ölçme sensörleri vb.) kullanılarak yapılmasında hata oranlarını azaltacağı son derece açıktır. Bununla beraber, aktif bir sensör kullanılması, kendi konumumuzun da açığa çıkmasına yol açarak, karşı kuvvet tarafından yerimizin tespiti edilmesine sebebiyet verebilecektir. Bu durum istenmeyen bir zafiyettir.

Pasif mesafe ölçüm sistemleri kullanarak gizliliği korumak ve kendi konumunu açığa çıkarmadan işlem yapmak mümkün olur. Bu tür bir sistem, yalnızca karşı tedbir almaya karar verildiği anda aktif hale getirilerek fark edilme riskini asgari seviyelere indirir. Böylece, sistemin askeri operasyonlara entegre edilmesinin hem etkinliği artıracağı hem de operasyonel esnekliğe katkı sağlayacağı değerlendirilmektedir.

Bu yaklaşım, yalnızca askeri stratejilerin geliştirilmesine katkıda bulunmakla kalmaz, aynı zamanda muharebe sahasında gizlilik ve etkili müdahale kapasitesini artırır. Dolayısıyla, bu sistemin operasyonel süreçlere entegre edilmesi, etkinliği önemli ölçüde artırabilecek bir yenilik olarak değerlendirilmektedir.

YOLOv8 algoritması hızlı ve doğru tespit kabiliyeti sayesinde, belirlenen parametreler ve karar matrisi ile askeri operasyonlarda stratejik bir araç olarak güvenle kullanılabilir. Bu tür yenilikçi yaklaşımlar, sahada karar alma süreçlerini optimize edebilir ve askeri etkinliği büyük ölçüde artırabilir. Ana sınıflandırmayı takiben, ilgili tablolardan elde edilen ek veriler kullanılarak mesafe tahmini ve uygun karar destek matrislerinin kullanımı, sonuçların sunumu ile milisaniyeler içinde hesaplanmaktadır. HBT sistemi kullanılarak elde edilen tekli ve çoklu hedef tespit sonuçları örnek olarak Şekil 4.21 ve Şekil 4.22'de gösterilmektedir.

Tür	: HMAT (80,35%)
Tespit Edilen Yükseklik(h)	: 490,35px
Gerçek Yükseklik (R)	: 2,50m
Odak Uzunluğu (F)	: 14718,75 px
Mesafe	: 74,98 metre
Orta Zırh Delici Mühimmat, RPG	

Şekil 4.21. Örnek Tekli Hedef Uygulama Sonucu

Tür	: LAW (92,896%)
Tespit Edilen Yükseklik(h)	: 411,84 px
Gerçek Yükseklik (R)	: 2,10m
Odak Uzunluğu (F)	: 14718,75 px
Mesafe	:75,05 metre
Orta Zırh Delici Mühimmat, RPG	
Tür	: LAW (77,89%)
Tespit Edilen Yükseklik (h)	: 83,87 px
Gerçek Yükseklik (R)	: 2,10m
Odak Uzunluğu (F)	: 14718,75 px
Mesafe	:368,54 metre
Orta Zırh Delici Mühimmat, RPG	

Şekil 4.22. Örnek Çoklu Hedef Uygulama Sonucu

4.8. Genel Değerlendirme ve Sistemin Başarısı

Önceki bölümlerde sistemin her bir basamağı için değişik metriklerde ve farklı bakış açılarıyla değerlendirmeler ve analizler yapılmıştır. Bununla beraber, operasyonel bakış açısı ile sistemin genel başarısı değerlendirilmek istendiğinde her basamağın başarısı bir önceki basmağa dayanmaktadır. Bu nedenle en önemli basamak hedefin tespiti ve teşhisi safhasıdır. Hedefin türü doğru tespit edilemediğinde mesafe ölçümü için gerekli yükseklik verileri de yanlış olacaktır. Bu durumda hedefin mesafesi de yanlış olarak tespit edilecek, müteakiben de yanlış karar destek matrisi kullanılmasından ötürü HBT sistemi yanlış öneri oluşturulacaktır.

5. GELİŞTİRİLEN HEDEF BELİRLEME VE TANIMLAMA (HBT) SİSTEMİ

5.1. Sistemin Genel Açıklaması

Hedef Belirleme ve Tanımlama (HBT) sistemi, elektro-optik sistemler aracılığıyla elde edilen görüntülerden hedeflerin tespit ve tanımlanmasını desteklemek amacıyla geliştirilmiştir. Sistem, hedef yönetim süreçlerinde kritik bir öneme sahip olan hedef tespit ve tanımlama faaliyetlerine odaklanmaktadır. Bu sistem, derin öğrenme tabanlı mimariler ve modern algoritmalar kullanılarak tasarlanmıştır.

HBT sistemi, derin öğrenme tabanlı bir mimariye sahiptir. Sistem, görüntülerin ön işlemden geçirilmesi ve ardından bölütleme, öznitelik çıkarma ve sınıflandırma adımlarıyla hedeflerin tespit ve sınıflandırılmasını sağlamaktadır. HBT sistemi, özellikle kara muharebe araçlarının tespiti ve tanımlanması için optimize edilmiştir.

HBT sistemi, odaklanılan kara muharebe araçlarını tespit etmek ve tanımlamak için YOLO v8 algoritmasını (derin öğrenme yöntemlerini) kullanarak otomatik olarak hedefleri saptar ve sınıflandırır. Bu kapsamda; otomatik özellik çıkarımı yapılmaktadır. Derin öğrenme mimarileri kendi başlarına veriden yüksek düzeyde özellikler çıkartmaktadır. Bu sistemin sonuçlarını kullanarak ikincil bir hesaplama (mesafe tahmini) ile ilave bilgi elde edilir. Burada uygulanan transfer öğrenme metodu ile önceden eğitilmiş bir modeli kullanarak, belirli görevler için ek özniteliklerin çıkarılma işlevidir. Belirlenen hedefler ve diğer bağlamsal özellikler kullanılarak önerilerde (karşı tedbir) bulunur.

HBT'ye ait yazılım ve kaynak kodu detayları EK 5'tedir.

5.2. Sistemin Kullanım Adımları

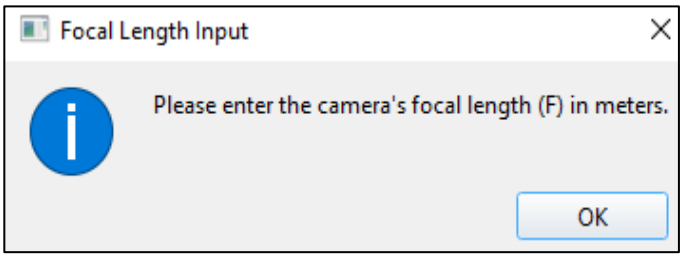
HBT sistemi, hedef belirleme ve tanımlama süreçlerini otomatikleştirerek insan kaynaklı hataları en aza indirmeyi ve hızlı karar alma süreçlerini desteklemeyi amaçlamaktadır. Sistemin kullanım adımları, kullanıcı dostu bir arayüz aracılığıyla basitleştirilmiştir.

Sistem, çalıştırılmaya müteakip canlı elektro-optik sistemlerine bağlanabilmekte, bağlanmadan önce odak mesafesini kullanıcıdan talep etmektedir. Ayrıca mesafesi doğru bilinen bir nesne (muharebe aracı) kullanılarak odak uzunluğu kalibrasyonu da

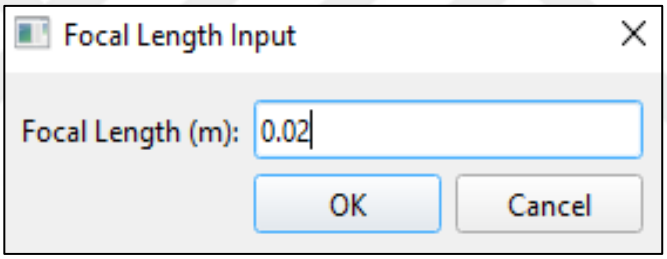
sağlanabilmektedir. Bu özellik sayesinde daha doğru mesafe tahminleri yapılabilmektedir.

HBT sisteminin kullanım adımları şu şekilde özetlenebilir:

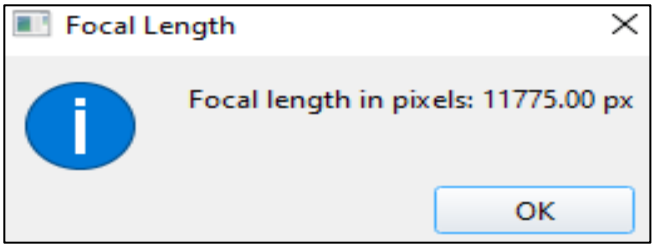
Sistem çalıştırma komutunu aldığında kullanıcıdan “Odak Mesafesi” bilgisini talep eder (Şekil 5.1). Kullanıcı “Tamam” (OK) tuşuna bastığında odak mesafesi giriş ekranı açılır (Şekil 5.2). Odak mesafesi piksel cinsinden belirlenir ve sisteme “OK” tuşuna basılarak kaydedilir. Sistem odak mesafesini hesaplayarak kullanıcıyı bilgilendirir (Şekil 5.3).



Şekil 5.1. Odak Mesafe Girişi İsteği

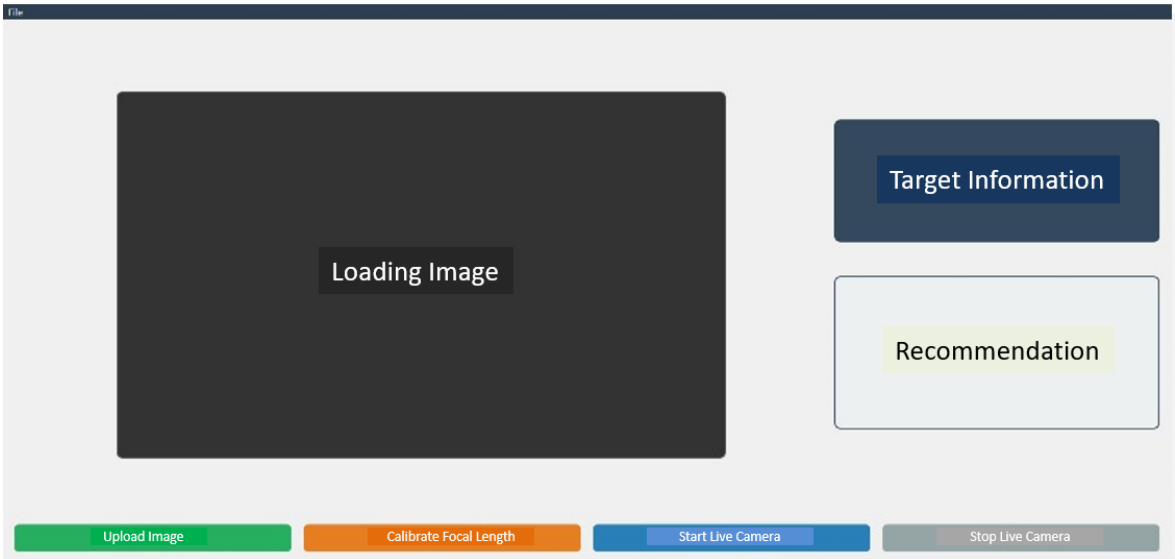


Şekil 5.2. Odak Mesafe Girişi



Şekil 5.3. Pixel Olarak Odak Mesafesi

Bu adımların ardından sistem, hedef tespiti ve mesafe tahmini işlemlerine başlar. Uygulama Ekranı (Şekil 5.4) “Upload Image” (resim yükleme), “Calibrate Focal Length” (odak mesafe kalibrasyonu), “Start Live Camera” (elektro-optik canlı kameraya bağlanma) ve “Stop Live Camera” (canlı kamera akışı durdurma) kesme tuşları bulunmaktadır.



Şekil 5.4. Uygulama Ana Ekranı

Resim yüklendiğinde, HBT sistemi tespit, teşhis, mesafe tahminini ve tavsiyesini Şekil 5.5'de gösterildiği şekilde oluşturarak göstermektedir.

Şekil 5.5. HBT Sistemi Örnek Deneme

5.3. Sistem Kalibrasyonu

Mesafe tahminlerinde, hesaplanan mesafelerin gerçek mesafelerden ortalama %10 daha kısa olduğu tespit edilmiştir. Kalibrasyonun referans nesneler kullanarak ve/veya çoklu nokta kalibrasyonu kullanılarak geliştirilebileceği bilinmektedir.

