DÜZCE ÜNİVERSİTESİ | LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

OTONOM ARAÇLAR İÇİN METASEZGİSEL ALGORİTMALAR KULLANARAK OLUMSUZ HAVA KOŞULLARINDA YOLO NESNE ALGILAMA PERFORMANSININ İYİLEŞTİRİLMESİ

İBRAHİM ÖZCAN

DOKTORA TEZİ
ELEKTRİK-ELEKTRONİK VE BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

DANIŞMAN
DOÇ. DR. YUSUF ALTUN

DÜZCE, 2024

T.C.
DÜZCE ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

OTONOM ARAÇLAR İÇİN METASEZGİSEL ALGORİTMALAR KULLANARAK OLUMSUZ HAVA KOŞULLARINDA YOLO NESNE ALGILAMA PERFORMANSININ İYİLEŞTİRİLMESİ

İbrahim ÖZCAN tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik–Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı'nda **DOKTORA TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Doç. Dr. Yusuf ALTUN

Düzce Üniversitesi

Eş Danışman

Dr. Öğr. Üyesi Cevahir PARLAK

Fenerbahçe Üniversitesi

Jüri Üyeleri

Doç. Dr. Yusuf ALTUN
Düzce Üniversitesi _____

Prof. Dr. Pakize ERDOĞMUŞ
Düzce Üniversitesi _____

Prof. Dr. Eyyüp GÜLBANDILAR
Eskişehir Osmangazi Üniversitesi _____

Prof. Dr. Ayhan İSTANBULLU
Balıkesir Üniversitesi _____

Dr. Öğr. Üyesi Ali ÇALIM
Bolu Abant İzzet Baysal Üniversitesi _____

Tez Savunma Tarihi: 10/07/2024

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

10 Temmuz 2024

İbrahim ÖZCAN

TEŞEKKÜR

Doktora öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Doç. Dr. Yusuf Altun'a en içten dileklerimle teşekkür ederim.

Tez çalışmam boyunca değerli katkılarını esirgemeyen çok değerli hocalarım Prof. Dr. Pakize Erdoğmuş ve Prof. Dr. Eyyüp Gülbandılar'a ve eş danışmanım Dr. Öğr. Üyesi Cevahir Parlak'a da şükranlarımı sunarım.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen, pes etme noktasına geldiğimde bana güvendiğini hissettiren sevgili eşim Hülya Özcan'a ve sabırla babalarını destekleyen oğullarım Mevlüt Yekta ve Ali Yağız Özcan'a sonsuz teşekkürlerimi sunarım.

Bu tez çalışması, Düzce Üniversitesi BAP- 2020.06.01.1060 numaralı Bilimsel Araştırma Projesiyle desteklenmiştir.

10 Temmuz 2024 **İbrahim ÖZCAN**

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

ÇİZELGE LİSTESİ

Sayfa No

KISALTMALAR

AOD	Image Dehazing Module
AP	Average Precision
ARO	Artificial Rabbit Optimizer
AWBLP	Automatic White Balance Fused with Laplace Pyramid
AWCs	Adverse Weather Conditions
BDD	Berkeley Deep Drive
BDD-IW	Berkeley Deep Drive – Inclement Weather
BOA	Balina Optimizasyon Algoritması
BoF	Bag of Freebies
BoS	Bag of Specials
CBAM	Convolutional Block Attention Module
CDnet	changedetection.net
CIOU	Complete Intersection Over Union
CLEO	Chimpanzee Leader Selection Optimization
CMCOA	Cauchy Mutation based Coyotes Optimization Algorithm
CNN	Convolutional Neural Network
CNN-PP	Convolutional Neural Network-Physics Parameter
COCO	Common Objects in Context
CRC	Colorectal Cancer
CSPNet	Cross Stage Partial Networks
DAWN	Vehicle Detection in Adverse Weather Conditions
DBOA	Değiştirilmiş Balina Optimizasyon Algoritması
DenseNet	Dense Convolutional Network
DIP	Differentiable Image Processing
DL	Deep Learning
DLIE	Deep Learning-based Image Enhancement
DÖ	Derin Öğrenme
E-ELAN	Extended-Efficient Layer Aggregation Network
ELAN	Efficient Layer Aggregation Network
Fast R-CNN	Fast Region-based Convolutional NeuralNetwork
Faster R-CNN	Faster Region-based Convolutional NeuralNetwork
FN	False Negatif
FP	False Pozitif
FPN	Feature Pyramid Network
FPS	Frame Per Second
GELAN	Generalized Efficient Layer Aggregation Network
GKO	Gri Kurt Optimizasyon
GLCM	Gray Level Co-Occurrence Matrix
GWO	Gray Wolf Optimizer
HDA	Hierarchical Data Aggregation
HoG	Histogram of Oriented Gradients
HSV	Hue Saturation Value
IA-YOLO	Image Adaptive – You Look Only Once
IDOD-YOLO	Image-Dehazing YOLO
IDP	Image Dehazing and Enhancement Processing
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
LBP	Local Binary Pattern

LeakyReLU	Leaky Rectified Linear Unit
LISA	Laboratory for Intelligent and Safe Automobiles
mAP	Mean Average Precision
Mask R-CNN	Mask Region-based Convolutional NeuralNetwork
MCO	Modified Canny Operator
MP	Maximum Pooling
MRBLSTM	Modified ReLU-based Bidirectional Long Short Term Memory
MSRCR	Multi-Scale Retinex with Color Restoration
NMS	Non-Maximum Suppression
OHK	Olumsuz Hava Koşulları
PAN	Path Aggregation Network
PGI	Programmable Gradient Information
pH	Potential of Hydrogen
PSNR	Peak Signal-to-Noise Ratio
RCNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
ResNET	Residual Neural Network
RevCol	Reversible Column Network
RF	Random Forrest
RPNs	Region Proposal Networks
RTTS	Real-Time Transportation System
SAHI	Slicing-Aided Hyper Inference
SAIP	Adaptive Image Enhancement Module
SIFT	Scale Invariant Feature Transform
SILU	Sigmoid Linear Unit
SPP-NET	Spatial Pyramid Pooling in Deep Convolutional Networks
SSD	Single Shot Dedector
SSIM	Structural Similarity
ŞLSO	Şempanze Lider Seçme Optimizasyon
TN	True Negatif
TP	True Pozitif
TRA	Three-Weather Removal Algorithm
VGG	Very Deep Convolutional Networks
YAK	Yapay Arı Kolonisi
YOLO	You Look Only Once
YTO	Yapay Tavşan Optimizasyon

SİMGELER

α	Alfa
β	Beta
δ	Sigma
θ	Teta
ω	Omega

ÖZET

OTONOM ARAÇLAR İÇİN METASEZGİSEL ALGORİTMALAR KULLANARAK OLUMSUZ HAVA KOŞULLARINDA YOLO NESNE ALGILAMA PERFORMANSININ İYİLEŞTİRİLMESİ

İbrahim ÖZCAN
Düzce Üniversitesi
Lisansüstü Eğitim Enstitüsü, Elektrik-Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi
Danışman: Doç. Dr. Yusuf ALTUN

Temmuz 2024, 118 sayfa

Otonom sürüşlerde en büyük problemlerden biri nesnenin doğru ve hızlı bir şekilde algılanamamasıdır. Özellikle olumsuz hava koşullarında (OHK) nesne algılama önemli bir problem olmaktadır. Sisli, kar fırtınalı, sağanak yağışlı, kum fırtınalı gibi hava durumları nesne algılama algoritmalarının performansını düşürmektedir. Genel olarak nesne algılama algoritmaları çift aşamalı ve tek aşamalı olmak üzere iki kategoriye ayrılır. Çift aşamalı yaklaşım potansiyel nesne bölgeleri önerir ve bu bölgelerde nesne algılama işlemi gerçekleştirilir. Tek aşamalı yaklaşım da görüntünün tüm piksellerinin tek bir seferde işlenmesi ve analiz edilmesiyle nesne algılama işlemi gerçekleştirilir. Tek aşamalı yaklaşımda nesne tespitinde, hız ön plana çıkarken, çift aşamalı yaklaşımda ise daha yüksek doğruluk ön plana çıkmaktadır. Bununla birlikte tek aşamalı yaklaşıma sahip nesne algılama algoritmalarından YOLO (You Look Only Once) ve sonra çıkan sürümleri, nesneyi doğru algılama konusunda çift aşamalı algoritmalardan daha iyi olduğu durumlar da olmaktadır. Bu tez çalışması, bir Derin Öğrenme (DÖ) algoritması olan YOLO sürüm 5, 7 ve 9'u kullanan otonom araçlar için OHK'de nesne algılamanın iyileştirilmesine odaklanmıştır. DÖ'nün başarısı, YOLO'nun bu üç versiyonunun optimizasyon ve eğitim için kullanılan hiperparametrelerinin etkili bir şekilde ayarlanmasına bağlıdır. Hiperparametrelerin optimizasyonu, genellikle manuel olarak uygulandığı için, YOLO için açık bir araştırma konusudur. Bu tez çalışmasında, YOLOv5, YOLOv7 ve YOLOv9 hiperparametrelerini optimize etmek için Gri Kurt Optimizasyon (GKO), Yapay Tavşan Optimizasyon (YTO) ve Şempanze Lider Seçim Optimizasyonu (ŞLSO) meta-sezgisel algoritmaları ayrı ayrı uygulandı. Çalışma, OHK'de DAWN (Vehicle Detection in Adverse Weather Nature) veri kümesini ve RTTS (Real-Time Transportation System) veri kümesini kullanarak optimize edilmiş hiperparametrelerin OHK'de nesne algılama üzerindeki etkisine odaklanmıştır. Sonuçlar, GKO, YTO ve ŞLSO'ya sahip en yeni YOLO modellerinin, özellikle farklı zorluklardaki hava koşullarını ve yalnızca yola ait verileri içeren DAWN veri setinde, nesne tespitini önemli ölçüde iyileştirdiğini göstermiştir. YOLO modellerinin OHK için nesne algılama konusundaki genel performansı sırasıyla YOLOv7 + YTO ile %6,146, YOLOv7 + ŞLSO ile %6,277 ve YOLOv9 + GKO ile %6,764 artmıştır.

Anahtar Sözcükler: Derin öğrenme, Meta-Sezgisel, Nesne tespiti, Otonom araçlar, YOLO

ABSTRACT

IMPROVING YOLO OBJECT DETECTION PERFORMANCE IN ADVERSE WEATHER CONDITIONS USING METAHEURISTIC ALGORITHMS FOR AUTONOMOUS VEHICLES

İbrahim ÖZCAN
Düzce University
Graduate School, Department of Electrical-Electronics and Computer Engineering
Doctoral Thesis
Supervisor: Assoc. Prof. Dr. Yusuf ALTUN

July 2024 118 pages

One of the biggest problems in autonomous driving is that the object cannot be detected accurately and quickly. Especially in adverse weather conditions (AWCs), object detection is a major problem. Weather conditions such as fog, snowstorms, thunderstorms, sandstorms, etc. degrade the performance of object detection algorithms. Object detection algorithms are generally divided into two categories: two-stage and one-stage. The two-stage approach suggests potential object regions and object detection is performed in these regions. In the single-stage approach, object detection is performed by processing and analyzing all image pixels simultaneously. While the single-stage approach emphasizes speed in object detection, the two-stage approach emphasizes higher accuracy. However, there are cases where single-stage object detection algorithms such as YOLO and its later versions are better than two-stage algorithms in detecting objects correctly. This thesis is focused on improving object detection in AWCs for autonomous vehicles using YOLO versions 5, 7, and 9, a Deep Learning (DL) algorithm. The success of YOLO depends on the effective tuning of the hyperparameters used for the optimization and training of these three versions of YOLO. Optimization of hyperparameters is an open research topic for YOLO, as it is usually implemented manually. In this thesis, Grey Wolf Optimization (GWO), Artificial Rabbit Optimization (ARO), and Chimpanzee Leader Election Optimization (CLEO) meta-heuristic algorithms were applied separately to optimize YOLOv5, YOLOv7 and YOLOv9 hyperparameters. The study focused on the impact of optimized hyperparameters on object detection in AWCs using the DAWN and the RTTS datasets. The results show that the state-of-the-art YOLO models with GWO, ARO, and CLEO significantly improve object detection, especially in the DAWN dataset, which includes weather conditions of different challenges and road-only data. The overall performance of the YOLO models on object detection for AWCs improved by 6.146% with YOLOv7 + ARO, 6.277% with YOLOv7 + CLEO, and 6.764% with YOLOv9 + GWO, respectively.