Kısaca, referans nesneler kullanma yönteminde, lazer mesafe ölçer kullanarak bilinen mesafelere sahip referans nesnelerle sistem kalibre edilebilir. Bu, hesaplanan mesafeler ile gerçek mesafeler arasında doğrudan bir karşılaştırma yapılmasına ve sistemin bu farkı öğrenecek şekilde ayarlanmasına imkân tanıyabilir. Çoklu nokta kalibrasyonu yönteminde ise farklı mesafelerde (örneğin, yakın, orta, uzak) yapılan ölçümlerle, sistemin tüm menziller için doğruluğu sağlanır. Her bir mesafe aralığında düzeltme faktörleri geliştirilebilir.

Çalışmada, sisteme eklenen bilinen bir mesafedeki zırhlı araç tipi ve mesafesine göre odak mesafesinin yeniden hesaplanabilmesi kabiliyeti eklenmiştir. Bu kabiliyet kullanıldığında, bilinen mesafelere göre yapılan ölçümlere göre hata oranının %1'e kadar düşürüldüğü tespit edilmiştir. Verilerin kayıt altına alınarak, elde edilecek bir çarpan yardımıyla kullanım esnasında da doğruluk artırımının sağlanabileceği değerlendirilmektedir.

6. SONUÇ

Bu çalışmada, YOLOv8 nesne algılama modeli ve bu modelin üzerine eklenen SE modülü kullanılarak muharebe araçlarının tespiti gerçekleştirilmiş ve bu araçların tipleri kategorize edilmiştir. Transfer öğrenme metodu yardımıyla araçların mesafeleri tahmin edilmiş ve elde edilen verilerle bir karar matrisi oluşturularak karşı tedbir önerileri geliştirilmiştir. Modelin tespit ve teşhis oranları, ortalama kesinlik yüzdesi (mAP) cinsinden %88,38 ve genel kesinlik yüzdesi ise %93 olarak hesaplanmıştır. Bu sonuçlar değerlendirildiğinde, modelin araç tespitinde yüksek doğruluk ve etkinlik sağladığını göstermektedir.

Özellikle aracın türü ve mesafesinin pasif bir şekilde doğru olarak tanımlanması, modelin gelecekte elektro-optik sistemler veya askeri araçlarla kolayca entegre edilebileceğini göstermektedir. Bu bulgular, modelin askeri uygulamalar için pratik bir çözüm sunabileceğine işaret etmektedir.

Bulgular, nesne tanıma ve mesafe tahmini uygulamalarının askeri alandaki potansiyelini açıkça ortaya koymaktadır. Gelecek çalışmalarda, modelin performansının daha da iyileştirilmesi ve farklı operasyonel senaryolar altında test edilmesi önerilmektedir. Modelin geliştirilmesine yönelik öneriler, gelecekteki çalışmalara rehberlik edebilir.

Ana algoritmaya entegre edilen pasif mesafe ölçüm algoritması sayesinde, tespit edilen muharebe araçlarına yönelik etkili karşı tedbir önerileri geliştirilmiştir. Algoritmalara eklenecek detay öğrenme teknikleri, tespit doğruluğunu önemli ölçüde artırabilir. Derin öğrenme modellerine bu detayları öğrenme kapasitesi kazandıracak ek katmanlar ve tekniklerin entegrasyonu, mevcut sınıflandırma sisteminin değerini daha da artıracaktır. Bu tür tekniklerin uygulanması, modelin doğruluğunu ve kullanım alanlarını genişletebilir.

Mevcut araç sınıfları, kendi içlerinde daha ayrıntılı alt bölümlere ayrılabilir. Örneğin, zırhlı olmayan kamyonlar, hava savunma araçları, çekili toplar vb. Ayrıca, bazı zırhlı muharebe araçlarında kullanılan "Ek Izgara ve Ağ Zırhları" ile "Reaktif Zırhlar", görsel olarak kolayca ayırt edilebilmektedir.

Algoritmalara eklenecek detay öğrenme teknikleri, bu tür tespitlerin doğruluğunu önemli ölçüde artırabilir. Bu bağlamda, derin öğrenme modellerinin bu detayları öğrenmesine olanak tanıyan ek katmanlar ve teknikler eklemek, mevcut sınıflandırma sistemine değer katacaktır.

Gelecek çalışmalar, bu yöntemlerin entegrasyonu ve yeni algoritmaların uygulanabilirliğini inceleyerek, askeri alanda nesne tanıma ve sınıflandırma sürecini daha da geliştirebilir.

Deneyler, modelin hassasiyet (precision) ve hatırlama (recall) açısından tatmin edici sonuçlar verdiğini açıkça göstermiştir. Bu bulgular, modelin genel olarak başarılı performans sergileyebileceğini göstermektedir. Bununla birlikte, modelin daha fazla özellik çıkarmasına olanak tanıyacak şekilde eğitilmesi için daha kaliteli fotoğraflar ve videoların kullanılması gerektiği de tespit edilmiştir.

Bu özellik, eğitim sürecinin ardından, yaygın olarak kullanılan muharebe araçlarının gömülü elektro-optik sistemlerinde kullanılması açısından son derece yararlı olabilir. Böylece, yanıt süresi iyileştirilmiş olur ve sistem, özellikle hedef yönetimi için verilere dayalı karar desteği sağlayarak, verilerin korelasyonu ve askeri perspektifte veri füzyonu gerçekleştirebilir.

Sonuç olarak, oluşturulan modelin mevcut yeteneklerinin artırılması, muharebe sahasında daha hızlı ve etkili karar alma süreçlerini destekleyecek önemli bir potansiyele sahiptir. Gelecek çalışmalar, bu doğrultuda modelin entegrasyonunu ve performansını daha da artırmaya yönelik stratejileri değerlendirebilir.

KAYNAKLAR

[1] G.-I. Toroi, "The Importance of Mission Analysis in Modern Military Operations," PhD thesis, National Defense University, [online]

[2] United States. Targeting. "Army Techniques Publication 3-60," Headquarters, Department of the Army, 2015.

[3] United States. Joint Operations. "Army Techniques Publication 3-0," Headquarters, Department of the Army, 2018.

[4] N. Smeu, and C. Dobrescu, "Targeting, Target Acquisition and the Military Decision-Making Process," Master's thesis, Carol I National Defence University.

[5] D. 01well, and A. Washbura, "Internetting of Fires," NP&OR-02-003-PR, 2002, pp. 3-5.

[6] H. Cai, J. Liu, Y. Chen, and H. Wang, "Survey of the research on dynamic weapon-target assignment problem," Journal of Systems Engineering and Electronics, vol. 17, no. 3, pp. 559-565, 2006.

[7] T. P. Banerjee, and S. Das, "Multi-sensor data fusion using support vector machine for motor fault detection," Information Sciences, vol. 217, pp. 96–107, 2012.

[8] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," Information Fusion, vol. 14, pp. 28–44, 2013.

[9] R. C. King et al., "Application of data fusion techniques and technologies for wearable health monitoring," Medical Engineering & Physics, vol. 42, pp. 1–12, 2017.

[10] Joint Directors of Laboratories, "JDL Model 4".

[11] S. K. Das, "High-level data fusion," Artech House, 2008.

[12] P. Sharma, S. Gupta, S. Vyas, and M. Shabaz, "Object detection and recognition using deep learning-based techniques," IET Communications, vol. 00, pp. 1–11, 2022. doi: 10.1049/cmu2.12513

[13] Z. Li et al., "Wide Area Remote Sensing Image On Orbit Target Extraction and Identification Method," in Proc. IEEE Int. Conf. Signal, Inf. and Data Process. (ICSIDP), 2019, pp. 1-8. doi: 10.1109/ICSIDP47821.2019.9173401.

[14] B. Selbes and M. Sert, "Multimodal vehicle type classification using convolutional neural network and statistical representations of MFCC," in Proc. 14th IEEE Int. Conf. Adv. Video and Signal Based Surveillance (AVSS), Lecce, 2017, pp. 1-6.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Proc. 25th Int. Conf. Neural Inf. Process. Syst.

(NIPS), 2012, pp. 1097-1105. doi: 10.1145/3065386.

[16] Suprayitno et al., "Real-time military person detection and classification system using deep metric learning with electrostatic loss," Bulletin of Electrical Engineering and Informatics, vol. 12, pp. 338-354, 2023. doi: 10.11591/eei.v12i1.4284.

[17] F. Huang et al., "Using deep learning in an embedded system for real-time target detection based on images from an UAV," Int. J. Digital Earth, vol. 16, pp. 910-936, 2023. doi: 10.1080/17538947.2023.2187465.

[18] A. Vaswani et al., "Attention is all you need," in Advances in Neural Inf. Process. Syst., vol. 30, 2017.

[19] R. Gandhi, "R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms," Medium. [Online]. Available://medium.com/tow-towards-data-science/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[20] J. Hu, "Research on Fast Tracking Algorithm of Moving Target Based on Optical Flow Method," PhD dissertation, Xidian University, Xi'An, 2014. [Google Scholar].

[21] X. Liu, "End-to-end Target Detection and Attribute Analysis Algorithm Based on Deep Learning and its Application," MSc thesis, South China University of Technology, Guangzhou, 2017. [Google Scholar].

[22] NATO, "APP-6 (D) NATO Joint Military Symbology, Chapter 3, Land Symbols," Oct. 2017.

[23] United States Army, "ATP 2-01.3 Intelligence Preparation of the Battlefield, Chapter 5, Step 3: Evaluate the Threat," Mar. 2019.

[24] United States Army, "ATP 3-21.10 Infantry Rifle Company, Appendix F-Armored, Stryker, and Mounted Employment," May 2018.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR), 2016.

[26] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.

[27] O. Oluwaseyi, M. Irhebhude, A. Evwiekpaefe, "A Comparative Study of YOLOv5 and YOLOv7 Object Detection Algorithms," Journal of Computing and Social Informatics. 2. 1-12. 10.33736/jcsi.5070.2023.

[28] R. Zhang, C. Xie and L. Deng, "A Fine-Grained Object Detection Model for Aerial Images Based on YOLOv5 Deep Neural Network," in Chinese Journal of Electronics, vol. 32, no. 1, pp. 51-63, January 2023, doi: 10.23919/cje.2022.00.044.

[29] C. Cui, R. Wang, Y. Wang, F. Zhou, X. Bian and J. Chen, "Research on Optical

Remote Sensing Image Target Detection Technique Based on DCH-YOLOv7 Algorithm," in IEEE Access, vol. 12, pp. 34741-34751, 2024, doi: 10.1109/ACCESS.2024.3368877.

[30] J. Hu, L. Shen and G. Sun, "Squeeze-and-Excitation Networks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 7132-7141, doi: 10.1109/CVPR.2018.00745.

[31] Wu, Tianyong & Dong, Youkou. (2023). YOLO-SE: Improved YOLOv8 for Remote Sensing Object Detection and Recognition. Applied Sciences. 13. 12977. 10.3390/app132412977.

[32] M. Lan, J. Wang and L. Zhu, "Perception and Range Measurement of Sweeping Machinery Based on Enhanced YOLOv8 and Binocular Vision," in IEEE Access, vol. 11, pp. 126398-126408, 2023, doi: 10.1109/ACCESS.2023.3331013.

[33] B. Wei et al., "Remote Distance Binocular Vision Ranging Method Based on Improved YOLOv5," in IEEE Sensors Journal, vol. 24, no. 7, pp. 11328-11341, 1 April1, 2024, doi: 10.1109/JSEN.2024.3359671.

[34] G. Jocher, "YOLOv5 by Ultralytics (Version 7.0)" [Computer software]. 2020 doi: 10.5281/zenodo.3908559

[35] M. A. Khan, P. Paul, M. Rashid, M. Hossain, and M. A. R. Ahad, "An AI-Based Visual Aid With Integrated Reading Assistant for the Completely Blind," IEEE Trans. Human-Machine Syst. doi: 10.1109/THMS.2020.3027534

[36] M. Vajgl, P. Hurtik, and T. Nejezchleba, "Dist-YOLO: Fast Object Detection with Distance Estimation," Applied Sciences, vol. 12, p. 1354, 2022. doi: 10.3390/app12031354

[37] Roboflow. "Roboflow Annotate," [Online]. Available://roboflow.com/annotate.

[38] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8", v=8.0.0, 2023. [Online]. Available: //github.com/ultralytics/ultralytics

[39] Ultralytics, "YOLOv8 - Ultralytics YOLO Docs." [Online]. Available://docs.ultralytics.com/models/yolov8/#overview

[40] VISO, "YOLOv8: A Complete Guide [2025 Update]," viso.ai. [Online]. Available://viso/ai/deep-learning/yolov8-guide/

[41] J. R. Glenn et al., "Ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations," 2021. [Online]. Available: //doi.org/10.5281/zenodo.3908559

[42] Woo, S., Park, J., Lee, J. Y., & Kwon Lee, K. (2018). CBAM: Convolutional Block Attention Module. Proceedings of the European Conference on Computer Vision (ECCV), 3-19.

[43] Wang, X., Girshick, R., Farhadi, A., et al. (2018). Non-Local Neural Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 7794-7803.

[44] Vaswani, A., Shard, N., Parmar, N., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems, 5998-6008

EKLER

EK 1: Diğer YOLO Algoritmaları Metrik ve Tabloları

Yapılan çalışmalar kapsamında farklı YOLO versiyonları da karşılaştırma yapabilmek maksadıyla denenmiştir. Denemelerde iyileştirilmiş veri kümesi kullanılmıştır. Eğitim, test ve geçerlemeler aynı bölümlenme kullanılarak yapılmıştır. Müteakip bölümde yapılan denemeler sonucunda, elde edilen temel metrikler Tablo Ek 1.1'de sunulmuştur.

Tablo Ek 1.1. YOLO 5, 10 ve 11 Algoritmaları Deneme Sonuçları

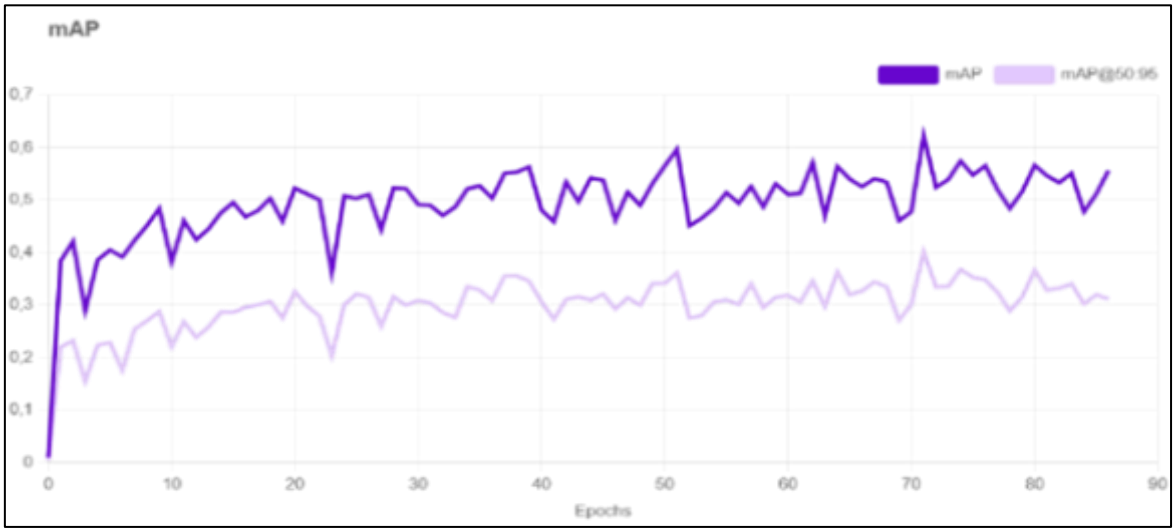
Sınıf	**Görüntü**	**Ortalama Doğruluk Oranı**		
		YOLO5m	**YOLO10m**	**YOLO11m**
Hepsi	**5984**	62,3%	68,7%	70,5%
Tespit Hızı		79 FPS	80 FPS	100 FPS

Eğitim sırasında, veri kümesinin yollarını ve eğitilecek sınıf sayısını tanımlamak amacıyla bir YAML dosyası yapılandırılmıştır. Sadece YOLO'nun ilgili versiyonuna göre zorunlu düzenlemeler yapılmıştır. Model, parti büyüklüğü 32 olacak şekilde eğitilmiş ve süreç boyunca veriler, "Roboflow" platformu üzerinden gerçek zamanlı olarak görselleştirilip izlenmiştir. Dönem sayılarının farklı olması YOLO mimarisinde aşırı öğrenmeyi engellemek için geliştirimiş olan otomatik durma sisteminden kaynaklanmaktadır.

EK 1.1. YOLOV5

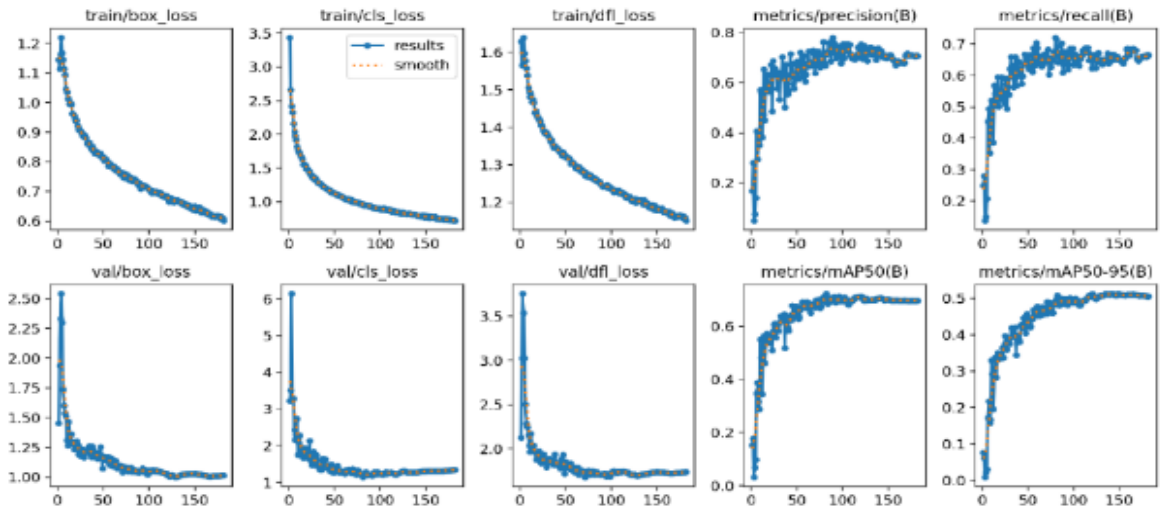
YOLOv5 nesne algılama metodolojilerinde kullanıma verildiği dönemde bir eşik olarak kabul görmektedir. Temel mimarisinden kaynaklanan YOLOv5 deneysel sonuçlar ve genel özellikleri göz önüne alındığında verimli bir alternatif sunmaktadır. Aşağıda sırasıyla, döngüler bazında YOLOv5 kesinlik sonuçları, genel kutu/sınıf/obje kayıpları ve

eğitim/validasyon kayıp görselleri ile hata matrisi bulunmaktadır.



Şekil EK 1. 1. Veri Kümesi YOLO V5 Sonuçları

Şekil EK 1. 2. Veri Seti YOLO V5 (Box-Class-Object Loss)



Şekil EK 1. 3. Eğitim ve Validasyon Kayıpları ve Metrikleri

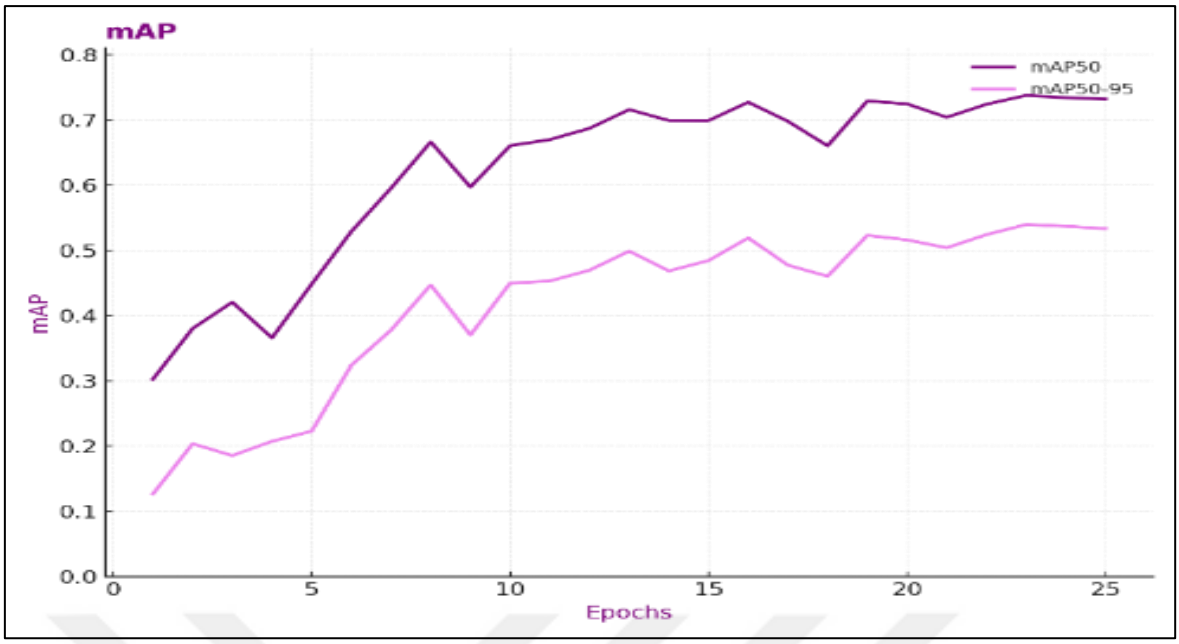
Şekil-EK-1- 1. YOLO V5 Hata Matrisi

EK 1.2. YOLOV10

YOLOv10, şu temeller üzerine inşa edilmiştir. UltralyticsPython Tsinghua Üniversitesi'ndeki araştırmacılar tarafından geliştirilen bir pakettir. Gerçek zamana yakın görüntü algılamaya yeni bir yaklaşım getirerek önceki YOLO sürümlerinde bulunan hem işlem sonrası hem de model mimarisi eksiklikleri giderilmeye çalışılmıştır.

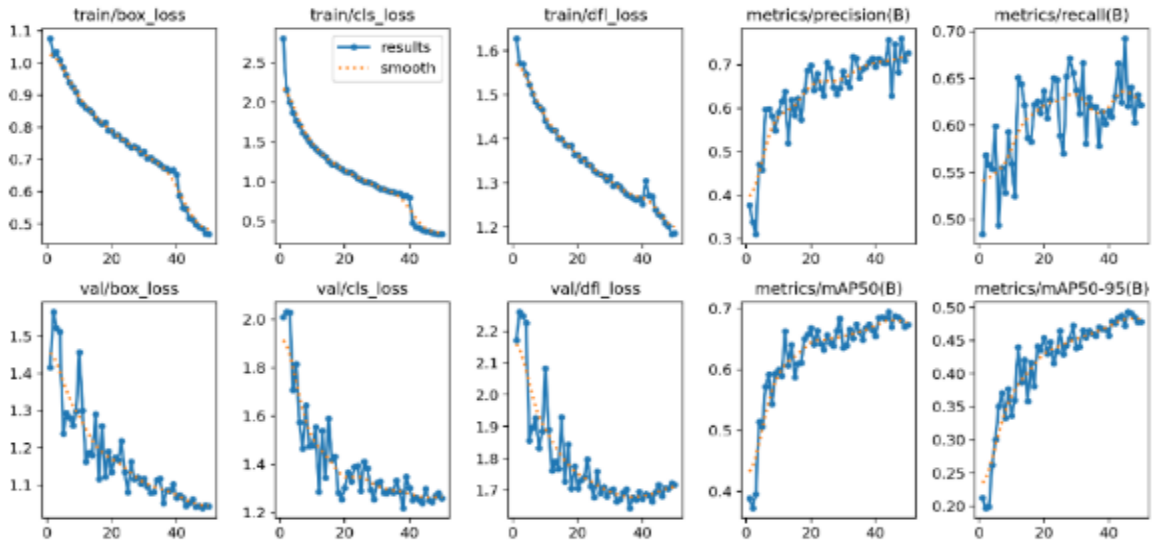
Maksimum olmayan bastırmayı ortadan kaldırarak ve çeşitli model bileşenlerini optimize etmeye çalışmaktadır. Dezavantajı, daha basit mimarilere kıyasla karmaşıklık yaratabileceği ve henüz tam olarak olgunlaşmadığı olarak ifade edilebilir. Büyük görüntülerin ayrıntılı analizlerinde son derece başarılıdır.