Keywords: Autonomous vehicle, Deep learning, Metaheuristics, Object detection, YOLO,

EXTENDED ABSTRACT

DEVELOPMENT OF SAFE DRIVING TECHNIQUES WITH DEEP REINFORCEMENT LEARNING IN AUTONOMOUS VEHICLES

İbrahim ÖZCAN
Düzce University
Graduate School, Department of Electrical-Electronics and Computer Engineering
Doctoral Thesis
Supervisor: Assoc. Prof. Dr. Yusuf ALTUN

July 2024, 118 pages

1. INTRODUCTION

In today's world, people generally prefer motorized vehicles to get from one place to another. The population of cities is increasing day by day and the number of motorized vehicles traveling in traffic is increasing in parallel. Although traffic rules are predetermined and there are people with a driver's license who use motor vehicles in traffic and know these rules, it is determined that 95% of traffic accidents occurring in Turkey are caused by drivers [1]. These accidents cause both material losses and loss of life. The loss of trained people and the need to import spare parts for the repair of vehicles, most of which are imported, cause serious damage to the country's economy. This is precisely where self-driving vehicles are designed to prevent human-caused accidents. The inspiration for this autonomous vehicle dream comes directly from the challenges of driving in heavy traffic. Almost everyone agrees that driving is one of the most dangerous activities. Autonomous vehicles are becoming a great necessity to relieve the stress of just driving, let alone the fatigue caused by the constant focus. If traffic fatalities can be minimized or completely eliminated by eliminating human error through automation, then the benefits of autonomous vehicles cannot even be discussed. In addition, autonomous vehicles will enable the elderly, disabled, sick, and similar people to go where they want to go without the need for another human being.

A motorized vehicle becomes self-driving thanks to software that processes and interprets the data it receives from sensors and makes decisions. This software, which performs decision-making processes, is developed not with classical software logic, but with machine learning or deep learning techniques under artificial intelligence. In classical software, inputs are given to the software and the software produces an output. It cannot know the output in advance. Therefore, classical software cannot be used in autonomous vehicles. Because this software cannot process and interpret the data coming from the

sensors in autonomous vehicles. To interpret the data and make decisions, the inputs and outputs of similar data must be trained using machine learning or deep learning techniques. In this way, the data from the sensors is analyzed and an output is obtained to decide what should be done in the autonomous vehicle.

Deep learning can be generally defined as a multi-layered structure consisting of artificial neural networks. It consists of an input, hidden, and output layer. The input layer is given the data to be processed as input. In the hidden layers, the data from the input layer, which has been made suitable for training, is processed in individual neurons and the resulting data is transferred to the output layer. The output layer gives a predicted output in the same way as the output from the last hidden layer. In this thesis, the estimated output is given as, for example, a car, since it is desired to detect the object from the camera data and find out which class it belongs to.

Object detection algorithms perform very well during daytime and clear weather conditions. However, in the case of AWCs, such as extreme sunshine, fog, snowstorms, rainstorms, or sandstorms, object detection algorithms experience noticeable performance degradation. This is also the case with the YOLO algorithm. However, thanks to the open-source nature of YOLO, its performance can be improved by modifying its structure or by using it together with optimization algorithms in such AWCs. In this thesis, the performance of object detection algorithms (YOLOv5, YOLOv7, and YOLOv9) in AWCs is investigated by using GWO, ARO, and CLEO.

2. MATERIAL AND METHODS

The most important criteria for object detection in autonomous vehicles are speed and accuracy. To achieve real-time object detection, speed takes precedence over accuracy. This is very important for safe driving. As mentioned in the introduction, algorithms with a single-pass approach are the most suitable algorithms for this task. In this thesis, among these algorithms, the YOLO algorithm, which is constantly updated, is preferred. The difficulty of object detection in AWCs is well known. To improve the performance of the YOLO algorithm in these AWCs, the hyperparameters of the algorithm are optimized. For these optimizations, GWO, ARO, and CLEO are applied separately to improve the object detection performance of the YOLO algorithm in AWCs.

YOLO : As a significant development in computer vision and object detection, the YOLO algorithm has revolutionized real-time object detection. First introduced in 2016,

YOLO stands out with its ability to quickly and accurately identify objects in complex images. Unlike traditional object detection methods, YOLO allows a single neural network to process the image as a whole. This allows YOLO to process the entire image at once, unlike methods such as R-CNN, which divide the image into regions and analyze each region separately. YOLO divides the image into a grid of SxS dimensions and estimates class probabilities and bounding boxes for each grid cell. In this study, YOLOv5, which is the most commonly used model in object detection operations of YOLO, and the newer models YOLOv7 and YOLOv9 are used.

GWO : Based on grey wolf tactics and social behaviors, GWO is a metaheuristic optimization method. A balance between exploring the whole solution space and focusing on local regions is achieved with this algorithm by using alpha, Beta, Delta, and Omega wolves to represent potential solutions within the search space [7]. Wolf positions are dynamically adapted in this process toward potential solutions, following the movement of predators tracking prey. Wolf positions are refined iteratively during the algorithm, leading to the optimal solution being reached with each iteration. Using these techniques, the GWO algorithm gradually narrows down the search space to find the most optimal solution. As a result, it is useful for optimizing a wide range of tasks.

ARO : ARO was released as a new meta-heuristic optimization algorithm in 2022 [8]. Various optimization algorithms inspired by nature are developed to find the best solution. In ARO, the process of foraging and exploring the environment of rabbits is modeled and simulated in this way. Rabbits, by nature, avoid dangerous environments and at the same time explore new environments to find the best food source. Similarly, ARO provides an algorithm where optimal solutions can exist by discovering new search areas. In addition to machine learning, ARO has been applied for optimization in various fields such as finance, engineering, and logistics.

CLEO: CLEO is one of the latest Meta-heuristic optimization algorithms released in November 2022 [9]. It is inspired by the way the chimpanzee family chooses leaders [9]. The leader must be the alpha male. However, it is observed that the relationships of this chosen leader with other males and females play an important role in the CLEO algorithm [9]. The memorization ability of chimpanzees shows an inverted U-shaped performance improvement with age. The CLEO algorithm uses an iterative approach to find the optimal solution in the population. At each iteration, the algorithm calculates the fitness value for each male and female chimpanzee, selects the male and female chimpanzee with

the highest fitness value, creates a new solution using the fitness values of the selected chimpanzees, calculates the fitness value of the new solution, replaces the new solution with the least optimal solution in the current population, and stores the current best solution. The algorithm stops when the maximum number of iterations is reached or when the best solution meets a certain criterion.

3. RESULTS AND DISCUSSIONS

3.1. PERFORMANCE OF YOLO MODELS ON DAWN DATASET

The training results of YOLOv5, YOLOv7, and YOLOv9 models with the original hyperparameters are 60.20%, 68.50%, and 68.00% respectively. Different results are seen when the training results after optimizing the hyperparameters of YOLO models with optimization algorithms are examined. For example, while CLEO provides the best performance improvement in YOLOv5 and YOLOv7, GWO is the best in YOLOv9. YOLOv5 + CLEO has a 63.90% mAP value with a 6.146% performance improvement. YOLOv7 + CLEO has 72.80% mAP with a 6.277% performance increase. YOLOv9 + GWO has 72.60% mAP with a 6.764% performance increase. Precision, Recall, F1-Score, and Accuracy values are calculated separately for 6 objects in the DAWN dataset, and different results are obtained for each model applied. When the values are analyzed according to the average of all objects, the best model for F1-Score is YOLOv7 + ARO. When we look at the Accuracy values, the highest value is also seen in the YOLOv7 + ARO model. Since the detection and positioning of the object are more important in autonomous driving, mAP values are primarily considered. Therefore, it is observed that the YOLOv7 + CLEO and YOLOv9 + GWO models are superior to other models in object detection and localization.

3.2 PERFORMANCE OF YOLO MODELS ON RTTS DATASET

When the RTTS dataset is analyzed, it is seen that not only traffic images are included in the dataset, but also images where persons are in the foreground. Using the RTTS dataset, the mAP values of the training results of the YOLOv5, YOLOv7, and YOLOv9 models with the original hyperparameters are 75.60%, 76.80%, and 78.40% respectively. It is observed that the optimization algorithms used with YOLOv5 do not affect the mAP result in YOLOv5 in any way. YOLOv7 + GWO model realized a performance improvement of 0.13%, the YOLOv7 + ARO model realized a performance improvement of 0.65%, and YOLOv7 + CLEO model realized a performance improvement of 1.04%

compared to the original YOLOv7 model. YOLOv9 + GWO model realizes a performance increase of 0.51%, the YOLOv9 + ARO model realizes a performance increase of 1.02%, and the YOLOv9 + CLEO model realizes a performance increase of 1.14% compared to the original YOLOv9 model compared to the original YOLOv7 model. YOLOv9 + GWO model realizes a performance increase of 0.51%, the YOLOv9 + ARO model realizes a performance increase of 1.02%, and YOLOv9 + CLEO model realizes a performance increase of 1.14% compared to the original YOLOv9 model. The models with the best mAP value are YOLOv9 models.

4. CONCLUSION AND OUTLOOK

In this thesis study–the object detection performances of the models, especially if the hyperparameters of YOLO models used for object detection in autonomous driving in AWCs are optimized. When the data sets are compared, especially in the DAWN data set, which contains only traffic images and only images with AWCs, an increase of more than 6% in performance is observed by optimizing the YOLO models with optimization algorithms. On the RTTS dataset, optimizing YOLO models does not show a noticeable performance increase. The fact that there is generally only one adverse weather condition (foggy) in the RTTS data set and that it does not contain only traffic data distinguishes the RTTS data set from the DAWN data set.

In the DAWN dataset, even if the YOLOv5 model is optimized with GWO, ARO or CLEO, it cannot perform better than the more recent YOLOv7, and YOLOv9. However, the YOLOv7 model, with its original hyperparameters and hyperparameters optimized with GWO, ARO, and CLEO, shows almost the same performance as YOLOv9's original and optimized models with optimization algorithms. According to these results, new YOLO models including AWCs perform well in object detection in autonomous vehicles.

1. GİRİŞ

Günümüz dünyasında insanlar bir yerden bir yere gitmek için genellikle motorlu araçları tercih etmektedirler. Şehirlerin nüfusu gün geçtikçe artmakta, buna paralel olarak trafikte dolaşan motorlu araç sayısı da benzer şekilde artmaktadır. Trafik kuralları önceden belirlenmiş ve trafikte motorlu araçları kullanan, bu kuralları bilen ehliyet sahibi insanlar olmasına rağmen, Türkiye'de meydana gelen trafik kazalarının %95'inin sürücü kaynaklı olduğu tespit edilmektedir [1]. Bu kazalar yüzünden hem maddi kayıplar hem de can kayıpları yaşanmaktadır. Yetişmiş insanların yitirilmesi ve çoğunluğu ithal olan araçların tamiri için yedek parça ithal edilmek zorunda kalınması, ülkenin ekonomisine ciddi zararlar vermektedir. Otonom yani kendi kendine gidebilen araçlar işte tam da burada, insan kaynaklı kazaların önüne geçmek için tasarlanmaktadır. Bu otonom araç rüyasının ilham kaynağı doğrudan yoğun trafikte araç kullanmanın zorluklarından gelmektedir. Hiç şüphe yok ki düzenli olarak yapılan en tehlikeli şeylerden biri araba kullanmaktır. Sadece araba sürerken yaşanılan stresin, sürekli odaklanmaktan kaynaklanan yorgunluğun bile giderilmesi için otonom araçlar büyük bir ihtiyaç haline gelmektedir. Otomasyon yoluyla insan hatasını ortadan kaldırarak trafik ölümleri en aza indirilebilirse veya tamamen ortadan kaldırılabilirse o zaman otonom araçların faydalarının tartışılması bile söz konusu olamaz. Ayrıca otonom araçlar sayesinde yaşlı, engelli, hasta ve benzeri durumda olan insanların da başka bir insana ihtiyaç duymadan, gitmek isteyecekleri yere gidebilme imkânı sunulabilecektir.

Motorlu bir aracın kendi kendine gidebilir hale gelmesi, sensörlerden aldığı verileri işleyip yorumlayan ve karar veren yazılımlar sayesindedir. Karar verme işlemlerini gerçekleştiren bu yazılımlar, klasik yazılım mantığıyla değil yapay zekanın altında yer alan makine öğrenimi veya DÖ teknikleriyle geliştirilmektedir. Otonom sistemler, sensörlerden gelen veriler gerçek zamanlı olarak işleyip yorumlama kabiliyetine duyarlıdır. Klasik yazılımlar, bu tür dinamik ve değişken verilerin anlık olarak işlenmesi ve uygun parça alma işlemlerine sahip değildir. Çünkü klasik yazılım sistemleri, belirli girişleri alır ve bu girişlere göre belirli çıktılar üretir. Önceden çıktıyı bilmesine imkân yoktur. Dolayısıyla nesne algılama da önüne gelen nesnenin ne olduğuna karar verecek bir yapıda değildir ve bu nedenle otonom araçlarda kullanılamazlar. Verilerin yorumlanıp

karar verme işleminin yapabilmesi için, önceden benzer verilerin girdi ve çıktıları, makine öğrenimi veya DÖ teknikleri kullanılarak eğitilmelidir. Böylelikle sensörlerden gelen veriler analiz edilerek, otonom araçta ne yapılması gerektiğine karar verilen bir çıktı elde edilmesi sağlanır.