Aşağıda sırasıyla, döngüler bazında YOLOv10 kesinlik sonuçları, genel kutu/sınıf/obje kayıpları ve eğitim/validasyon kayıp görselleri ile hata matrisi bulunmaktadır.

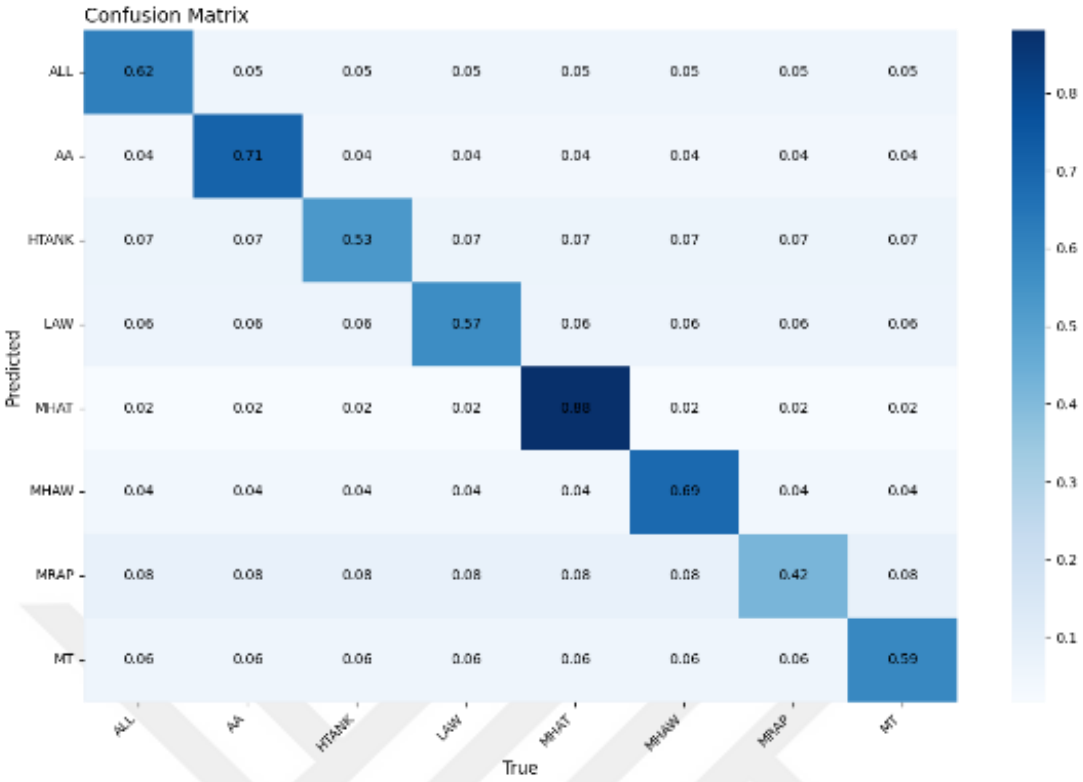


Şekil EK 1. 5. Veri Kümesi YOLO V10 Sonuçları

Şekil EK 1. 4. Veri Seti YOLO V10 (Box-Class-Object Loss)



Şekil EK 1. 6. YOLO V10 Eğitim ve Validasyon Kayıpları ve Metrikleri

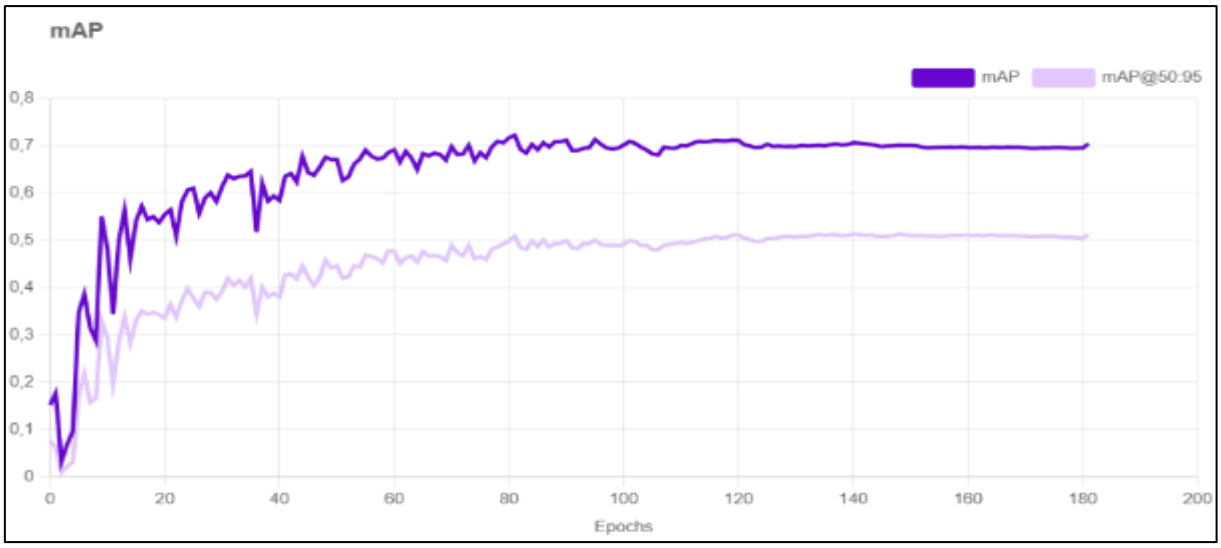


Şekil EK 1. 7. YOLO V10 Hata Matrisi

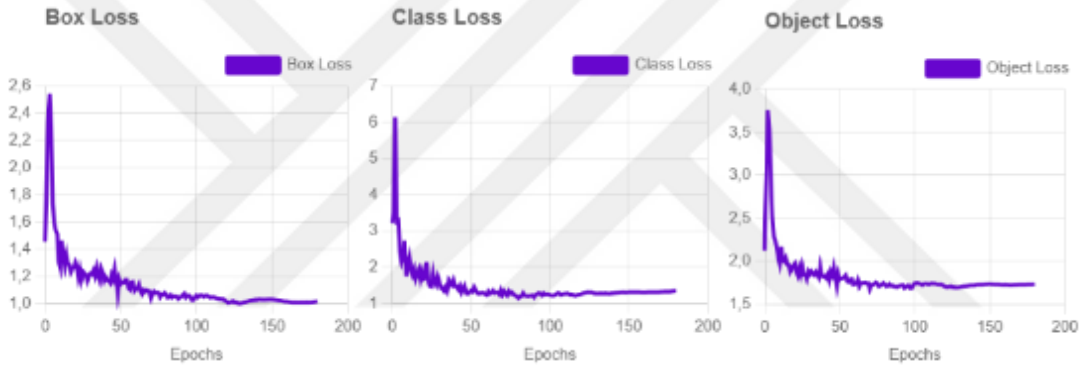
EK 1.3. YOLOV11

YOLOv11 en son yinele olarak ortaya çıkmıştır. Önceki YOLO sürümlerinin etkileyici ilerlemelerini temel alan YOLOv11, mimari ve eğitim yöntemlerinde önemli iyileştirmeler sunmaya çalışmaktadır. Parametre sayıları YOLOv8'e göre %22 daha azaltılmıştır. Entegrasyon yeteneği YOLOv8'e göre daha düşüktür. Avantajı, tespit hızının daha yüksek ve hassas olmasıdır.

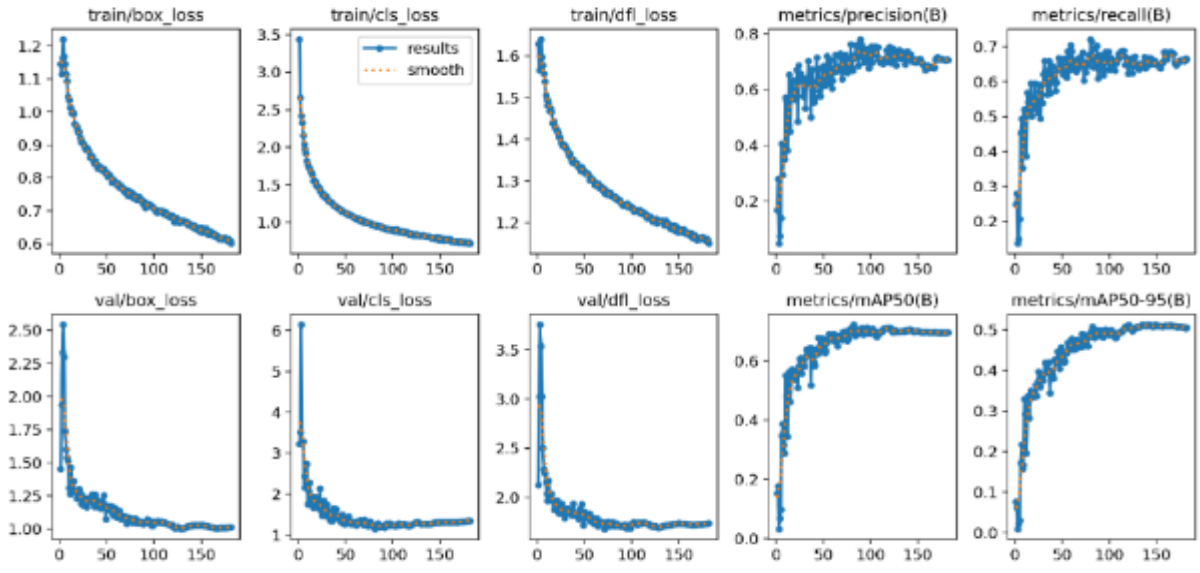
Aşağıda sırasıyla, döngüler bazında YOLOv11 kesinlik sonuçları, genel kutu/sınıf/obje kayıpları ve eğitim/validasyon kayıp görselleri ile hata matrisi bulunmaktadır.



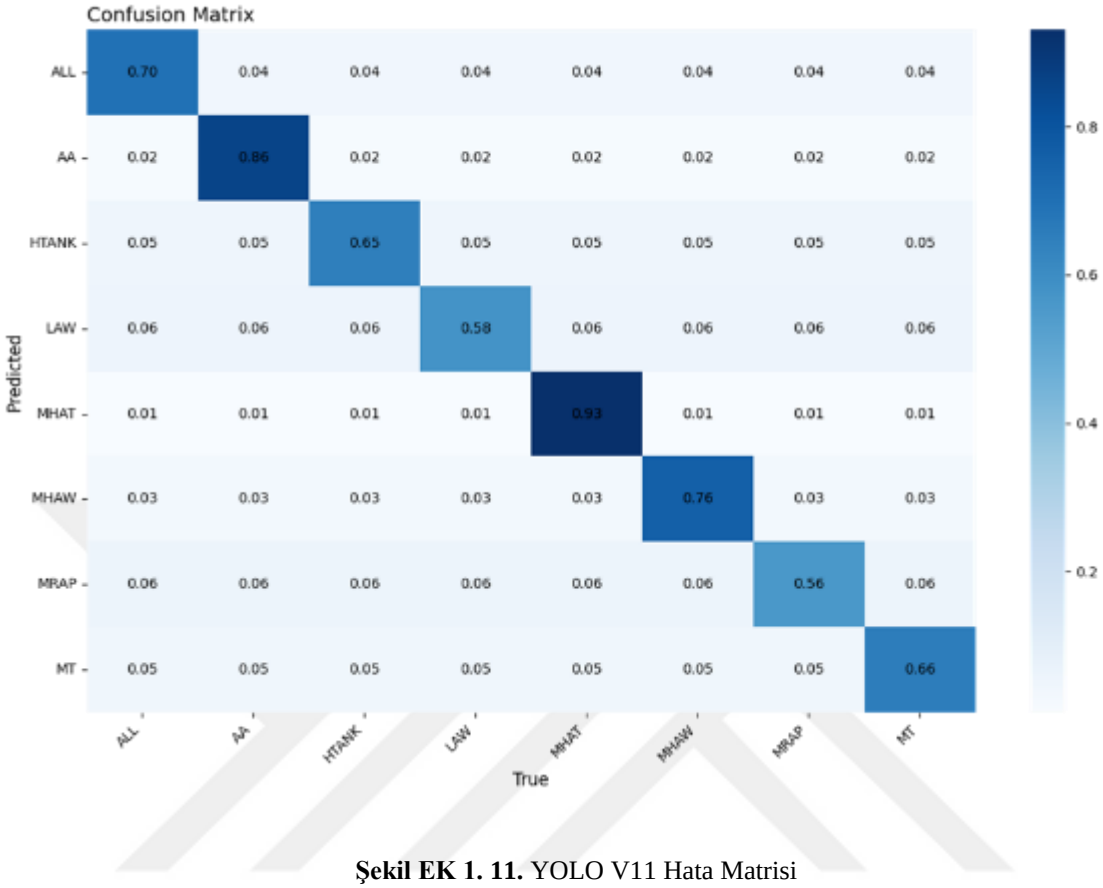
Şekil EK 1. 8. Veri Kümesi YOLO V11 Sonuçları



Şekil EK 1. 9. Veri Seti YOLO V11 (Box-Class-Object Loss)



Şekil EK 1. 10. YOLO V11 Eğitim ve Validasyon Kayıpları ve Metrikleri



Şekil EK 1. 11. YOLO V11 Hata Matrisi

EK 2: Dikkat Modülü- Sıkıştırma ve Uyarım Bloğu

Sıkıştırma ve Uyarım Bloğu (Squeeze-and-Excitation - SE) modülü oluşturularak, YOLOv8 algoritmasına adapte edilmiştir. Bu pseudo kod, SE modülünün temel iç işleyişini ve bir konvolüsyon katmanına nasıl entegre edildiğini göstermektedir.

EK 2.1. Sıkıştırma ve Uyarım bloğu – SE (Pseudo Kod)

```
Fonksiyon SE_Module (input_tensor, reduction_ratio)
# 1. Squeeze işlemine başla
# Giriş tensörünü global ortalama havuzlama ile sıkıştır
squeezed_tensor = GlobalAveragePooling(input_tensor)
# 2. İki tam bağlı (dense) katman oluştur
# İlk katman (sıkıştırma)
squeezed_tensor = Dense(squeezed_tensor, units=input_tensor.boyutları[kanal], activation='ReLU')
# İkinci katman (heyecanlandırma)
excited_tensor = Dense(squeezed_tensor, units=input_tensor.boyutları[kanal], activation='Sigmoid')
# 3. Giriş tensörünün kanallarını yeniden ölçeklendir
# Her bir kanal için ağırlıklar hesapla
scaled_tensor = input_tensor * excited_tensor
# 4. Sonucu geri döndür
Geri dön scaled_tensor
```

EK 2.2. Açıklamalar

Squeeze İşlemi: Giriş tensörü global ortalama havuzlama işlemi ile sıkıştırılır. Bu işlem, her kanaldaki özelliklerin ortalamasını alır ve daha düşük boyutlu bir tensör oluşturur.

Dense Katmanlar: İlk Dense katmanı (sıkıştırma katmanı), giriş kanalının boyutunu azaltır. İkinci Dense katmanı (heyecanlandırma katmanı), yeniden boyutlandırılmış ve ReLU aktivasyon fonksiyonuna tabi tutulmuş olan değerlere sigmoid uygulayarak her bir kanal için önem değerleri üretir.

Ağırlıkların Uygulanması: Elde edilen `excited_tensor`, giriş tensörüne çarpılarak her kanal için dikkat ağırlıkları uygulanır.

Sonuç: Sonuç tensörü dönme işlemine tabi tutulur, bu da giriş verisinin özelliklerinin önemine göre yeniden ölçeklendirilmiş halidir.

EK 2.3. Kullanım

YOLOv8'in mimarisinin içinde SE modülü, konvolüsyon katmanlarının ardından yerleştirilmiştir.

```
plaintext
giriş_tensörü = Convolution(input)
output_tensörü = SE_Module(giriş_tensörü, reduction_ratio)
```

EK 3: Mesafe Ölçme Yazılım Kodu

EK 3.1. Mesafe Hesaplama (Pseudeo Code)

Algoritma 1: Mesafe hesaplamak için kullanılan kod satırları aşağıda sunulmuştur.

```
1: def calculate_distance (self, actual_height, box_height_pixels):
2: """
3: Calculates the distance to the object.
4: Distance = (F * R) / h'
5: """
6: if box_height_pixels == 0:
7: return float('inf')
8: distance = (self.focal_length_pixels * actual_height) / box_height_pixels
9: return distance
```

Mesafe Hesaplama "Calculate_Distance" Metodu: Nesnenin gerçek yüksekliği (actual_height) ve algılanan piksel yüksekliği (box_height_pixels) kullanılarak mesafe hesaplanır.

Formül: Distance = (F * R) / h' (EK-3.1)

F: Odak uzunluğu (focal_length_pixels)

R: Gerçek nesne yüksekliği

h': Algılanan nesne yüksekliği piksel cinsinden box_height_pixels sıfır ise, mesafe sonsuz (float('inf')) olarak atanır.

EK 3.2. Odak Uzunluğu Kalibrasyonu (Pseudeo Code)

Algoritma 2: Lazer mesafe ölçer veya gerçek ölçüm yapılarak bilinen mesafelere sahip referans nesneler kullanılarak sistemin kalibre edilmesine yönelik hazırlanan kod satırları aşağıda sunulmuştur.

```
1: def calibrate_focal_length(self):
2: """
```

```
3: Calibration process: Upload an image with a known distance and object height.
"""
4: QMessageBox.information( self, "Calibration", "To calibrate, upload an image
   containing an object at a known distance with a known height."    )
5: options = QFileDialog.Option.ReadOnly
6: file_path, _ = QFileDialog.getOpenFileName( self, "Select Calibration Image", "",
   "Images (*.png *.jpg *.jpeg)", options=options)
7: if file_path:
8: image = cv2.imread(file_path)
9: if image is None:
10:    QMessageBox.critical(self, "Error", "Failed to load the image.")
11:     return
12:    self.frame_height, self.frame_width = image.shape[:2]
13:    detections, annotated_image = self.detect_objects(image)
14:    if detections:
15:    # Ask user for the known distance
16:    distance, ok = QInputDialog.getDouble(
17:    self, "Calibration Distance",
18:    "Enter the known distance to the object (meters):",
19:    decimals=2, min=0.1, step=0.1)
20:    if not ok or distance <= 0:
21:    QMessageBox.warning(self, "Calibration Cancelled", "Calibration process was
       cancelled.")
22:    return
23:    # Iterate through detections to find a suitable object for calibration for detection
       in detections:
            class_name = detection['class']
            confidence = detection['confidence']
            box_height = detection['height']
24:    # Debug: Print detection details
25:    print(f"Calibration Details - Class: {class_name}, Confidence: {confidence},
       Height: {box_height}")
26:    if (confidence > 0.7 and class_name in VEHICLE_DIMENSIONS):
          actual_height = VEHICLE_DIMENSIONS[class_name]["height"]
```

```
27:  # Convert normalized height to pixels
28:  if box_height <= 1.0:
29:  box_height_pixels = box_height * self.frame_height else:
30:  box_height_pixels = box_height  # Already in pixels
31:  # Prevent division by zero
32:  if box_height_pixels == 0:
33:  QMessageBox.critical( self, "Error", "Detected object's height is zero pixels. Cannot calibrate." )
34:  return
35:  # Calibration formula: F = (h' * distance) / R
36:  self.focal_length_pixels = (box_height_pixels * distance) / actual_height
37:  QMessageBox.information (self, "Calibration Successful", f"Focal length (pixels): {self. focal_length_pixels:.2f} px"  )
38:  print(f"Calibrated Focal Length: {self.focal_length_pixels:.2f} px")
39:  return
40:  else:
41:  QMessageBox.warning (self, "Calibration Error", "No suitable object found for calibration. Please check the image.")
42:  else:
43:  QMessageBox.warning( self, "Calibration Error", "No objects detected in the image for calibration." )
```

Odak Mesafesi Kalibrasyonunun Kullanım Adımları "Calibrate_focal_length":

Kullanıcıya, bilinen bir mesafede ve bilinen bir yüksekliğe sahip bir nesne içeren bir görüntü yüklemesi için bilgi verir. Yüklenen görüntüde nesneler tespit edilir. Kullanıcıdan nesnenin bilinen mesafesi alınır.

Tespit edilen nesneler arasında uygun bir nesne bulunursa, kalibrasyon formülü F = (h' * distance) / R kullanılarak odak uzunluğu (focal_length_pixels) hesaplanır ve saklanır.

Eğer uygun bir nesne bulunamazsa veya tespit edilen nesnenin yüksekliği sıfır ise, hata mesajı gösterilir.

EK 4: Karşı Tedbir Öneri Karar Matrisi

Muharebe Araçlarına karşı alınacak tedbirler bölüm 4.7 başlığı altında belirtildiği üzere silah sistemleri genel olarak hedefin tipi, korunma derecesi (zırh tipi) ve mesafesine göre "Kısa, Orta ve Uzun Menzilli" veya "Hafif, Orta ve Ağır Sınıf" kategorilerine ayrılmaktadır. Bu çalışmada, hedeflerin mesafesine bağlı olarak "Kısa", "Orta" ve "Uzak" mesafeleri tanımlamaları yapılmıştır. Bu sınıflandırmalar doğrultusunda da Tablo EK4.1,2,3 oluşturulmuştur.

Tablo EK 4. 1. Karşı Tedbir Öneri Matrisi-Kısa Mesafe

Muharebe Araç Türleri	**Minimum Mühimmat/Silah Sistemi Türü**
Kısa Mesafe (0-700)	**Tavsiye**
Hafif Zırhlı Tekerlekli (LAW) Araçlar	Orta Zırh Delici Mühimmat, RPG
Orta ve Ağır Zırhlı Tekerlekli (MHAW) Araçlar	Kısa Mesafe Silah Sistemi ile A/T Mühimmat
Orta ve Ağır Zırhlı Paletli (MHAT) Araçlar	Kısa Mesafe Silah Sistemi ile A/T Mühimmat
Askeri Kamyonlar (MT)	Kısa Mesafe Silah Sistemi ile A/T Mühimmat
Ağır (Ana Savaş) Tanklar (HT)	RPG
Kundağı Motorlu Topçu (AA)	RPG
Mayın Dayanıklı Pusuya Karşı Korumalı Araçları (MRAP)	RPG

Tablo EK 4. 2. Karşı Tedbir Öneri Matrisi-Orta Mesafe

Muharebe Araç Türleri	**Minimum Mühimmat/Silah Sistemi Türü**
Orta Mesafe (700-2500m)	**Tavsiye**
Hafif Zırhlı Tekerlekli (LAW) Araçlar	Zırh Delici Mühimmat, Kinetik Enerji Delici -Tank
Orta ve Ağır Zırhlı Tekerlekli (MHAW) Araçlar	Kısa Mesafe A/T Silahı ile Orta A/T Mühimmat, Kinetik Enerji Delici -Tank
Orta ve Ağır Zırhlı Paletli (MHAT) Araçlar	Zırh Delici Mühimmat, HEAT -Tank
Askeri Kamyonlar (MT)	Zırh Delici Mühimmat
Ağır (Ana Savaş) Tanklar (HT)	Kısa Mesafe A/T Silahı ile Ağır A/T Mühimmat, HEAT -Tank
Kundağı Motorlu Topçu (AA)	Kısa Mesafe A/T Silahı ile Ağır A/T Mühimmat, HEAT -Tank
Mayın Dayanıklı Pusuya Karşı Korumalı Araçları (MRAP)	Kısa Mesafe A/T Silahı ile Ağır A/T Mühimmat, HEAT -Tank

Tablo EK 4. 3. Karşı Tedbir Öneri Matrisi-Uzun Mesafe

Muharebe Araç Türleri	**Minimum Mühimmat/Silah Sistemi Türü**
Uzun Mesafe (2500m.den fazla)	**Tavsiye**
Hafif Zırhlı Tekerlekli (LAW) Araçlar	Orta A/T Mühimmat, Orta Mesafe A/T Silahı ile
Orta ve Ağır Zırhlı Tekerlekli (MHAW) Araçlar	Orta A/T Mühimmat, Orta Mesafe A/T Silahı ile
Orta ve Ağır Zırhlı Paletli (MHAT) Araçlar	Orta A/T Mühimmat, Orta Mesafe A/T Silahı ile
Askeri Kamyonlar (MT)	Orta A/T Mühimmat, Orta Mesafe A/T Silahı ile
Ağır (Ana Savaş) Tanklar (HT)	Ağır A/T Mühimmat, Orta Mesafe A/T Silahı ile
Kundağı Motorlu Topçu (AA)	Orta A/T Mühimmat, Orta Mesafe A/T Silahı ile
Mayın Dayanıklı Pusuya Karşı Korumalı Araçları (MRAP)	Orta A/T Mühimmat, Orta Mesafe A/T Silahı ile

EK 5: Kaynak Kod Açıklamaları

EK 5.1. HBT Sistemine Giriş;

PyQt6, OpenCV ve Roboflow API kullanılarak geliştirilmiş olan "HBT Sistemi" uygulamasının detaylı bir analizini ve paketlenme sürecini kapsamaktadır. Uygulama, kullanıcının yüklediği veya canlı kameradan aldığı görüntülerde nesne tespiti yaparak, tespit edilen nesnelerin mesafesini hesaplamakta ve bu mesafeye bağlı olarak öneriler sunmaktadır.