DÖ genel olarak, yapay sinir ağlarından oluşan çok katmanlı bir yapıdır, şeklinde ifade edilebilir. Giriş katmanı, gizli katmanlar ve çıkış katmanından oluşur. Giriş katmanına işlenmesi istenen veriler girdi olarak verilir. Gizli katmanlarda, giriş katmanından eğitime uygun hale getirilmiş veriler ayrı ayrı nöronlarda işlenir ve işlem sonucundaki veriler çıkış katmanına aktarılır. Çıkış katmanı, son gizli katmandan gelen verileri nasıl bir çıktı verecekse o şekilde tahmini bir çıktı verir. Bu tezde, kameradan elde edilen verilerden nesnenin tespiti ve hangi sınıfa ait olduğu bulunmak istenildiği için tahmini çıktı, örneğin bir araba olarak verilmektedir.

Otonom araçlarda nesne tespiti için, DÖ tabanlı çeşitli algoritmalar geliştirilmiş ve geliştirilmeye devam etmektedir. Bu algoritmaların bazıları YOLO [10] ve versiyonları, SSD (Single Shot Dedector) [11], RCNN (Region-based Convolutional Neural Network) [12], Fast R-CNN (Fast Region-based Convolutional NeuralNetwork) [13], Faster R-CNN (Faster Region-based Convolutional NeuralNetwork) [14], Mask R-CNN (Mask Region-based Convolutional NeuralNetwork) [15] ve RetinaNet [16] isimli algoritmalardır. YOLO ve versiyonları, SSD ve RetinaNet tek aşamalı yaklaşıma sahip nesne tespit algoritmalarıdır. RCNN ve yukarıda adı geçen RCNN temelli geliştirilmiş diğer versiyonlar ise iki aşamalı yaklaşıma sahip nesne tespit algoritmalarıdır. Tek aşamalı yaklaşım, görüntünün tüm piksellerinin tek bir seferde işlenmesi ve analiz edilmesiyle nesne tespitinin gerçekleştirilmesi demektir. Çift aşamalı yaklaşım da iki aşama vardır. Birinci aşama da bölge önerileri oluşturulur yani potansiyel nesne bölgeleri belirlenir. İkinci aşamada, bu bölgelerde nesne tespit işlemi gerçekleştirilir. İki yaklaşımın da birbirine sağladığı üstünlükler söz konusudur. Tek aşamalı yaklaşımda nesne tespitinde, hız ön plana çıkarken çift aşamalı yaklaşımda ise daha yüksek doğruluk ön plana çıkmaktadır. Tez çalışmasında otonom araçlar için gerçek zamanlı nesne tespit işlemlerinde hız öncelikli tercih olacağı için tek aşamalı yaklaşıma sahip nesne tespit algoritmaları tercih edilmektedir. SSD ve RetinaNet algoritmaları, YOLO algoritması gibi sürekli güncellenmemektedir. YOLO algoritmasının birçok versiyonu bulunmakta en son YOLOv9 versiyonu yayınlanmıştır. YOLOv4 ile yapılan bir çalışma da bile nesne tespit performansı SSD ve RetinaNet algoritmalarına göre daha üstün bir performans

göstermektedir [17]. Dolayısıyla YOLOv4'ten daha güncel YOLO versiyonlarının performansları daha iyi olduğu için tez çalışmasında YOLO algoritmasının güncel versiyonları tercih edilmektedir.

Nesne tespit algoritmaları gündüz ve havanın açık yani görüntünün berrak olduğu durumlarda oldukça yüksek performans göstermektedir. Ancak OHK söz konusu olduğunda örneğin sisli, kar fırtınalı, yağmur fırtınalı veya kum fırtınalı havalarda nesne tespit algoritmalarında gözle görülür performans düşüşleri meydana gelmektedir. YOLO algoritmasında da aynı durum söz konusudur. Ancak YOLO'nun açık kaynaklı yapısı sayesinde, bu tür OHK'de, kendi yapısında değişiklikler yaparak veya optimizasyon algoritmalarıyla birlikte kullanılarak, performansı artırılabilmektedir. Bu tez çalışmasında Meta-sezgisel optimizasyon algoritmalarından GKO [18], YTO [8] ve ŞLSO [9] ile YOLOv5, YOLOv7 ve YOLOv9 nesne tespit algoritmaları kullanılarak OHK'de nesne tespit algoritmalarının performansı araştırılmaktadır.

2. LİTERATÜR TARAMASI

Nesne tespit çalışmalarının tarihi, 1960'lı yıllarda çok az sayıda görüntü işleme ve sınıflandırma yöntemleri kullanılarak basit nesnelerin tespiti ile başlamaktadır. Ancak 2010 yılından itibaren DÖ teknolojisinin gelişmeyle beraber nesne tespit algoritmaları hızlı bir şekilde gelişme göstermektedir. DÖ 2012'de AlexNet'in ImageNet yarışmasında büyük bir başarı elde etmesiyle popüler hale gelmektedir. Bu başarı, büyük veri kümelerindeki karmaşık özelliklerin tanımlanması ve nesne tespiti gibi görevlerin daha doğru bir şekilde gerçekleştirilmesi için önemli bir adım olarak görülmektedir. Son yıllarda, özellikle otonom araçlar için gerçek zamanlı nesne tespit uygulamaları, önemli bir araştırma konusu olduğu için birçok çalışmada, daha çok DÖ algoritmaları örneğin; YOLO ve versiyonları, SSD, RCNN ve versiyonları, kullanılmaktadır.

[19], [20] ve [21]'deki çalışmalar, önce görüntü iyileştirmeyi ve ardından nesne tespitini gerçekleştiren yaklaşımlar önermektedir. [19]'da TogetherNet adı verilen, OHK'de nesneleri ayırt etmek için öğrenmeyi dinamik olarak geliştirerek bu iki alt görevi birbirine bağlayan etkili ancak birleştirilmiş bir algılama paradigması önerilmektedir. TogetherNet, çok görevli bir ortak öğrenme problemini ele alır. Ortak öğrenme planının ardından, restorasyon ağı tarafından üretilen temiz özellikler, algılama ağında daha iyi nesne algılamayı öğrenmek için paylaşılabilir, böylece TogetherNet'in OHK'de algılama kapasitesini artırmasına yardımcı olur. Ortak öğrenme mimarisinin yanı sıra, TogetherNet'in özellik çıkarma ve temsil yeteneklerini geliştirmek için yeni bir Dinamik Transformatör Özellik Geliştirme modülü önerilmektedir. Hem sentetik hem de gerçek dünya veri kümeleri üzerinde yapılan kapsamlı deneyler, TogetherNet'in hem niceliksel hem de niteliksel olarak YOLO'nun bir versiyonu olan YOLOx [22]'in bazı sürümleri karşısında, daha iyi performans gösterdiğini göstermektedir.

[20]'deki çalışma sürücüsüz araçların zorlu koşullarda tespit yeteneğinin geliştirilmesinin yanı sıra trafik izleme ve araç gözetimine yönelik stratejiler sunulmaktadır. Sis, kum fırtınası ve kar koşullarının zorlu olduğu durumlarda sınıflandırma, takip yörüngeleri ve hareket hesaplamaları yapılmaktadır. Başlangıçta, yolların net olmayan görüntülerini iyileştirmek için görüntü iyileştirme yöntemleri uygulanır. Geliştirilmiş görüntüler daha sonra araçları tespit etmek için bir nesne tespit ve sınıflandırma algoritmasına tabi tutulur. Son olarak, en az yörünge hatasını elde etmek için yeni yöntemler değerlendirilmektedir. (Düzeltilmiş Optik akış/Düzeltilmiş Kalman filtresi). Ayrıca gözlemlenen nesnelerin

koordinatlarından araç sayısı, türü, yörüngelerinin (Optik akış, Kalman Filtresi, Öklid Mesafesi) takip edilmesi ve bağıl hareket hesaplaması gibi özellikler çıkarılmaktadır. Bu teknikler, özellikle kötü hava koşullarında yolların havadan görünümleri üzerinden araç tespitini, takibini ve hareketini iyileştirmeyi amaçlamaktadır. OHK'de havadan görüntüleme araçları için önerilen yöntem gerçek değerden 5 pikselden daha az hataya sahip olup, iyi sonuçlar vermektedir. Havadan görüntülenen araçların araç tespiti ve takibi, geliştirilmiş MSRCR (Multi-Scale Retinex with Color Restoration) YOLOv5s modeli kullanılarak gerçekleştirilmektedir.

[22]'deki çalışma çeşitli OHK altında enerji inşaat sahalarında nesne tespitine yönelik IDP-YOLOV9 adı verilen gelişmiş bir yaklaşım önermektedir. Bu model, IDP (Image Dehazing and Enhancement Processing) modülünü ve geliştirilmiş bir YOLOV9 nesne algılama modülünü içeren paralel bir mimariden yararlanır. Özellikle OHK'de çekilen görüntüler için, TRA (Three-Weather Removal Algorithm) modülünü ve DLIE (Deep Learning-based Image Enhancement) modülünü içeren ve birden fazla hava durumu faktörünü birlikte filtreleyerek iyileştirme sağlayan paralel bir mimari kullanılmaktadır. Daha sonra, nesne tespiti için üç katmanlı bir yönlendirme dikkat mekanizması içeren geliştirilmiş bir YOLOV9 tespit ağı modülü tanıtılmaktadır. Deneyler, IDP modülünün çeşitli OHK'nin etkisini azaltarak görüntü kalitesini önemli ölçüde artırdığını göstermektedir.

[23]'deki çalışma, araç tespitini hızlandırmak için Çapasız YOLOv4 önerilmektedir. Deneysel veri seti BDD-IW, BDD100K (Berkeley Deep Drive) [24] veri setinden belirli etiketli fotoğrafların çıkarılması ve bazılarının, önerilen yöntemin OHK'de algılama hassasiyeti ve hızı açısından pratik sonuçlarını test etmek için sislenmesiyle oluşturulmaktadır. Önerilen yöntem bu veri kümesindeki gelişmiş hedef tespit algoritmalarıyla karşılaştırılmaktadır. Deneysel sonuçlar, önerilen yöntemin orijinal YOLOv4'ten yüzde 5,8 puan daha yüksek olan %60,3 mAP'ye ulaştığını göstermektedir. Sağlamlık testi sonuçları, önerilen modelin zorlu hava koşullarında hedefleri tanıma kapasitesini önemli ölçüde geliştirdiğini ve gerçek zamanlı tespitte yüksek hassasiyete ulaştığını göstermektedir.

[25]'deki çalışmada, OHK'de nesne tespiti için hava durumuna özgü bilgileri ortadan kaldıran ve daha fazla gizli bilgiyi ortaya çıkaran IA-YOLO (ImageAdaptive–YOLO) yöntemi önerilmiştir. Spesifik olarak, CNN-PP (CNN-Physics Parameter) tarafından tahmin edilen YOLO dedektörü için OHK'yi hesaba katmak üzere DIP (Differentiable

Image Processing) modülü sunulmaktadır. Önerilen IA-YOLO yaklaşımı, görüntüleri hem normal hem de OHK'ye uyarlanabilir şekilde işleyebilir. Deneysel sonuçlar önerilen IA-YOLO yönteminin hem sisli hem de düşük ışıklı senaryolarda etkinliğini göstermektedir.

[26]'daki çalışmada gece otonom araçların OHK'den dolayı önlerindeki yol koşullarını tespit edememeleri nedeniyle, doğru tespit yapabilmek için CNN, SqueezeNet [27], VGG [28], ResNet [29] ve DenseNet [30] modelleri başta olmak üzere farklı DÖ modelleri uygulanmaktadır. Ayrıca bu modellerin performansları karşılaştırılmaktadır. Mevcut gece algılamasının mevcut sınırlaması göz önüne alındığında, bu yazıda farklı yol yüzeylerinin yansıma özellikleri araştırılmaktadır. Özelliklere göre gece veritabanları ortam aydınlatmalı ve aydınlatmasız olarak toplanır. Bu veritabanları, seçilen modellerin daha fazla sahneye daha uygun hale getirilmesi amacıyla halka açık çeşitli videolardan toplanmıştır. Ayrıca seçilen modeller, toplanan bir veritabanına dayalı olarak eğitilir. Son olarak doğrulamada, bu modellerin gece kuru, ıslak ve karlı yol yüzeyi koşullarını sınıflandırma doğruluğu %94'e kadar çıkabilmektedir.

[31]'deki çalışma sisli havalarda araç tespit teknolojisinin yetersizliğine vurgu yapmakta ve araç tespitinin doğruluğunu artırmak için yalnızca bilgisayar görüşü üzerine bir çözüm önermektedir. Farklı pratik uygulama senaryolarını içeren, ön işleme için MSRCR algoritmasını ve eğitim için YOLOv4 modelini kullanmaktadır. Veri kaynağı olarak BDD100K [24] veri seti kullanılmaktadır. Sonuç olarak gelişmiş doğruluğa sahip bir araç algılama modeli elde edilmektedir. Sonuçlar, modelin farklı derecelerdeki sisli havalarda doğruluğu arttırdığını göstermektedir.

[32]'deki çalışma da YOLO-v5'in önceden eğitilmiş mimarisini ince ayar yoluyla ayarlayarak araç tespiti için transfer öğrenme kullanılmaktadır. Transfer öğrenimi konseptini kullanmak için, yazarlar tarafından sıkışık trafik düzenlerinin resim ve videolarından oluşan kapsamlı veri setleri toplanmaktadır. Bu veri kümeleri, örneğin yüksek ve düşük yoğunluklu trafik düzenleri, tıkanmalar ve farklı hava koşulları gibi çeşitli niteliklere işaret edilerek daha kapsamlı hale getirilmektedir. Toplanan bu veri kümelerinin tümü manuel olarak açıklanmaktadır. Veri kümeleri aracılığıyla önceden eğitilmiş ağa ince ayar yaparak, önerilen YOLO-v5, algılama doğruluğu ve yürütme süresi açısından diğer birçok geleneksel araç algılama yöntemine üstünlük sağlamaktadır. PKU [33], COCO [34] ve DAWN [35] veri kümeleri üzerinde gerçekleştirilen ayrıntılı simülasyonlar, önerilen yöntemin çeşitli zorlu durumlarda etkinliğini göstermektedir.

[36]’daki çalışma da YOLO'nun çalışma sırasındaki son sürümü olan YOLOx'un yağmur, sis, kar ve kum fırtınası gibi kötü hava koşullarındaki araçları tespit etmek için kullanımını araştırmaktadır. Model, dört kötü hava koşulu, farklı aydınlatma, arka plan ve bir çerçevedeki araç sayısını içeren görüntüleri içeren, halka açık DAWN karşılaştırma veri seti üzerinde test edilmektedir. Modelin etkinliği kesinlik, geri çağırma ve mAP açısından değerlendirilir. Sonuçlar, YOLOx-s'nin YOLOx-m ve YOLOx-l varyantlarına göre daha iyi performans gösterdiğini ortaya koymaktadır. Modellerin performansı kar ve sisli havalarda yağmurlu hava kum fırtınalarına göre daha iyidir. Diğer deneyler, çok ölçekli retinex kullanılarak görüntü kalitesinin arttırılmasının YOLOx performansını iyileştirdiğini göstermektedir. DAWN veri setindeki hava durumları ayrı ayrı ele alınıp eğitilmekte ve nesne olarak sadece araçları içeren veriler üzerinde eğitim gerçekleştirilmektedir.

[37]’deki çalışma da kamera tabanlı algılama doğruluğunu iyileştirmeye çalışılmaktadır. Çeşitli sert hava durumu veri kümelerinden birleştirilmiş verileri kullanarak transfer öğrenimi yoluyla OHK’de YOLOv8 tabanlı nesne algılamanın iyileştirilmesi önerilmektedir. İki başarılı açık kaynaklı veri seti (ACDC [38] ve DAWN [35]) ve bunların birleştirilmiş veri seti, sert hava koşullarında yoldaki birincil nesneleri tespit etmek için kullanılmaktadır. Bireysel veri kümeleri, bunların birleştirilmiş versiyonları ve bu veri kümelerinin çeşitli alt kümeleri üzerindeki eğitimlerden özelliklerine göre bir dizi eğitim ağırlıkları toplanmaktadır. Daha önce bahsedilen veri kümeleri ve bunların alt kümeleri üzerindeki algılama performansı değerlendirilerek eğitim ağırlıkları arasında bir karşılaştırma da yapılmaktadır. Değerlendirme, eğitim için özel veri kümelerinin kullanılmasının, YOLOv8 temel ağırlıklarına kıyasla tespit performansını artırdığını ortaya çıkarmaktadır. Ayrıca, özelliğe bağlı veri birleştirme tekniği aracılığıyla daha fazla görüntünün kullanılması, nesne algılama performansını istikrarlı bir şekilde artırmaktadır.

[39]’daki araştırmada, sis, toz ve kum fırtınası, karlı ve yağmurlu hava gibi birden fazla hava senaryosunda hem gündüz hem de gece bir olay yerindeki araçların tespit edilmesi ele alınmaktadır. Önerilen mimari, SPP-NET (Spatial Pyramid Pooling in Deep Convolutional Networks) katmanı ve azaltılmış Toplu normalizasyon katmanları ile değiştirilmiş temel mimari olarak CSPDarknet53'ü kullanır. Ayrıca çalışma da DAWN Veri Kümesi Ton, Doygunluk, Pozlama, Parlaklık, Karanlık, Bulanıklık ve Gürültü gibi farklı tekniklerle zenginleştirilmektedir.

[40]’daki çalışma, OHK'de otonom araç tanımlama için MRBLSTM (Modified ReLU-

based Bidirectional Long Short Term Memory) içeren bir DÖ modeli önermektedir. Temel olarak altı aşamadan oluşur. Önerilen sistem verileri için kamuya açık veri ayarlarından toplar. Daha sonra Retinex yazılımları kullanılarak veriler üzerinde ön işleme işlemi olarak görüntü onarımı devam eder. Daha sonra, MCO (Modified Canny Operator) çalıştırması, iki taraflı bir filtre ve Otsu'nun yaklaşmasıyla geri yüklenen görüntülerdeki kişileri korur. Daha sonra HoG, HaaR benzeri özellikler, LBP (Local Binary Pattern) ve SIFT (Scale Invariant Feature Transform) için özellik çıkarma modelleri ile özellikler, kenar algılamalı görüntülerden çıkarılır. Optimum özellikler, Cauchy'nin içerdiği Coyotes Optimizasyon Algoritması (CMCOA) kullanılarak özelliklerin boyutlarını azaltmak için düzeltilen özelliklerden seçilir. Son olarak MRBLSTM aracılığıyla araçlar tespit edilir.

[41]'deki araştırmada, otonom sürüşün karmaşık sahnelerinde çok ölçekli nesne algılamada kaçırılan algılama ve hatalı tanıma sorunlarını çözmek için YOLOX'a dayalı çok ölçekli bir nesne algılama modeli önerilmektedir. Nesnenin kritik özellik bilgisine odaklanabilen YOLOX modeline önerilen Gruplandırılmış CBAM (Convolutional Block Attention Module) eklenmektedir. Daha fazla anlamsal bilgi elde etmek ve farklı ölçeklerde nesne algısını geliştirmek için nesne-bağlamsal özellik birleştirme modülü önerilmektedir. KITTI [42] veri seti ve BDD100K [43] veri seti üzerinde karşılaştırmalı deneyler yapılmaktadır. Önerilen modelin yüksek mAP değerine sahip olması, modelin geniş uygulanabilirliğe sahip olduğunu ve otonom sürüş senaryolarında nesnelerin tanınmasına ilişkin tespit gereksinimlerini karşılayabildiğini göstermektedir.

[44]'deki çalışmada, araç tespiti ve sınıflandırmasında DÖ tekniklerinin uygulanmasıyla ilişkili bazı önemli büyüme, başarı ve dezavantajların kapsamlı bir araştırması sunulmaktadır. Araç tespiti ve sınıflandırmasında DÖ tekniklerinin ayrıntılı analizi ve araç tespiti ve sınıflandırmasında bazı önemli tespit ve sınıflandırma uygulamalarının gözden geçirilmesi, bunların zorluklarının derinlemesine analizi ve son yıllarda umut verici teknik gelişmeler ele alınmaktadır.

[45]'teki çalışmada, farklı parlaklıktaki hava görüntülerinden araç trafik parametrelerinin çıkarılması için bir YOLOX-IM-DeepSort modeli önerilmektedir. Veri seti, eğitim ve doğrulama verilerinin kalitesini iyileştiren HSV (Hue Saturation Value) alanı veri geliştirme, matris dönüştürme işlemi ve veri geliştirme için SAHI (Slicing-Aided Hyper Inference) algoritması ile işlenmektedir. VisDrone2021 veri seti üzerindeki ablasyon deneylerinin sonuçları, temel YOLOX-s modeliyle karşılaştırıldığında, geliştirilen

YOLOX-IM'nin %67,14 hacim azalması ve mAP50'de %8,13 artış (mAP50'de %3,9, %5,4 ve 8,4) göstermektedir. Ayrıca YOLOX-IM-DeepSort'un nesne algılama ve nesne izleme performansını doğrulamak için VisDrone2021 veri seti benimsenmiş ve deneysel sonuçlar, çeşitli parlaklık seviyelerinde ortalama araç algılama doğruluğunun %85,00 ve ortalama araç takip doğruluğunun %71,30 olduğunu göstermektedir. Her ikisi de ortak modelden daha iyidir. Ancak, trafik parametresi çıkarım modelinin iyi performansının tek başına ortak veri kümelerinde gösterilemeyeceği, bunun bir ölçümden yoksun olacağı ve modelin yorumlana bilirlikten yoksun olacağı ileri sürülmektedir.

[46]'da, kumlu havalarda farklı aktivasyon fonksiyonlarının performansını değerlendirmek için YOLOv5 ve YOLOv7 mimarileri ele alınmaktadır. Deneylerde üç aktivasyon fonksiyonu hedeflenmektedir: Sigmoid Doğrusal Birim (SiLU), Düzeltilmiş Doğrusal Birim (ReLU) ve Sızıntılı Düzeltilmiş Doğrusal Birim (LeakyReLU). Performanslarını değerlendirmek için kullanılan ölçümler hassasiyet, geri çağırma ve mAP'dir. Yalnızca kumlu görüntüler seçilmesine rağmen, çeşitli hava koşullarını içeren DAWN veri kümesi kullanılmaktadır.

[47]'deki makale, geleneksel Faster R-CNN yapısına birkaç RPNs (Region Proposal Networks) dahil ederek karayolundaki araç tespitine yönelik yeni bir yöntem önermektedir. Önerilen strateji, mevcut çalışmada çeşitli RPN'lerin devreye girmesi nedeniyle değişen büyüklükteki araçları daha doğru bir şekilde tespit etmektedir. Sonuçlar, önerilen stratejinin hem iyi hem de zorlu hava koşullarında değişen büyüklükteki araçları tespit etmede geleneksel Faster R-CNN'nin yanı sıra DAWN, CDNet 2014 ve LISA [48] veri kümelerindeki benzer diğer son teknoloji yöntemlerden daha iyi performans gösterdiğini göstermektedir.

[49]'daki makale, ilk önce üç aşamadan oluşan bir görünürlük geliştirme şeması önermektedir: aydınlatma geliştirme, yansıma bileşeni geliştirme ve performansı artırmak için doğrusal ağırlıklı füzyon. Ardından, çok ölçekli derin evrişimli bir sinir ağı kullanarak sağlam bir araç algılama ve izleme yaklaşımı sunmaktadır. Geleneksel Gauss karışımı olasılık hipotezi yoğunluk filtresi tabanlı izleyici, tespitten ize ve izden ize ilişkilere ayrılan HDA (Hierarchical Data Aggregation) ile birlikte kullanılır. Burada, her aşamanın maliyet matrisi, kaçırılan tespitten kaynaklanan kayıp izleri telafi etmek için Macar algoritması kullanılarak çözülmektedir. Hızlı yürütme için görsel özellik bilgisi olmadan HDA'da yalnızca algılama bilgileri (yani algılama puanlarına sahip sınırlayıcı kutular) kullanılır. Ayrıca, otonom araçların OHK'deki uygulamalarına ilişkin

araştırmalar için tasarlanmış, DAWN adı verilen yeni bir kıyaslama veri kümesi de tanıtılmaktadır. Farklı türdeki OHK'yle toplanan gerçek dünya görüntülerinden oluşur. Önerilen yöntem DAWN, KITTI ve MS-COCO veri setleri üzerinde test edilmekte ve 21 araç dedektörüyle karşılaştırılmaktadır. Deneysel sonuçlar, OHK'de en son teknoloji araç tespit ve takip yaklaşımlarından daha iyi performans göstermesi, önerilen yöntemin etkinliğini doğrulamaktadır.

Bu çalışmada [50], düşük ışıklı sisli trafik ortamları için görüntü buğu giderme ve görüntü iyileştirmenin ortak öğrenilmesiyle bir nesne algılama algoritması IDOD-YOLOV7 (Image-Dehazing-YOLOV7) önerilmektedir. IDOD-YOLOV7 algoritması, çeşitli gelişmiş sis giderme ve nesne algılama algoritmalarıyla karşılaştırılır. Görüntü bulanıklaştırma modülü AOD ve buğu giderme modülü SAIP, gerçek düşük ışıklı trafik görüntü veri kümeleri üzerinde nesnel değerlendirme metrikleri (PSNR (Peak Signal to Noise Ratio) , SSIM (Structural Similarity) ve öznel değerlendirme yöntemleri kullanılarak değerlendirilir. Deneysel sonuçlar, düşük ışıklı sis trafiği ortamlarında görüntü buğu giderme için buğu giderme + iyileştirme yönteminin (AOD + SAIP) üstünlüğünü, sağlamlığını ve etkinliğini göstermektedir.