EK 5.2. Uygulamanın Genel Görünümü

Uygulama, kullanıcı dostu bir grafik arayüzü sunarak aşağıdaki temel işlevleri yerine getirmektedir:

Odak Uzunluğu Girişi ve Kalibrasyonu: Kullanıcı kameranın odak uzunluğunu (fokal length) sisteme girmesini sağlar. Bununla beraber kullanıcı bilinen bir nesne ve mesafe kullanarak odak uzunluğu kalibrasyonunuda yapılabilir.

Görüntü Yükleme: Kullanıcı, nesne tespiti yapmak istediği bir görüntüyü yükleyebilir.

Canlı Kamera Akışı: Uygulama, bilgisayara bağlı bir kameradan canlı görüntü alarak nesne tespiti yapabilir.

Nesne Tespiti ve Mesafe Hesaplama: Yüklenen veya canlı akıştan alınan görüntülerde nesneler tespit edilir, bu nesnelerin boyutlarına ve odak uzunluğuna bağlı olarak mesafeleri hesaplanır.

Öneri Sistemi: Hesaplanan mesafelere göre kullanıcılara uygun karşı tedbir ve mühimmat kullanımı için öneriler sunulur.

EK 5.3. Detaylı Kod Açıklaması

Aşağıda, uygulamanın kodunun her bir bileşeni ve işlevi detaylı bir şekilde açıklanmıştır.

Ek 5.3.1İthalatlar ve Bağımlılıklar

```
import sys
import base64
import requests
import cv2
import numpy as np
from PyQt6.QtWidgets import (
QApplication, QMainWindow, QLabel, QPushButton, QVBoxLayout, QWidget, QFileDialog, QHBoxLayout, QTextEdit, QMessageBox, QMenuBar, QMenu, QInputDialog,
QDialog, QLineEdit, QFormLayout, QDialogButtonBox )
from PyQt6.QtGui import QPixmap, QImage, QFont, QAction
from PyQt6.QtCore import QTimer, Qt
```

sys: Sistem parametrelerine ve fonksiyonlarına erişim sağlar.

base64: Veriyi Base64 formatına dönüştürmek için kullanılır (görüntü verilerini kodlamak için).

requests: HTTP istekleri göndermek için kullanılır (Roboflow API ile iletişim için).

cv2 (OpenCV): Görüntü işleme ve kamera akışı için kullanılır.

numpy: Sayısal hesaplamalar ve görüntü verisi manipülasyonu için kullanılır.

PyQt6**:** Grafik kullanıcı arayüzü (GUI) oluşturmak için kullanılır.

EK 5.3.2. API Kimlik Bilgileri ve Konfigürasyon

API_KEY: Roboflow API'ye erişim sağlamak için kullanılan anahtar.

MODEL_ENDPOINT**:** Kullanılacak olan Roboflow modelinin uç noktası (endpoint).

EK 5.3.3. Araç Boyutları ve Mesafe Kategorileri

Mesafe tahminleri için yazılımda kullanılan araç ortalama uzunluk ve yükseklik değerleri aşağıda sunulmuştur. Ayrıca, örnek olarak ağır sınıf tanklar için kullanılan uzunluk ve yükseklik bilgileri Tablo EK 5. 1'de verilmektedir.

Tablo EK 5. 1. Ana Muharebe Tankları Uzunluk ve Yükseklik Bilgileri

Ana Muharebe Tankı (HT)	Uzunluk	Yükseklik
Abraams	9,77	2,44
M-60	9,3	3,27
K2	7,5	2,4
Chalenger	8,3	2,5
Leopard	9,6	2,82
T-90	9,53	2,22
Merkava	8,63	2,66
Sabra	6,95	3,27
Leclerc	6,95	3,27
T-99	7	2,35
Ortalama	**8,353**	**2,72**
Standart Sapma	1,174716	0,41328

```
# Vehicle dimensions for distance calculation (in meters)
VEHICLE_DIMENSIONS = {
"AA": {"height": 3.4, "length": 12.0},    # Armored Artillery
"MHAW": {"height": 2.4, "length": 7.8},    # Medium Heavy Armored Wheeled
"HT": {"height": 2.7, "length": 8.35},    # Heavy Tank
"LAW": {"height": 2.1, "length": 5.0},    # Light Armored Wheeled
"MHAT": {"height": 2.5, "length": 6.2},    # Medium Heavy Armored Tracked
"MRAP": {"height": 3.2, "length": 7.35},  # Mine-Resistant Ambush Protected
"MT": {"height": 2.75, "length": 8.1},    # Military Trucks}
# Distance categories for recommendations
DISTANCE_CATEGORIES = {
"Short": {"min": 50, "max": 700, "recommendation": "Use short-range ammunition (RPG).", "color": "#27ae60"},
"Medium": {"min": 700, "max": 2500, "recommendation": "Use medium-range ammunition.", "color": "#e67e22"},
```

"Long": {"min": 2500, "max": float('inf'), "recommendation": "Use long-range ammunition.", "color": "#c0392b"},}

VEHICLE_DIMENSIONS: Tespit edilecek araç türlerinin gerçek boyutlarını (yükseklik ve uzunluk) tanımlar. Bu bilgiler, mesafe hesaplamalarında kullanılır.

DISTANCE_CATEGORIES: Hesaplanan mesafelere göre kullanıcılara önerilecek mühimmat türlerini ve bu önerilere karşılık gelen renkleri tanımlar.

EK 5.3.4. Odak Uzunluğu Girişi İçin Dialog Sınıfı

```
class FocalLengthDialog(QDialog):
def __init__(self):
super().__init__()
self.setWindowTitle("Focal Length Input")
self.layout = QFormLayout(self)
self.line_edit = QLineEdit(self)
self.line_edit.setPlaceholderText("Enter focal length (meters)")
self.layout.addRow("Focal Length (m):", self.line_edit)
self.buttons = QDialogButtonBox(
QDialogButtonBox.StandardButton.Ok |
QDialogButtonBox.StandardButton.Cancel,
Qt.Orientation.Horizontal, self )
self.buttons.accepted.connect(self.accept)
self.buttons.rejected.connect(self.reject)
self.layout.addWidget(self.buttons)
def get_focal_length(self):
return self.line_edit.text()
```

FocalLengthDialog: Kullanıcıdan kameranın odak uzunluğunu metre cinsinden almak için özel bir dialog penceresi sağlar.

get_focal_length**:** Kullanıcının girdiği odak uzunluğunu döndürür.

EK 5.3.5. Ana Uygulama Sınıfı

```
class ObjectDetectionApp(QMainWindow):
def __init__(self):
super().__init__()
self.setWindowTitle("Object Detection, Distance Measurement, and Recommendation System")
self.setGeometry(100, 100, 1400, 900)  # Adjusted height for better UI balance
 # self.setWindowIcon(QIcon('icon.png'))  # Optional: Add an icon
# Focal length in pixels, to be calculated from user input in meters
self.focal_length_pixels = None
# Conversion factor based on the corrected mapping:
# 0.25m = 1471.875 px => 1m = 5887.5 px/m
self.conversion_factor = 5887.5  # px per meter (corrected from 588750)
# Main widget and layout
self.central_widget = QWidget()
self.setCentralWidget(self.central_widget)
self.main_layout = QVBoxLayout(self.central_widget)
self.main_layout.setContentsMargins(20, 20, 20, 20)
self.main_layout.setSpacing(10)
# Create UI sections
self.create_menu()
self.create_main_sections()
# Prompt user for focal length in meters upon startup
self.get_focal_length()
# Initialize camera and timer
self.cap = None
self.timer = QTimer()
self.timer.timeout.connect(self.update_frame)
```

ObjectDetectionApp: Ana uygulama penceresini tanımlayan sınıf.

“init” Metodu: Pencere başlığını ve boyutlarını ayarlar. Odak uzunluğu (focal_length_pixels) başlangıçta None olarak tanımlanır ve kullanıcı girdisi ile hesaplanır.

conversion_factor: Metre başına piksel dönüşüm faktörü. Bu değer, mesafe hesaplamalarında kullanılır.

create_menu ve create_main_sections: UI'nın menü ve ana bölümlerini oluşturur.

get_focal_length: Uygulama başlatıldığında kullanıcıdan odak uzunluğunu girmesini sağlar.

cap ve timer: Canlı kamera akışı için kamera nesnesi ve zamanlayıcı başlatılır.

EK 5.3.5.1. Menü Oluşturma

```
def create_menu(self):
menu_bar = self.menuBar()
menu_bar.setStyleSheet("""
QMenuBar {
background-color: #2c3e50;
color: white;
font-size: 16px;        }
QMenuBar::item {
background: transparent;        }
MenuBar::item:selected {
background: #34495e;        }
QMenu {
background-color: #34495e;
color: white;
font-size: 14px;        }
QMenu::item:selected {
background-color: #3d566e;        }
""")
```

```
# File menüsü
file_menu = menu_bar.addMenu('File')
# Görüntü Yükleme Eylemi
upload_action = QAction('Upload Image', self)
upload_action.triggered.connect(self.upload_image)
file_menu.addAction(upload_action)
# Odak Uzunluğu Kalibrasyonu Eylemi
calibrate_action = QAction('Calibrate Focal Length', self)
calibrate_action.triggered.connect(self.calibrate_focal_length)
file_menu.addAction(calibrate_action)
# Çıkış Eylemi
exit_action = QAction('Exit', self)
exit_action.triggered.connect(self.close)
file_menu.addAction(exit_action)
```

“create_menu” Metodu: Menü çubuğunu oluşturur ve stilini ayarlar.

File Menüsü: İçerisinde "Upload Image", "Calibrate Focal Length" ve "Exit" eylemleri bulunur.

Eylemler:

Upload Image: Kullanıcının görüntü yüklemesini sağlar.

Calibrate Focal Length: Odak uzunluğunu kalibre etmek için kullanılır.

Exit: Uygulamadan çıkışı sağlar.