[51]'deki makale de nesne algılamada tespit hassasiyetini artırmak için YOLOv5 modelinin performansını artırmak amacıyla yeni bir optimizasyon algoritması sunulmaktadır. İlk olarak, GKO algoritmasının avlanma davranışının iyileştirilmesi ve bunu balina optimizasyon algoritmasına (BOA) dahil edilmesiyle, değiştirilmiş bir balina optimizasyon algoritması (DBOA) önerilmektedir. DBOA, GKO veya BOA'nın avlanma dalını seçmek amacıyla pH'ı hesaplamak için popülasyonun konsantrasyon oranından yararlanır. Altı kıyaslama işlevi tarafından test edilen DBOA'nın daha iyi küresel arama yeteneği ve kararlılığa sahip olduğu kanıtlanmıştır. İkinci olarak, YOLOv5'teki C3 modülü, G-C3 ile değiştirilir ve ekstra bir algılama kafası eklenir, böylece yüksek derecede optimize edilebilir bir algılama G-YOLO ağı oluşturulur. Kendi kendine oluşturulan veri setine dayanarak, G-YOLO modelindeki 12 başlangıç hiperparametresi, bileşik göstergelerin skor uygunluk fonksiyonu kullanılarak DBOA tarafından optimize edilir, böylece son hiperparametreler optimize edilir ve balina optimizasyonu G-YOLO (BOA-GYOLO) modeli oluşturulur. YOLOv5s modeliyle karşılaştırıldığında, genel mAP %1,7, yayaların mAP'si %2,6 ve bisikletçilerin mAP'si %2,3 artmaktadır.

[52]'deki makale, mevcut hava koşullarını tahmin eden pus ve düşük görünürlük nedeniyle kötü hava koşullarında algılama performansını iyileştirmek için pus seviyesini

kullanarak bulanıklığı ortadan kaldıran bir görüntü bulanıklaştırma ağı önermektedir. Termal görüntüyle birleştirildiğinde, geri yüklenen görüntü, sırasıyla hareketli hedefleri bağımsız olarak algılayan ve geç füzyon kullanarak nesne algılama performansını artıran iki YOLO nesne dedektörüne atanır. Önerilen model, mevcut görüntü giderme modelleriyle karşılaştırıldığında gelişmiş bulanıklık giderme performansı göstermektedir ve otonom sürüş için sisli çekilen görüntülerin normal görüntülere döndürülebileceğini kanıtlamaktadır. Önerilen görüntü geri yüklenen RGB görüntüsünün termal görüntülerle birleştirilmesiyle, mevcut görüntü giderme tekniklerini kullanan modellerle karşılaştırıldığında, algılama doğruluğunu %22'ye kadar artırmaktadır.

[53]'deki çalışma, nesne algılama algoritmalarının hiperparametrelerinin optimizasyonu için önerilen ilk yöntemi önermektedir. YOLOv5 ve YAK (Yapay Arı Kolonisi) optimizasyon algoritması kullanılmaktadır. Üzerinde çalışılan konu ise CRC (Colorectal Cancer)'in öncüsü olan poliplerin tespit oranının artırılmasıdır. Deneysel çalışmalar YAK algoritmasının YOLOv5 algoritmasını başarıyla optimize ettiğini ve orijinal YOLOv5 algoritmasından çok daha yüksek doğruluk sunduğunu göstermektedir. Önerilen yöntemin, gerçek zamanlı polip tespitindeki yüksek performansıyla, hız ve doğruluk açısından literatürdeki mevcut yöntemlerden ileride olduğunu belirtilmektedir.

Literatür çalışmalarında, nesne tespit işlemleri için farklı yöntemler önerilmiş özellikle OHK'ye yönelik nesne tespite yönelik çalışmalar gerçekleştirildiği görülmektedir. Yapılan çalışmalarda otonom araçlarda OHK'de nesne tespit işlemleri çalışmaları yeterli sayıda değildir. Özellikle, bu çalışmalarda YOLOv5'in sıklıkla kullanıldığı, YOLOv7'nin çok az ve en son çıkan YOLOv9'un neredeyse hiç kullanılmadığı gözlemlenmektedir. Bu tez çalışmasındaki amaç, OHK'de nesne tespit performansını artırmak olduğu için, YOLOv5, YOLOv7 ve YOLOv9 ile meta-sezgisel algoritmalarından GKO, YTO ve ŞLSO birlikle kullanılmaktadır. Sonuçlara bakıldığında güncel YOLO versiyonlarıyla birlikte kullanılan optimizasyon algoritmalarının gözle görülür bir performans artışı sağladığı gözlemlenmektedir.

3. VERİ KÜMELERİ VE UYGULANAN ALGORİTMALAR

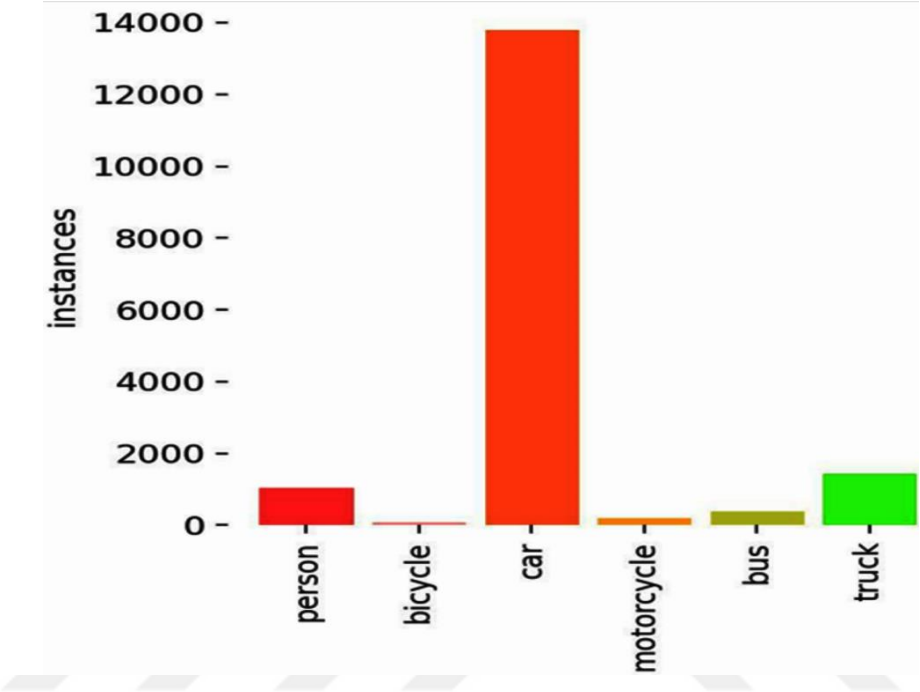
Otonom araçlarda nesne tespiti için en önemli kriter hız ve doğruluktur. Gerçek zamanlı nesne tespiti elde etmek için hız doğru tespitten öncelikli olmaktadır. Güvenli bir sürüş için bu oldukça önemlidir. Giriş bölümünde de bahsedildiği üzere tek aşamalı yaklaşıma sahip algoritmalar bu iş için en uygun algoritmalardır. Bu tez çalışmasında ise bu algoritmalar arasında sürekli güncel versiyonları çıkan YOLO algoritması tercih edilmektedir. OHK'de nesne tespit işlemlerinin zorluğu bilinmektedir. YOLO algoritmasının, bu OHK'de performansını artırmak için, algoritmanın hiper parametreleri optimize edilmektedir. Bu optimize işlemleri için GKO, YTO ve ŞLSO ayrı ayrı uygulanarak YOLO algoritmasının OHK'deki nesne tespit performansının artırılması hedeflenmektedir. Bu tezde OHK olarak kullanılan veri setleri DAWN ve RTTS veri setidir.

3.1. VERİ KÜMELERİ

Literatürde veri seti olarak benzer nesne etiketleme türlerine sahip DAWN ve RTTS veri setleri tercih edilmektedir. DAWN veri seti yalnızca yol görüntülerini ve tüm OHK'yi içeren veri içeriğine sahiptir. RTTS ise sadece sisli ve normal görsellerden oluşan veri içeriğine sahip. Bu iki veri seti uygulanan yöntemlerin performansını karşılaştırmak için kullanılır.

DAWN: Olumsuz Hava Durumu Doğa Veri Setinde Araç Tespiti: Çok az sayıda veri seti, sentetik hava durumu ve gerçek dünya görüntülerinin bir kombinasyonu aracılığıyla OHK'yi ele alır [35]. Bu eksikliğin üstesinden gelmek için Mourad A. Kenk ve Mahmoud Hassaballah, 2020 yılında araç tespitinde kullanılacak çeşitli OHK'yi içeren DAWN adlı bir veri kümesini tanıtmaktadırlar [35]. DAWN veri seti, farklı hava koşullarında kaydedilen yüksek çözünürlüklü görüntü ve videoların bir koleksiyonudur [35]. Kentsel ve kırsal ortamlardaki yollar, kavşaklar, yaya geçitleri ve trafik işaretleri veri setine dahil edilmektedir. Nesne tespiti, sınıflandırma ve izleme, veri kümesinin etiketli örnekleri kullanılarak gerçekleştirilebilir. DAWN veri seti, geniş kapsamı ve çeşitliliği sayesinde araştırmacılara ve geliştiricilere otonom sürüş algoritmalarını test etme ve geliştirme fırsatı sunuyor. Bu veri seti, DÖ modellerinin eğitimi ve performans değerlendirmesi için yaygın olarak kullanılmaktadır. Bu çalışmada OHK'deki araçların tespiti için DAWN

veri setinden yararlanılmaktadır. DAWN, OHK için özel olarak oluşturulmuş bir veri kümesidir. Bu araştırmada DAWN veri setinin Roboflow web sitesinden elde edilen YOLO için özelleştirilmiş versiyonu kullanılmaktadır [35]. Şekil 3.1'de DAWN veri setindeki nesnelerin etiketleme oranları gösterilmektedir. Etiketlenen nesneler kişi, bisiklet, araba, motosiklet, otobüs ve kamyondan oluşmaktadır.

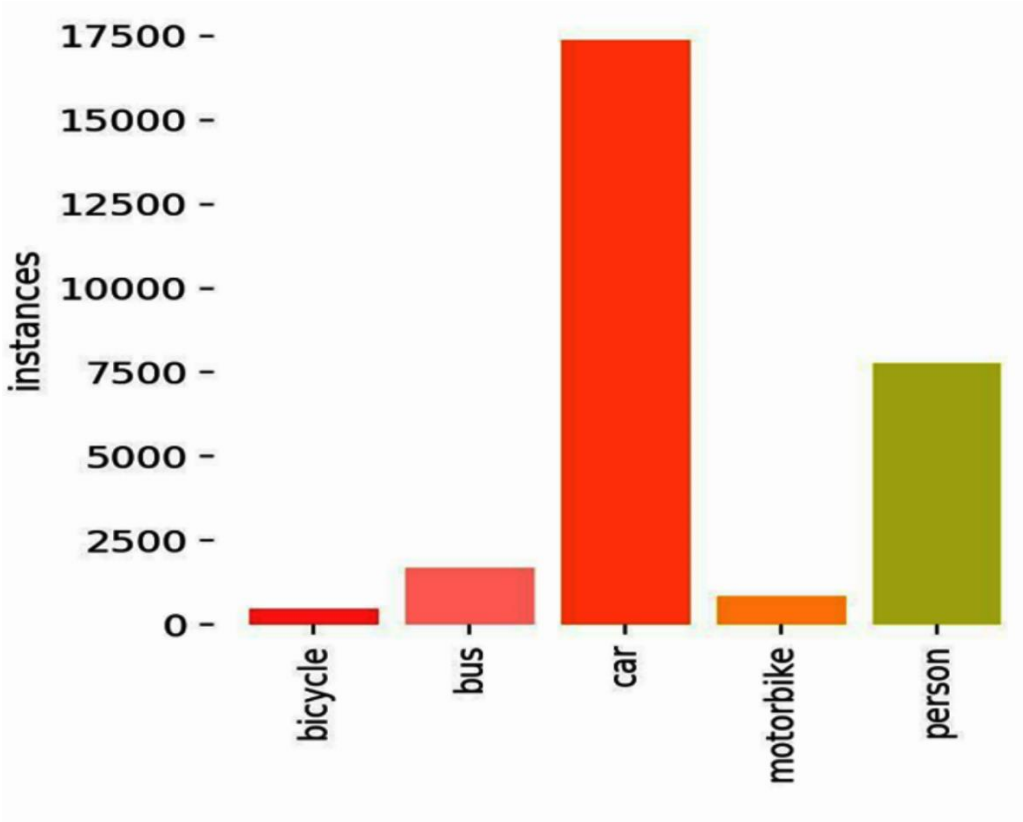


Şekil 3.1. DAWN veri kümesindeki etiketli nesnelerin örnek sayıları.

Bu veri seti DÖ yöntemiyle eğitilecek 2053 eğitim, 197 doğrulama ve 99 test verisi olarak kategorize edilmektedir. Bu veri setinde sis, kar, yağmur ve kum fırtınası görüntüleri rastgele seçilmektedir. En verimli hiperparametrelerin belirlenmesine yardımcı olmak için 197 doğrulama görüntüsü kullanılır. Nesne tespitinin etkinliğini değerlendirmek için eğitime hiç girmemiş 99 test görüntüsünden oluşan bir set kullanılmaktadır. Şekil 3.2'de DAWN veri setinin bazı görüntüleri gösterilmektedir.

Şekil 3.2. DAWN veri seti örnek görselleri.

Gerçek Zamanlı Ulaşım Sistemi (RTTS) Veri Kümesi: RTTS, puslu koşullarda en büyük açıklamalı algılama veri kümesidir [54]. Çoğunlukla trafik sahnelerinde olmak üzere 4322 adet gerçek dünyaya ait puslu görüntü içerir. Kişi, bisiklet, motosiklet, otobüs ve araba olmak üzere beş kategori bulunmaktadır. 41203 sınırlayıcı kutular etiketlenmektedir [55]. Toplu taşıma sistemleri RTTS veri seti yardımıyla izlenmekte ve iyileştirilmektedir. Bu veri kümesi genellikle otobüsler, trenler ve diğer toplu taşıma araçları için konum, hız, yön ve durak bilgilerini içermektedir. GPS verileri, sensör verileri ve yolcu sayıları gibi çeşitli veri türlerini içermektedir. RTTS veri seti, daha doğru tahminler yapmak, trafik akışını optimize etmek ve yolcu bilgi sistemlerini iyileştirmek için gerçek zamanlı ulaşım verilerini kullanmayı amaçlamaktadır. Bu çalışmada eğitim için kullanılan RTTS veri seti, Roboflow web sitesindeki YOLO algoritmaları için üretilen veri setidir [56]. Bu veri seti DÖ yöntemiyle eğitilecek 3026 eğitim, 864 doğrulama ve 432 test verisi olarak kategorize edilmektedir. Şekil 3.3'te RTTS veri kümesindeki nesnelerin etiketleme oranları gösterilmektedir. Şekil 3.4'te RTTS veri kümesinin bazı görüntüleri gösterilmektedir. Ayrıca önerilen yaklaşımın etkileri açısından normal hava koşullarını da içeren RTTS veri seti, OHK içeren DAWN veri seti ile karşılaştırma amacıyla da kullanılmaktadır.



Şekil 3.3. RTTS veri kümesindeki etiketli nesnelerin örnek sayıları.

Şekil 3.4. RTTS veri kümesi örnek görüntüleri.

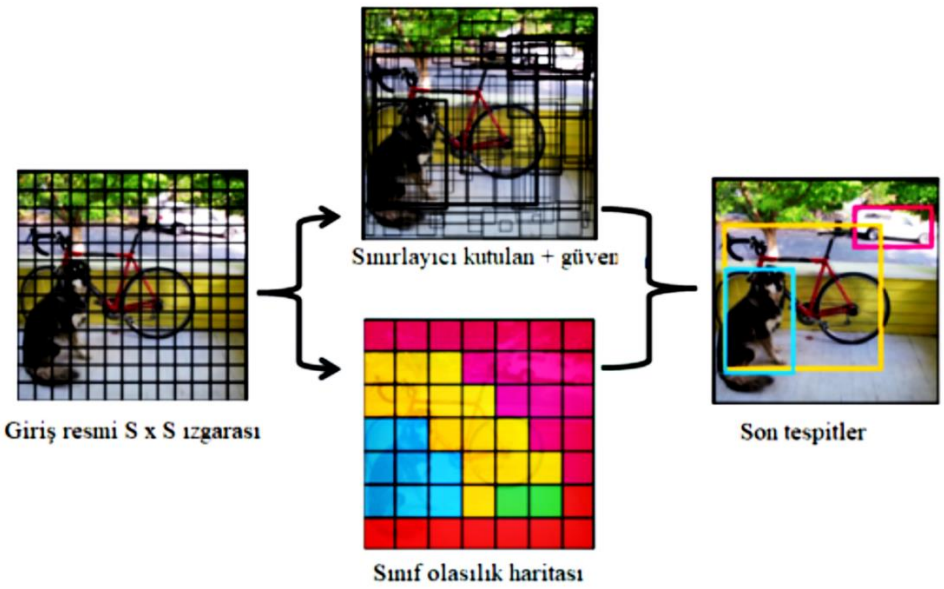
3.2. UYGULANAN ALGORİTMALAR

Görüntü işleme yapılırken nesne takibinde diğer algoritmalar görüntünün tamamı üzerinde işlem yaparken YOLO mimarisi görüntüyü bölgelere ayırarak bu bölgelerdeki nesneleri içerisine alan kutular çizer. Bu kutulara Bounding Box denir. Her bölgede nesne bulunma olasılığının hesabını yapar. Bunun yanında Güven Skoru hesaplar. Güven Skoru diye isimlendirilen değer her bir bounding box içerisinde bulunan nesnenin yüzde kaç tahmin edilen nesnenin benzeri olduğunu söylemesidir. Tabii tek neden görüntüyü bölgelere ayırarak çalışması değildir. R-CNN gibi nesne takip algoritmaları nesne bulunması muhtemel alanları belirleyip daha sonra ayrı ayrı CNN sınıflandırıcılarını kullanmaktadır. Bu işlem iki ayrı aşamadan geçtiği için görüntü üzerindeki işlem sayısı artmakta dolayısıyla YOLO bu durumda R-CNN'e göre daha hızlı olmaktadır. YOLO görüntüyü tek seferde nöral ağdan geçirmektedir. Bu tezde nesne tespiti için YOLOv5, YOLOv7 ve YOLOv9 DÖ algoritmaları tercih edilmektedir. YOLOv5 ile ilgili birçok çalışmanın olduğu görülmektedir. YOLOv5, OHK'deki diğer iki yeni algoritmanın performansını karşılaştırmak için seçilmektedir. YOLO modellerinin hiperparametrelerinin ilk değerleri COCO verisetine göre ayarlanmaktadır. Özellikle OHK'de YOLO modellerinin bu değerlerle nesne algılamada performans düşüklüğü gösterdiği görülmektedir. Bu YOLO modellerinin performansını iyileştirmek için hiperparametrelerin değerlerini optimize etmek gerekmektedir. Bu optimize işlemleri için optimizasyon algoritmaları kullanılmaktadır. Bu görev için çeşitli optimizasyon

algoritmaları kullanılabilmektedir. Bu çalışmada, nesne tespit performansını artırmak için GKO'yu ve performans karşılaştırması yapmak için ise daha yeni ve nesne tespiti özelinde başarılı sonuçlar elde edilen YTO ve ŞLSO meta-sezgisel algoritmaları seçilmektedir.

3.2.1. You Look Only Once (YOLO)

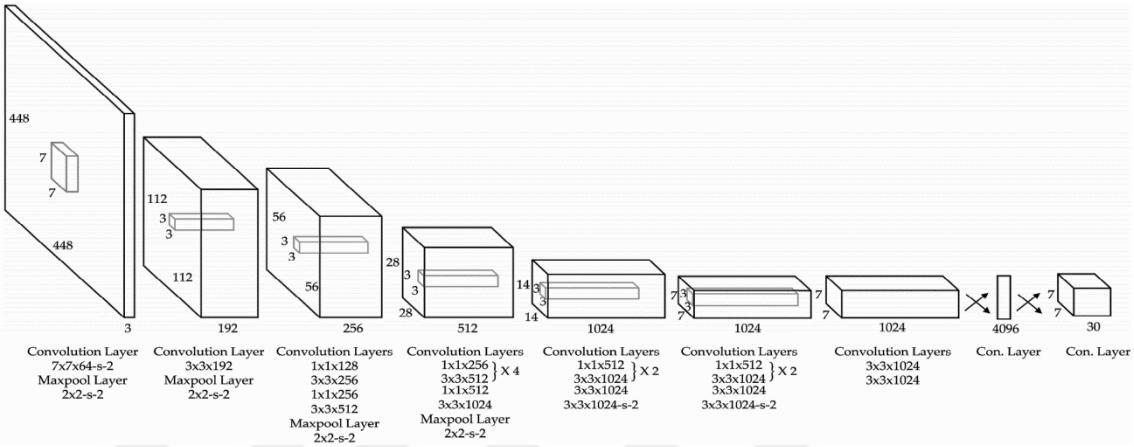
Bilgisayarla görme ve nesne tespiti alanında önemli bir gelişme olarak, YOLO algoritması, gerçek zamanlı nesne tespiti konusunda devrim yaratmıştır. İlk olarak 2016 yılında tanıtılan YOLO [57], karmaşık görüntülerde nesneleri hızlı ve doğru bir şekilde tanımlama yeteneği ile öne çıkmaktadır. YOLO, geleneksel nesne tespiti yöntemlerinden farklı olarak, tek bir nöral ağın görüntüyü bir bütün olarak işlemesini sağlar. Bu, görüntüyü bölgelere ayırıp her bir bölgeyi ayrı ayrı analiz eden R-CNN gibi yöntemlerin aksine, YOLO'nun tüm görüntüyü tek seferde işlemesine olanak tanır. YOLO, görüntüyü SxS boyutlarında bir ızgaraya böler ve her bir ızgara hücresi için sınıf olasılıkları ve sınırlayıcı kutular (bounding boxes) tahmin eder. Şekil 3.5'te bu durum görsel olarak gösterilmektedir.



Şekil 3.5. YOLO modeli [57].

YOLO'nun mimarisi, geleneksel CNN'yi temel alır. Şekil 3.6'da YOLO mimarisi yer almaktadır. Şekil incelendiğinde bu mimarinin evrişim katmanlarından oluştuğu görülmektedir. 448x448 boyutunda renkli bir görüntü, giriş olarak 448x448*3 boyutunda bir tensör sağlamaktadır. YOLO'nun evrişim katmanları, özellik çıkarma işlemlerini gerçekleştirmek için kullanılır. Evrişim katmanları, görüntüdeki kenarlar, köşeler ve daha

karmaşık desenler gibi temel özellikleri çıkarır. Bu katmanlar, filtreler ve çekirdekler (kernels) kullanarak giriş görüntüsünden özellik haritaları oluşturur. İlk Evrişim Katmanı 7x7 ve büyük bir adım uzunluğuna yani 2 değerine sahiptir. Bu katman, giriş görüntüsünü küçültür ve ilk özellik haritalarını çıkarır. Ara katmanlar daha küçük çekirdek boyutlarına



Şekil 3.6. YOLO mimarisi [58].

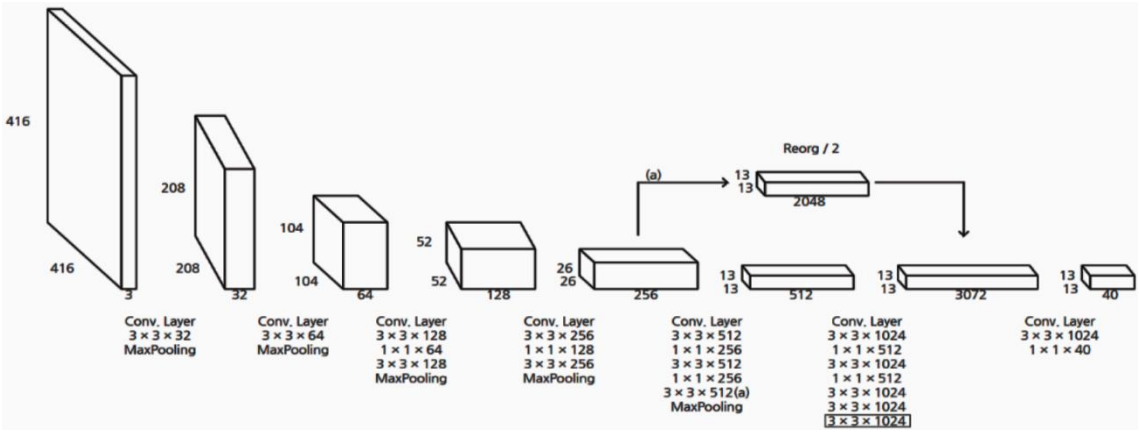
(örneğin, 3x3) ve daha küçük adım uzunluklarına sahip bir dizi evrişim katmanlarıdır. Bu katmanlar, görüntüdeki daha ince detayları çıkarır ve özellik haritalarını derinleştirir. Son katmanlarda ise YOLO mimarisinde, genellikle daha fazla filtre ve daha fazla derinlik içeren evrişim katmanları bulunur. Bu katmanlar, özelliklerin daha yüksek seviyelerde birleşmesini sağlar. Maksimum havuzlama katmanları, örneğin 2x2 maksimum havuzlama katmanı, 2x2 penceredeki maksimum değeri seçerek özellik haritasının boyutunu yarıya indirerek, evrişim katmanları tarafından oluşturulan özellik haritalarını aşağı örneklemek için kullanılır. Bu katmanlar, özellik haritalarının boyutunu azaltır ve modelin hesaplama karmaşıklığını düşürür. YOLO'nun son kısmında, tam bağlantılı katmanlar bulunur. Bu katmanlar, evrişim ve havuzlama katmanları tarafından çıkarılan özellikleri düzleştirir ve sınıflandırma ve sınır kutuları tahmini yapmak için kullanır. YOLO, her ızgara hücresi için sınıf olasılıkları ve sınır kutusu tahminleri üretir. Bu tahminler, her bir hücre için potansiyel nesnelerin yerini ve sınıfını belirtir. YOLO'nun çıkış katmanı, tüm tahminleri bir araya getirir ve son nesne tespiti sonuçlarını oluşturur. Çıkış katmanı, her ızgara hücresi için sınıf olasılıkları, sınır kutuları ve bu kutuların güven skorlarını içerir.