EK 5.3.5.2. Ana Bölümleri Oluşturma

```
def create_main_sections(self):
# Split the main layout into two parts: Image Display and Info Section
self.top_layout = QHBoxLayout()
self.bottom_layout = QVBoxLayout()
# Image Display Section
```

```
        self.image_label = QLabel("Loading image...")
        self.image_label.setAlignment(Qt.AlignmentFlag.AlignCenter)
        self.image_label.setStyleSheet("""
        QLabel {
        border: 2px solid #555;
        background-color: #333;
        color: #fff;
        font-size: 18px;
         border-radius: 10px;
        }
        """)
        self.image_label.setScaledContents(True)  # Ensures the image scales to fit the label
        self.image_label.setMinimumSize(1000, 600)  # Increased minimum size for prominence
        self.image_label.setMaximumSize(1200, 800)  # Adjusted maximum size
        self.top_layout.addWidget(self.image_label, stretch=5,
        alignment=Qt.AlignmentFlag.AlignCenter)
        # Info Section (Information Box and Recommendation Display)
        self.info_and_recommend_layout = QVBoxLayout()
        self.info_and_recommend_layout.setSpacing(10)
        # Information Box
        self.info_box = QTextEdit()
        self.info_box.setReadOnly(True)
        self.info_box.setFont(QFont("Segoe UI", 12))
        self.info_box.setStyleSheet("""
        QTextEdit {
        background-color: #34495e;
        color: #ecf0f1;
        border: 2px solid #2c3e50;
        padding: 10px;
        font-size: 14px;
        border-radius: 10px; }
        """)
        self.info_box.setFixedHeight(200)  # Increased height for better visibility
```

```
self.info_and_recommend_layout.addWidget(self.info_box)
# Recommendation Section
self.recommendation_display = QLabel("Awaiting recommendation...")
self.recommendation_display.setAlignment(Qt.AlignmentFlag.AlignCenter)
self.recommendation_display.setFont(QFont("Segoe UI", 14))
self.recommendation_display.setStyleSheet("""
QLabel {
background-color: #ecf0f1;
color: #2c3e50;
padding: 10px;
border-radius: 10px;
border: 2px solid #34495e;      }
""")
self.recommendation_display.setFixedHeight(250)  # Reduced height to 1/4
self.info_and_recommend_layout.addWidget(self.recommendation_display)
self.top_layout.addLayout(self.info_and_recommend_layout, stretch=2)
self.main_layout.addLayout(self.top_layout, stretch=7)
# Buttons Section
self.create_buttons()
```

“create_main_sections” Metodu:

Top Layout: Yatay düzen içerisinde sol tarafta görüntü gösterim alanı (image_label), sağ tarafta bilgi kutusu (info_box) ve öneri alanı (recommendation_display) bulunur.

image_label: Yüklenen veya canlı akıştan alınan görüntüyü göstermek için kullanılır. Stil verilerek görsel olarak belirginleştirilmiştir.

info_box: Nesne tespiti ve mesafe hesaplamaları ile ilgili bilgilerin gösterildiği bir metin kutusudur. Sadece okunabilir (setReadOnly(True)) olarak ayarlanmıştır.

recommendation_display: Hesaplanan mesafeye göre kullanıcıya sunulan önerilerin gösterildiği bir QLabel'dir. Stil verilerek dikkat çekici hale getirilmiştir.

Stretch Faktörleri: Görüntü alanına daha fazla alan sağlamak için stretch=5, bilgi ve öneri alanına ise stretch=2 verilmiştir.

Buttons Section: Alt kısımda butonların yer alacağı bölüm oluşturulur.

EK 5.3.5.3. Butonları Oluşturma

```
def create_buttons(self):
# Button layout
self.button_layout = QHBoxLayout()
self.button_layout.setSpacing(20)
# Upload Image Button
self.upload_button = QPushButton("Upload Image")
self.upload_button.clicked.connect(self.upload_image)
self.upload_button.setStyleSheet(self.button_style("#27ae60"))
self.button_layout.addWidget(self.upload_button)
# Calibrate Focal Length Button
self.calibrate_button = QPushButton("Calibrate Focal Length")
self.calibrate_button.clicked.connect(self.calibrate_focal_length)
self.calibrate_button.setStyleSheet(self.button_style("#e67e22"))
self.button_layout.addWidget(self.calibrate_button)
# Start Live Camera Button
self.start_camera_button = QPushButton("Start Live Camera")
self.start_camera_button.clicked.connect(self.start_camera)
self.start_camera_button.setStyleSheet(self.button_style("#2980b9"))
self.button_layout.addWidget(self.start_camera_button)
# Stop Live Camera Button
self.stop_camera_button = QPushButton("Stop Live Camera")
self.stop_camera_button.clicked.connect(self.stop_camera)
self.stop_camera_button.setEnabled(False)
self.stop_camera_button.setStyleSheet(self.button_style("#c0392b", enabled=False))
self.button_layout.addWidget(self.stop_camera_button)
self.main_layout.addLayout(self.button_layout, stretch=1)
```

“create_buttons” Metodu:

Button Layout: Yatay düzen içerisinde dört adet buton bulunmaktadır.

Upload Image: Kullanıcının görüntü yüklemesini sağlar.

Calibrate Focal Length: Odak uzunluğunu kalibre etmek için kullanılır.

Start Live Camera: Canlı kamera akışını başlatır.

Stop Live Camera: Canlı kamera akışını durdurur.

Buton Stil Verme: button_style metoduyla her buton için renk ve stil tanımlanmıştır.

Butonların Etkinlik Durumları: "Stop Live Camera" butonu, canlı akış başlamadan önce devre dışı bırakılmıştır (setEnabled(False)).

EK 5.3.5.4. Buton Stil Tanımlama

```
def button_style(self, color, enabled=True):
"""
Defines the style for buttons.
"""
if enabled:
return f"""
QPushButton {{
 background-color: {color};
 color: white;
 border-radius: 8px;
 padding: 12px 24px;
 font-size: 16px;
 }}
QPushButton:hover {{
 background-color: #fff;
 color: {color};
```

```
border: 2px solid {color};
}}
"""
else:
return f"""
QPushButton {{
background-color: #95a5a6;
color: white;
border-radius: 8px;
padding: 12px 24px;
font-size: 16px;
}}
"""
```

“button_style” Metodu: Her buton için arka plan rengi, yazı rengi, köşe yuvarlama, padding ve font boyutu gibi stil özelliklerini tanımlar. Buton etkinleştirildiğinde ve etkin olmadığında farklı stiller uygulanır.

EK 5.3.5.5. Odak Uzunluğu Girişi

```
def get_focal_length(self):
"""
Prompts the user to input the focal length (F) in meters, multiplies it by 100, and converts it to pixels.
"""
QMessageBox.information(
self, "Focal Length Input",
"Please enter the camera's focal length (F) in meters."
)
dialog = FocalLengthDialog()
result = dialog.exec()
if result == QDialog.DialogCode.Accepted:
focal_length_str = dialog.get_focal_length()
try:
focal_length_m = float(focal_length_str)
```

```
if focal_length_m > 0:
# Multiply by 100 as per user instruction
focal_length_m *= 100
# Convert focal length from meters to pixels using the corrected mapping
self.focal_length_pixels = focal_length_m * self.conversion_factor
QMessageBox.information(
self, "Focal Length",
f"Focal length in pixels: {self.focal_length_pixels:.2f} px" )
print(f"Focal Length (pixels): {self.focal_length_pixels:.2f} px")
else:
QMessageBox.critical(
self, "Error",
"Focal length must be greater than 0. The application will now exit." )
sys.exit()
except ValueError:
QMessageBox.critical(
self, "Error",
"Invalid input for focal length. The application will now exit." )
sys.exit()
else:
QMessageBox.critical(
self, "Error",
"Focal length input was cancelled. The application will now exit.")
sys.exit()
```

“get_focal_length” Metodu: Kullanıcıdan odak uzunluğunu (focal length) metre cinsinden girmesini ister. Girilen değer 100 ile çarpılarak piksel cinsine dönüştürülür ve focal_length_pixels olarak saklanır. Geçersiz veya negatif bir değer girilirse, uygulama hata mesajı göstererek kapanır.

EK 5.3.5.6. Görüntü Yükleme ve Nesne Tespiti

```
def upload_image(self):
"""
```

```
Allows the user to upload an image for object detection.
"""
options = QFileDialog.Option.ReadOnly
file_path, _ = QFileDialog.getOpenFileName(
self, "Select Image", "", "Images (*.png *.jpg *.jpeg)", options=options)
if file_path:
image = cv2.imread(file_path)
if image is None:
QMessageBox.critical(self, "Error", "Failed to load the image.")
return
self.frame_height, self.frame_width = image.shape[:2]
detections, annotated_image = self.detect_objects(image)
if annotated_image is not None:
self.display_image(annotated_image)
self.display_info(detections)
```

“upload_image” Metodu: Kullanıcının bilgisayarından bir görüntü seçmesini sağlar. Seçilen görüntü OpenCV ile okunur ve detect_objects metoduna gönderilerek nesne tespiti yapılır. Tespit edilen nesneler işaretlenmiş görüntü display_image metoduyla gösterilir ve ilgili bilgiler display_info metoduyla bilgi kutusuna yazılır.

EK 5.3.5.7. Odak Uzunluğu Kalibrasyonu

```
def calibrate_focal_length(self):
"""
Calibration process: Upload an image with a known distance and object height.
"""
QMessageBox.information(
self, "Calibration",
"To calibrate, upload an image containing an object at a known distance with a known height." )
options = QFileDialog.Option.ReadOnly
file_path, _ = QFileDialog.getOpenFileName(
self, "Select Calibration Image", "", "Images (*.png *.jpg *.jpeg)", options=options)
```

```
        if file_path:
        image = cv2.imread(file_path)
        if image is None:
        QMessageBox.critical(self, "Error", "Failed to load the image.")
        return
        self.frame_height, self.frame_width = image.shape[:2]
        detections, annotated_image = self.detect_objects(image)
        if detections:
        # Ask user for the known distance
        distance, ok = QInputDialog.getDouble(
        self, "Calibration Distance",
        "Enter the known distance to the object (meters):",
        decimals=2, min=0.1, step=0.1)
        if not ok or distance <= 0:
        QMessageBox.warning(self, "Calibration Cancelled", "Calibration process was
cancelled.")
        return
        # Iterate through detections to find a suitable object for calibration
        for detection in detections:
        class_name = detection['class']
        confidence = detection['confidence']
        box_height = detection['height']
        # Debug: Print detection details
        print(f"Calibration Details - Class: {class_name}, Confidence: {confidence},
        Height: {box_height}")
        if (confidence > 0.7 and class_name in VEHICLE_DIMENSIONS):
        actual_height = VEHICLE_DIMENSIONS[class_name]["height"]
        # Convert normalized height to pixels
        if box_height <= 1.0:
        box_height_pixels = box_height * self.frame_height
        else:
        box_height_pixels = box_height  # Already in pixels
        # Prevent division by zero
        if box_height_pixels == 0:
```

```
QMessageBox.critical(
self, "Error", "Detected object's height is zero pixels. Cannot calibrate." )
return
# Calibration formula: F = (h' * distance) / R
self.focal_length_pixels = (box_height_pixels * distance) / actual_height
QMessageBox.information(
self, "Calibration Successful",
f"Focal length (pixels): {self.focal_length_pixels:.2f} px" )
print(f"Calibrated Focal Length: {self.focal_length_pixels:.2f} px")
return
else:
QMessageBox.warning(
self, "Calibration Error",
"No suitable object found for calibration. Please check the image.")
else:
QMessageBox.warning(
self, "Calibration Error",
"No objects detected in the image for calibration."
  )
```

“calibrate_focal_length” Metodu: Kullanıcıya, bilinen bir mesafede ve bilinen bir yüksekliğe sahip bir nesne içeren bir görüntü yüklemesi için bilgi verir. Yüklenen görüntüde nesneler tespit edilir. Kullanıcıdan nesnenin bilinen mesafesi alınır. Tespit edilen nesneler arasında uygun bir nesne bulunursa, kalibrasyon formülü F = (h' * distance) / R kullanılarak odak uzunluğu (focal_length_pixels) hesaplanır ve saklanır. Eğer uygun bir nesne bulunamazsa veya tespit edilen nesnenin yüksekliği sıfır ise, hata mesajı gösterilir.

EK 5.3.5.8. Canlı Kamera Akışı Başlatma ve Durdurma

```
def start_camera(self):
"""
Starts the live camera feed and processes frames for object detection.
"""
self.cap = cv2.VideoCapture(0)  # Default camera
if not self.cap.isOpened():
```

```
QMessageBox.critical(self, "Error", "Failed to open the camera.")
return
# Set camera resolution
self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
# Check if calibration has been done
if self.focal_length_pixels is None:
QMessageBox.information(
self, "Calibration Required",
"Please calibrate the focal length before starting the live camera.")
self.cap.release()
return
self.timer.start(30)  # Update every 30 ms (~33 FPS)
self.start_camera_button.setEnabled(False)
self.stop_camera_button.setEnabled(True)
self.stop_camera_button.setStyleSheet(self.button_style("#c0392b"))
def stop_camera(self):
"""
Stops the live camera feed and releases resources.
"""
self.timer.stop()
if self.cap:
self.cap.release()
self.start_camera_button.setEnabled(True)
self.stop_camera_button.setEnabled(False)
self.stop_camera_button.setStyleSheet(self.button_style("#c0392b", enabled=False))
self.image_label.clear()
self.image_label.setText("Loading image...")
self.info_box.clear()
self.recommendation_display.setText("Awaiting recommendation...")
```

“Start_Camera” Metodu: Bilgisayara bağlı varsayılan kamerayı (genellikle 0) açar. Kameranın çözünürlüğünü ayarlar. Odak uzunluğu kalibre edilmemişse, kullanıcıya önce

kalibrasyon yapması gerektiğini bildirir ve kamerayı kapatır. Canlı kamera akışını başlatmak için zamanlayıcı (QTimer) kullanılır ve her 30 milisaniyede bir update_frame metodu tetiklenir. "Start Live Camera" butonu devre dışı bırakılır, "Stop Live Camera" butonu etkinleştirilir.