3.2.2. YOLOv2

YOLOv1'in ardından, YOLOv2 [59] 2017 yılında duyurulmuştur . YOLOv2 hem hız hem

de doğruluk açısından iyileştirmeler sunar. YOLOv2, daha derin ve daha güçlü bir evrişimsel sinir ağı olan Darknet-19'u kullanır. Darknet-19, Şekil 3.7'de de görüldüğü gibi 19 evrişim katmanı ve 5 maksimum havuzlama katmanından oluşur . Bu yapı, daha iyi özellik çıkarımı sağlar ve modelin doğruluğunu artırır. YOLOv2, her evrişim katmanından sonra batch normalization (toplu normalleştirme) uygular. Bu, eğitim sürecinde özelliklerin normalleştirilmesini sağlar, overfitting (aşırı uyum)'i azaltır ve modelin daha hızlı ve daha doğru bir şekilde eğitilmesini sağlar. YOLOv2, daha yüksek çözünürlüklü giriş görüntüleri (örneğin, 448x448 yerine 608x608) kullanır. Bu, küçük nesnelerin daha iyi tespit edilmesini sağlar ve genel doğruluğu artırır. YOLOv2, Faster R-CNN'den ilham alarak anchor kutuları kullanır. Anchor kutuları, çeşitli boyut ve oranlarda önceden tanımlanmış sınır kutularıdır. Bu yaklaşım, nesne tespitini daha esnek ve doğru hale getirir. YOLOv2, K-means kümeleme algoritmasını kullanarak eğitim veri setindeki nesnelerin boyutlarına göre optimal anchor kutuları belirler. YOLOv1'de olduğu gibi, YOLOv2'de doğrudan sınır kutusu merkez noktalarını ve boyutlarını tahmin eder. Ancak, YOLOv2'de bu tahminler, anchor kutularının ölçeklendirilmiş ve ofsetlenmiş versiyonları olarak yapılır. Bu, daha doğru sınır kutusu tahminleri sağlar. YOLOv2, daha düşük çözünürlüklü özellik haritalarını kullanarak yüksek çözünürlüklü tahminler yapar. Bu, daha küçük nesnelerin tespit edilmesine yardımcı olur ve genel doğruluğu artırır. Bu işlem, evrişim katmanlarının özelliklerini birleştirerek gerçekleştirilir. YOLOv2, farklı ölçeklerde eğitim yaparak modelin çeşitli giriş boyutlarında performans göstermesini sağlar. Bu, modelin daha esnek ve daha doğru olmasına katkıda bulunur. YOLOv2'nin YOLOv1'den farklılıkları şu şekilde sıralanabilir. Ağ mimarisi olarak YOLOv1, 24 evrişim katmanı ve 2 tam bağlantılı katman içeren bir ağ yapısı kullanırken, YOLOv2 daha derin ve daha verimli olan Darknet-19'u kullanır. YOLOv2, her evrişim katmanından sonra batch normalization uygular. Bu, eğitim sürecinde özelliklerin normalleştirilmesini sağlar, overfittingi azaltır ve modelin daha hızlı ve daha doğru bir şekilde eğitilmesini sağlar. YOLOv1 doğrudan sınır kutusu koordinatlarını tahmin ederken, YOLOv2 anchor kutuları kullanarak daha esnek ve doğru sınır kutusu tahminleri yapar. YOLOv2, daha yüksek çözünürlüklü giriş görüntüleri kullanarak küçük nesnelerin tespitini iyileştirir. YOLOv2, farklı ölçeklerde eğitim yaparak modelin çeşitli giriş boyutlarında performans göstermesini sağlar. YOLOv2, YOLOv1'e kıyasla bir dizi yapısal iyileştirme ve yenilik sunarak hem hız hem de doğruluk açısından önemli gelişmeler sağlar. Daha derin bir ağ mimarisi, batch normalization, anchor kutuları, yüksek çözünürlüklü girişler, fine-grained özellikler ve multi-scale eğitim gibi yenilikler,

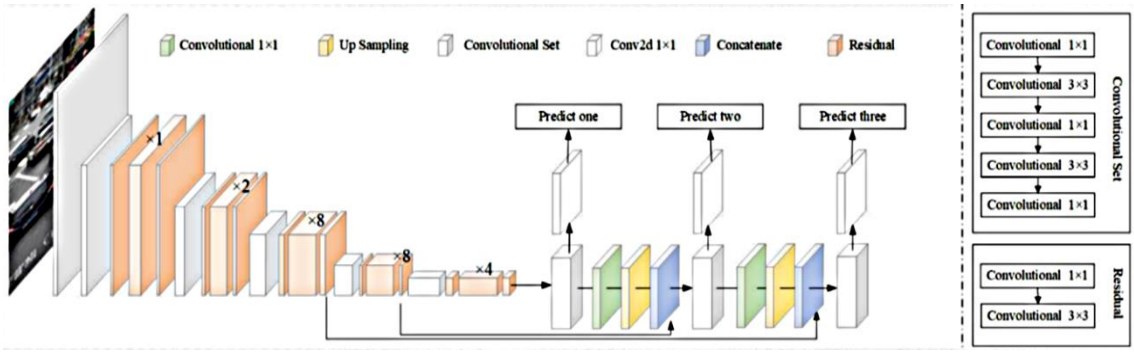
YOLOv2'nin nesne tespiti performansını artıran başlıca faktörlerdir. Bu iyileştirmeler, YOLOv2'yi bilgisayarla görme ve nesne tespiti alanında daha güçlü ve etkili bir araç haline getirmektedir.



Şekil 3.7. YOLOv2 mimarisi [60].

3.2.3. YOLOv3

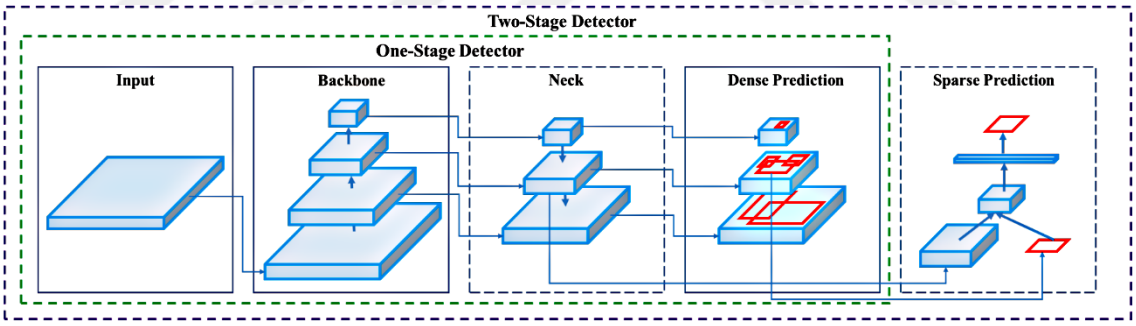
YOLOv1 ve YOLOv2'den sonra 2018 yılında aynı yazarlar son kez bir nesne algılama modeli olan YOLOv3 [61]'ü çıkarttılar. Şekil 3.8'de YOLOv3 yapısı detaylı bir şekilde gösterilmektedir. YOLOv2 de Darknet-19 mimarisini kullanırlarken YOLOv3'te yine kendi geliştirdikleri Darknet-53 isimli 53 evrişim katmanından oluşan yeni bir mimari kullanılmaktadır. Bu mimari 53 evrişim katmanı içerdiği için nesne algılama görevlerinde daha yüksek doğruluk sağlamaktadır. Darknet-53, ResNet mimarisinden ilham alınarak yığınlanmış kısımları (residual blocks) kullanır. Bu yapı derin ağlarda oluşabilecek öğrenme zorluklarını azaltmaktadır. YOLOv3 bounding boxlar için lojistik regresyon kullanarak daha hassas konum tahmini sağlar. Bu özellikle çakışan nesnelerin tespiti sırasında önemli bir avantaj sağlar. Dengesiz veri setlerin model eğitimi iyileştirme için kullanılan Focal Loss ile birlikte anchor kutuları kullanılarak özellikle nesne tespitinde önemli avantaj sağlar.



Şekil 3.8. YOLOv3'ün detaylı yapısı [62].

3.2.4. YOLOv4

YOLOv4 [63] farklı yazarlar tarafından geliştirilmiştir. Temel yapısal değişiklikler söz konusudur. Temel yapısal değişiklikler arasında, Backbone (omurga) olarak CSPDarknet53 kullanımı öne çıkar. Bu yapı, özellik çıkarımında daha verimli ve güçlü bir performans sergiler. YOLOv4, BoF (Bag of Freebies) ve BoS (Bag of Specials) adı verilen iki kategori altında çeşitli teknikleri bir araya getirir. BoF, modelin eğitim sürecinde doğruluğu artıran ve maliyeti olmayan teknikleri içerirken (örneğin, veri artırma, label smoothing, dropout), BoS ise modelin çıkarım aşamasında doğruluğu artıran ve az maliyetli teknikleri içerir (örneğin, Mish aktivasyon fonksiyonu, SPP blokları, PANet yol ağı). YOLOv4, ayrıca, daha iyi optimizasyon ve genelleme yeteneği sağlayan CıoU (Complete Intersection Over Union) kaybı, Mosaic veri artırma, DropBlock düzenlileştirme gibi yenilikçi teknikleri kullanır. Bu yenilikler, modelin hem eğitim sürecinde daha verimli öğrenmesini sağlar hem de gerçek zamanlı uygulamalarda daha yüksek performans sergilemesine yardımcı olur. YOLOv4, hız ve doğruluk arasındaki dengeyi optimize ederek, nesne tespiti görevlerinde hem akademik hem de endüstriyel uygulamalar için güçlü bir araç olarak öne çıkmaktadır.



Şekil 3.9. YOLOv4 modeli [63].

Şekil 3.9'da YOLOv4 modelinin mimarisi gösterilmektedir. YOLOv4, sabit boyutta giriş görüntüsü alır (genellikle 416x416 veya 608x608 piksel). Backbone, görüntüden özellik çıkarımında kullanılan ana ağ yapısıdır. YOLOv4'te Backbone olarak CSPDarknet53 kullanılır. CSPDarknet53, Darknet-53'ün bir versiyonudur ve CSPNet (Cross-Stage Partial Networks) yapısını içerir. CSPNet, bilgi akışını bölerek ve yeniden birleştirerek özellik çıkarımında daha etkili ve verimli bir performans sağlar. Bu yapı, daha derin ağlarda bilgi kaybını önler ve daha iyi özellik haritaları oluşturur. Neck, Backbone'dan çıkarılan özellikleri işleyerek farklı ölçeklerde daha fazla bilgi elde eden bir yapıdır. YOLOv4'te Neck kısmında SPP ve PANet (Path Aggregation Network) kullanılır. SPP,

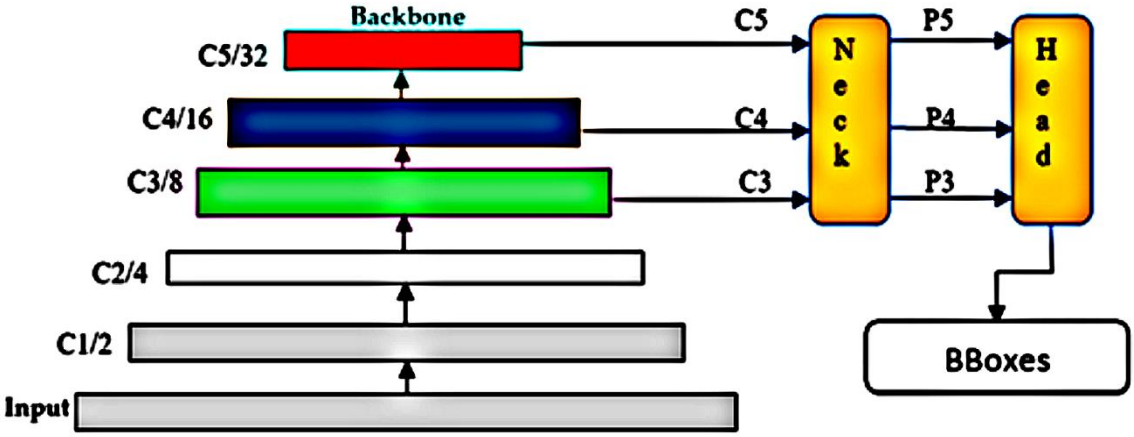
farklı ölçeklerdeki özellik haritalarını birleştirerek modelin nesne tespit yeteneğini artırır. PANet, farklı katmanlardan gelen bilgileri birleştirerek, küçük nesnelerin daha iyi tespit edilmesini sağlar ve ağın genel performansını iyileştirir. Dense Prediction, her hücre için belirli bir sayıda bounding box (sınır kutusu) koordinatları, obje skorları ve sınıf olasılıkları üretir. YOLOv4'te bu tahminler, Backbone ve Neck'ten gelen özellik haritaları kullanılarak yapılır. Dense Prediction, modelin tespit ettiği her nesne için yoğun bir tahmin seti üretir. Bu tahminler, genellikle üç farklı ölçekte gerçekleştirilir (küçük, orta ve büyük nesneler için). Sparse Prediction, NMS (Non-Maximum Suppression) gibi teknikler kullanarak üst üste binen ve düşük olasılığa sahip tahminleri filtreler. Bu aşamada, yalnızca yüksek olasılığa sahip ve güvenilir bounding boxlar seçilir. Bu yöntem, gereksiz ve yanlış tahminlerin önüne geçer, böylece daha kesin ve güvenilir nesne tespiti sağlar.