“Stop_Camera” Metodu: Canlı kamera akışını durdurur. Kamera kaynağını serbest bırakır. "Start Live Camera" butonu yeniden etkinleştirilir, "Stop Live Camera" butonu devre dışı bırakılır. Görüntü alanı temizlenir ve varsayılan metin gösterilir. Bilgi kutusu temizlenir ve öneri alanı "Awaiting recommendation..." olarak güncellenir.

EK 5.3.5.9. Frame Güncelleme

```
def update_frame(self):
"""
Processes each frame from the live camera for object detection and distance measurement.
"""
if self.cap and self.cap.isOpened():
ret, frame = self.cap.read()
if ret:
self.frame_height, self.frame_width = frame.shape[:2]
detections, annotated_image = self.detect_objects(frame)
if annotated_image is not None:
self.display_image(annotated_image)
self.display_info(detections)
else:
QMessageBox.warning(self, "Warning", "Failed to capture image from camera.")
self.stop_camera()
```

“Update_Frame” Metodu: Her zamanlayıcı tetiklendiğinde (her 30 ms), kameradan bir frame alınır. Frame başarılı bir şekilde alındıysa, nesne tespiti yapılır ve sonuçlar görüntü ile bilgi kutusuna aktarılır. Frame alınamazsa, kullanıcıya bir uyarı mesajı gösterilir ve canlı kamera akışı durdurulur.

EK 5.3.5.10. Nesne Tespiti ve API İletişimi

```
def detect_objects(self, image):
"""
Performs object detection using the Roboflow API.
"""
try:
, buffer = cv2.imencode('.jpg', image)
base64_image = base64.b64encode(buffer).decode('utf-8')
response = requests.post(
f"{MODEL_ENDPOINT}?api_key={API_KEY}&format=json",
data=base64_image,
headers={"Content-Type": "application/x-www-form-urlencoded"})
if response.status_code == 200:
detections = response.json().get("predictions", [])
annotated_image = self.annotate_image(image.copy(), detections)
return detections, annotated_image
else:
QMessageBox.critical(
self, "API Error",
f"Object detection failed! HTTP Status: {response.status_code}\n{response.text}")
return [], image
except Exception as e:
QMessageBox.critical(
self, "Error",
f"An error occurred during object detection: {str(e)}")
return [], image
```

“Detect_Objects” Metodu: Görüntü, JPEG formatında kodlanır ve Base64 formatına dönüştürülür. Roboflow API'ye POST isteği gönderilir ve nesne tespiti yapılır. API yanıtı başarılıysa (status_code == 200), tespit edilen nesneler alınır ve annotate_image metodu ile görüntü üzerine bounding box ve etiketler çizilir. Başarısızsa, hata mesajı gösterilir ve boş

bir liste ile orijinal görüntü döndürülür. Herhangi bir istisna oluşursa, hata mesajı gösterilir ve boş liste ile orijinal görüntü döndürülür.

EK 5.3.5.11. Görüntüye "Bounding Box" ve Etiket Çizme

```
def annotate_image(self, image, detections):
"""
Draws bounding boxes and labels on detected objects.
"""
for detection in detections:
x_center = detection['x']
y_center = detection['y']
width = detection['width']
height = detection['height']
# Convert normalized coordinates to pixels
if x_center <= 1.0:
x_center *= self.frame_width
if y_center <= 1.0:
y_center *= self.frame_height
if width <= 1.0:
width *= self.frame_width
if height <= 1.0:
height *= self.frame_height
x1 = int(x_center - width / 2)
y1 = int(y_center - height / 2)
x2 = int(x_center + width / 2)
y2 = int(y_center + height / 2)
class_name = detection['class']
confidence = detection['confidence']
if confidence > 0.7:
# Bounding box çiz
cv2.rectangle(image, (x1, y1), (x2, y2), (46, 204, 113), 2)
# Etiket ekle
cv2.putText(
```

```
image, f"{class_name} ({confidence*100:.1f}%)",
(x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
0.6, (46, 204, 113), 2)
return image
```

“Annotate_Image” Metodu: Tespit edilen her nesne için, bounding box çizilir ve sınıf adı ile güven oranı yazdırılır. Normalleştirilmiş koordinatlar (0-1 aralığında) piksel cinsine dönüştürülür. Yalnızca güven oranı 0.7'nin üzerinde olan tespitler işaretlenir.

EK 5.3.5.12. Görüntüyü Gösterme

```
def display_image(self, image):
"""
Displays the image on the QLabel.
"""
rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
h, w, ch = rgb_image.shape
bytes_per_line = ch * w
qt_image = QImage(rgb_image.data, w, h, bytes_per_line,
QImage.Format.Format_RGB888)
pixmap = QPixmap.fromImage(qt_image)
scaled_pixmap = pixmap.scaled(
self.image_label.size(), Qt.AspectRatioMode.KeepAspectRatio,
Qt.TransformationMode.SmoothTransformation)
self.image_label.setPixmap(scaled_pixmap)
```

“display_image” Metodu: OpenCV'nin BGR formatındaki görüntüsü RGB formatına dönüştürülür. QImage nesnesine çevrilir ve ardından QPixmap'e dönüştürülür. QLabel'in boyutuna uygun olarak ölçeklendirilir ve QLabel üzerinde gösterilir.

EK 5.3.5.13. Bilgi Kutusunu Güncelleme ve Mesafe Hesaplama

```
def display_info(self, detections):
"""
```

```
        Displays detection information and distance calculations.
        """
        self.info_box.clear()
        recommendations = []
        if not detections:
        self.info_box.append("No objects detected.")
        self.recommendation_display.setText("Awaiting recommendation...")
         return
        for detection in detections:
        class_name = detection['class']
        confidence = detection['confidence']
        box_height = detection['height']
        # Convert normalized height to pixels
        if box_height <= 1.0:
        box_height_pixels = box_height * self.frame_height
        else:
        box_height_pixels = box_height  # Already in pixels
        if (confidence > 0.7 and class_name in VEHICLE_DIMENSIONS and
self.focal_length_pixels is not None):
        actual_height = VEHICLE_DIMENSIONS[class_name]["height"]
        distance = self.calculate_distance(actual_height, box_height_pixels)
        # Display detailed calculation steps
         self.info_box.append(f"<b>Class:</b> {class_name} ({confidence*100:.2f}%)")
         self.info_box.append(f"<b>Detected Height (h'):</b> {box_height_pixels:.2f} px")
         self.info_box.append(f"<b>Actual Height (R):</b> {actual_height:.2f} m")
         self.info_box.append(f"<b>Focal Length (F):</b> {self.focal_length_pixels:.2f} px")
        self.info_box.append(f"<b>Distance:</b> {distance:.2f} meters")
        print(f"Class: {class_name}, h' (px): {box_height_pixels:.2f}, R (m):
actual_height:.2f}, F (px): {self.focal_length_pixels:.2f}, Distance: {distance:.2f} meters")
        # Provide recommendation based on distance
         recommendation, color = self.get_recommendation(distance)
        recommendations.append((recommendation, color))
         self.info_box.append(f"<b>Recommendation:</b> {recommendation}")
         self.info_box.append("<hr>")
```

```
else:
self.info_box.append(f"Class: {class_name} ({confidence*100:.2f}%) - "
"Insufficient confidence or focal length not set.")
self.info_box.append("<hr>")
# Update recommendation display
if recommendations:
# Top recommendation takes priority
top_recommendation, color = recommendations[0]
self.recommendation_display.setText(top_recommendation)
self.recommendation_display.setStyleSheet(f"""
QLabel {{
background-color: {color};
color: #fff;
padding: 10px;
border-radius: 10px;
border: 2px solid #34495e;
}}
""")
else:
self.recommendation_display.setText("Awaiting recommendation...")
```

“Display_Info” Metodu: Bilgi kutusunu temizler ve tespit edilen nesnelerle ilgili bilgileri günceller. Her tespit için:

Sınıf adı, güven oranı ve algılanan nesnenin yüksekliği yazdırılır.

Gerçek nesne yüksekliği ve odak uzunluğu kullanılarak mesafe hesaplanır.

Hesaplanan mesafeye bağlı olarak uygun öneri belirlenir ve gösterilir.

Eğer hiç nesne tespit edilmezse, kullanıcıya "No objects detected." mesajı gösterilir.

Recommendation_Display: Hesaplanan en üstteki öneri bu alanda gösterilir ve arka plan rengi öneriye bağlı olarak değişir.

EK 5.3.5.14. Mesafe Hesaplama

```
def calculate_distance(self, actual_height, box_height_pixels):
"""
Calculates the distance to the object.
Distance = (F * R) / h'
"""
if box_height_pixels == 0:
return float('inf')
distance = (self.focal_length_pixels * actual_height) / box_height_pixels
return distance
```

“Calculate_Distance” Metodu: Nesnenin gerçek yüksekliği (actual_height) ve algılanan piksel yüksekliği (box_height_pixels) kullanılarak mesafe hesaplanır.

EK 5.3.5.15. Karşı Tedbir Öneri Sağlama

```
def get_recommendation(self, distance):
"""
Provides recommendations based on the distance category.
"""
for category, details in DISTANCE_CATEGORIES.items():
if details["min"] <= distance < details["max"]:
return details["recommendation"], details["color"]
return "No valid recommendation found.", "#95a5a6"
```

“Get_Recommendation” Metodu: Hesaplanan mesafeye göre uygun mesafe kategorisi belirlenir. Kategorilere göre belirlenen öneri ve renk döndürülür. Eğer hiçbir kategoriye uymayan bir mesafe varsa, varsayılan bir öneri ve gri renk döndürülür.

EK 5.3.5.16. Uygulama Kapatma Olayı

```
def closeEvent(self, event):
"""
Releases camera resources when the application is closed.
"""
if self.cap:
self.cap.release()
event.accept()
```

“Close Event” Metodu: Uygulama kapatıldığında, açık olan kamera kaynakları serbest bırakılır.

EK 5.4. Uygulamanın Hızlı Kullanımı

EK 5.4.1. Uygulamayı Başlatma:

Uygulamayı çalıştırdığınızda, öncelikle kullanıcıdan kameranın odak uzunluğunu (focal length) metre cinsinden girmesi istenir.

Bu değer, mesafe hesaplamalarında kullanılmak üzere piksel cinsine dönüştürülür.

EK 5.4.2. Görüntü Yükleme:

"Upload Image" butonuna tıklayarak bilgisayarınızdan bir görüntü seçebilirsiniz.

Seçilen görüntü Roboflow API aracılığıyla işlenir, nesneler tespit edilir ve görüntü üzerinde işaretlenir.

Tespit edilen nesneler, mesafe hesaplamaları ve öneriler bilgi kutusunda gösterilir.

EK 5.4.3. Odak Uzunluğu Kalibrasyonu:

"Calibrate Focal Length" butonuna tıklayarak, bilinen bir mesafede ve bilinen bir yüksekliğe sahip bir nesne içeren bir görüntü yükleyebilirsiniz.

Bu işlem, odak uzunluğunu daha doğru bir şekilde hesaplamak için kullanılır.

5.4.4 Canlı Kamera Akışı:

"Start Live Camera" butonuna tıklayarak bilgisayarınızdaki varsayılan kameradan canlı görüntü alabilirsiniz.

Canlı akış sırasında her frame işlenir, nesneler tespit edilir ve mesafe hesaplamaları yapılır.

"Stop Live Camera" butonuna tıklayarak canlı akışı durdurabilirsiniz.