3.2.5. YOLOv5

YOLOv5, YOLOv4 temel alınarak geliştirildi ve çalışma hızı büyük ölçüde iyileştirildi, en yüksek hız saniyede 140 kareye ulaştı. Bu arada, YOLOv5'in boyutu küçüktür ve ağırlık dosyası YOLOv4 modelinden neredeyse %90 daha küçüktür; bu da YOLOv5'in gömülü cihazlara dağıtılmasına olanak tanır. YOLOv4 ile karşılaştırıldığında YOLOv5 daha yüksek doğruluk oranına ve küçük nesneleri daha iyi tanıma yeteneğine sahiptir [64]. İlk olarak Yolov5, CSPNet [65] Darknet'e dahil ederek CSPDarknet'i omurgası olarak yarattı. Eğitim modelinde mevcut olan parametreleri azaltabildiği için CSPNet ile çıkarım sürecinde daha yüksek hız ve doğruluk elde edilebilir. İkinci olarak Yolov5, bilgi akışını artırmak için PANet [66] uygulamıştır. PANet, düşük seviyeli özelliklerin yayılmasını iyileştiren, gelişmiş aşağıdan yukarıya yola sahip yeni bir FPN (Feature Pyramid Network) yapısını benimser. Aynı zamanda, özellik ızgarasını ve tüm özellik seviyelerini birbirine bağlayan uyarlanabilir özellik havuzu, her özellik seviyesindeki yararlı bilgilerin doğrudan aşağıdaki alt ağa yayılmasını sağlamak için kullanılır. PANet, alt katmanlarda doğru lokalizasyon sinyallerinin kullanımını geliştirir ve bu da açıkça nesnenin konum doğruluğunu artırabilir. Üçüncüsü, Yolov5'in başı, yani YOLO katmanı, çok ölçekli [61] tahmin elde etmek için 3 farklı boyutta (18 × 18, 36 × 36, 72 × 72) özellik haritası üretir ve modelin küçük, orta ölçekli işleri ele almasını sağlar [67]. YOLOv5'te tespit edilecek görüntü ile ilgili adımlar Şekil 3.10'da gösterilmektedir. Giriş katmanına gelen görüntü, işlenmek ve özellik çıkarımı için omurgaya gönderilir. Omurga farklı boyutlarda özellik haritaları elde eder ve bunları boyun ile birleştirerek P3, P4 ve P5

özellik haritalarını üretir. Son olarak bu haritalar, görüntüyü dikdörtgen bir çerçeve içine alan Sınırlayıcı Kutular (BBoxes) üretmek için Head ile birleştirilir [68].

YOLOv5 modelin genelleme yeteneğini geliştirmek ve aşırı uyumu azaltmak için çeşitli veri artırma teknikleri kullanır. Mosaic ve MixUp gibi ileri düzey veri artırma teknikleri kullanarak eğitim verisini çeşitlendirir ve modelin genelleme yeteneğini artırır. PyTorch üzerinde geliştirilmiş olan YOLOv5, GPU ve TPU'larda hızlı ve verimli çalışacak şekilde optimize edilmiştir, bu da gerçek zamanlı uygulamalarda kullanımını kolaylaştırır. Modelin kullanım kolaylığı, kullanıcı dostu arayüzü ve kapsamlı dokümantasyonu, araştırmacıların ve geliştiricilerin modeli hızlı bir şekilde kullanmaya başlamasını sağlar. YOLOv5, hız ve doğruluk arasında mükemmel bir denge kurarak hem düşük gecikmeli uygulamalara uygun hem de yüksek doğruluk sağlamaktadır. Bu yenilikler ve yapısal değişiklikler, YOLOv5'in hem akademik araştırmalar hem de endüstriyel uygulamalarda güçlü bir araç olarak kullanılmasını mümkün kılmaktadır.

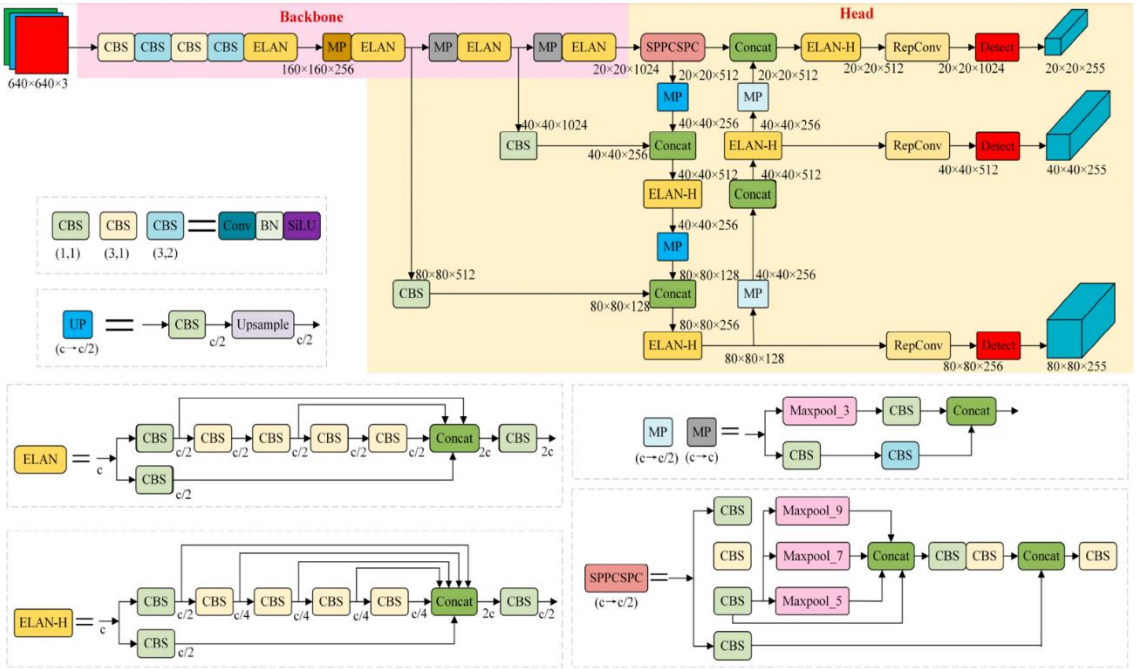


Şekil 3.10. YOLOv5'in varsayılan çıkarım akış şeması [68].

3.2.6. YOLOv7

YOLOv6 [69] ile YOLOv7 [2] farklı yazarlar tarafından 2022 yılının ortasında çıkmıştır. YOLOv7 modeli YOLOv6'ya göre karmaşık nesne algılama görevlerinde daha üstün performans sergilediği ve farklı veri modellerine göre hiperparametrelerinin optimize edilebilmesi bakımından YOLOv7 modeli bu tez çalışmasında tercih edilmektedir. YOLOv7 algoritması, 5FPS ila 160FPS aralığında bilinen diğer nesne dedektörlerinden daha iyi performans göstererek, nesne algılamadaki üstün hızı ve doğruluğu nedeniyle geniş çapta tanınmaktadır [3]. Gerekli donanım diğer sinir ağlarından birkaç kat daha ucuzdur ve önceden eğitilmiş ağırlıklar olmadan daha küçük veri kümeleri üzerinde çok daha hızlı eğitilebilir. Ayrıca YOLOv7, alternatif nesne algılama teknikleriyle

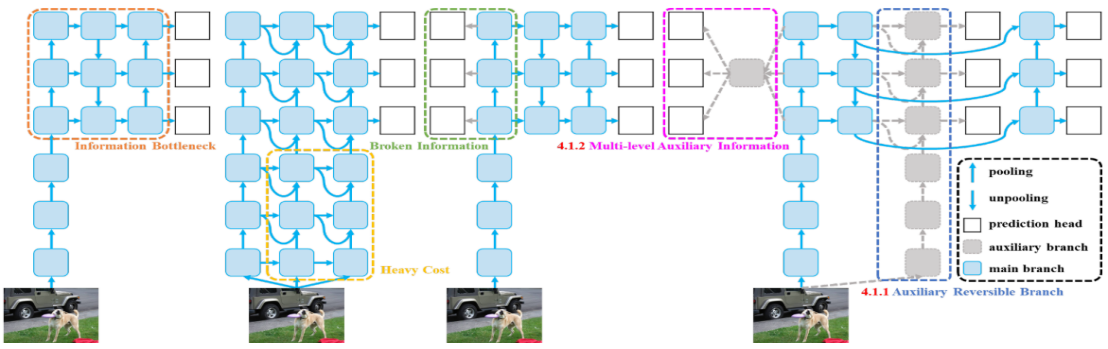
karşılaştırıldığında üstün hassasiyet, geri çağırma ve ortalama hassasiyet göstermiş, ilgilenilen nesnelerin tam olarak tanımlanması ve yerinin belirlenmesindeki etkinliğini vurgulamıştır [70]. Kentsel altyapı ve ulaşım alanında, YOLOv7 yol özelliği tespiti, çukur tespiti ve yaya-araç tespiti için kullanılmış olup, yol güvenliği ve altyapı bakımının arttırılmasındaki önemini vurgulamaktadır [4], [5]. Şekil 3.11'de görüldüğü gibi temelde YOLOv7, girdi (input), omurga (backbone), boyun (neck) ve kafa (head) bileşenlerinden oluşan ve bu bileşenleri kullanarak nesne tespiti gerçekleştiren bir YOLO modelidir. Girdi bileşeni, modele aktarılan görüntü veya videolardan oluşur. Omurga bileşeni, girdi görüntüsünden özellik çıkarımı yapan önceden eğitilmiş bir ağdır. Omurga katmanı, birkaç evrişimsel katmandan, E-ELAN (Extended-Efficient Layer Aggregation Network) katmanından ve dönüşümlü olarak boyutu yarıya indiren, kanalları ikiye katlayan ve özellik çıkarımı yapan MP (Maximum Pooling) katmanlarından oluşur. Boyun, omurgadan gelen özellikleri birleştirip zenginleştiren kısımdır. YOLOv7'de, Rep-PAN gibi bir özellik birleştirme mekanizması kullanılır. Bu sayede model, farklı ölçeklerdeki nesneleri daha iyi algılayabilir. Kafa bileşeni, son aşamada işlemleri gerçekleştirir. Bu katman, özellik haritalarına sınırlayıcı kutular uygular ve nihai çıktıyı oluşturur. Nihai çıktılar; nesne etiketleri, sınırlayıcı kutu çizimleri ve nesne tespit skorları olabilir.



Şekil 3.11. YOLOv7 model mimarisi [71].

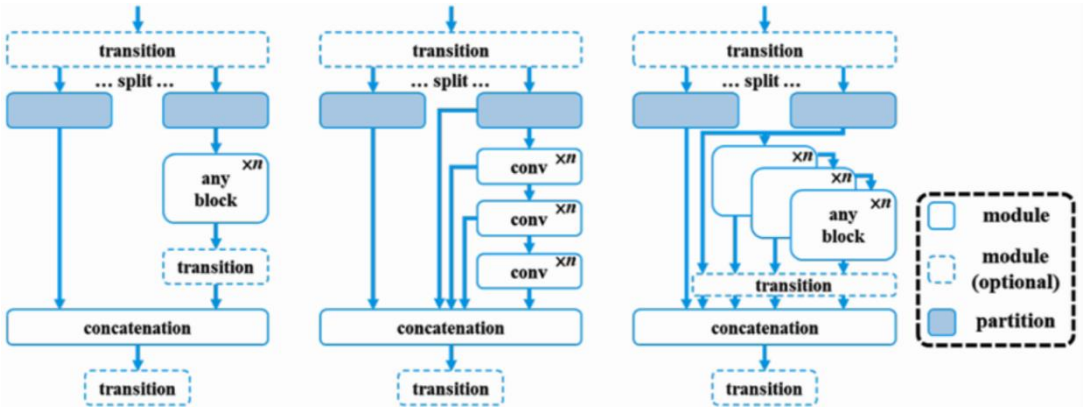
3.2.7. YOLOv9

YOLOv7 yazarları tarafından geliştirilen YOLOv9 [6], Şubat 2024'te piyasaya sürülen en son YOLO algoritmasıdır. YOLOv9 oluşturulurken önceki modellere kıyasla performansını artırmak için iki mimari birleştirilir. Şekil 3.12 ve Şekil 3.13'te YOLOv9'u oluşturan mimariler gösterilmektedir. PGI ve GELAN, YOLOv9'u oluşturmak için birleştirilir. Eğitilen verideki kritik bilgilerin birden fazla katmandan geçerken kaybolmasını önleyen PGI, öğrenmeyi daha verimli hale getiriyor. GELAN, katmanlardaki bilgilerin toplanması ve işlenmesi sırasında optimizasyon işlemlerini gerçekleştirerek modelin performansını artırır. Bu iki mimariyi birleştiren YOLOv9, nesne algılama uygulamaları için sağlam bir seçim haline gelir. PGI ve ilgili ağ mimarisi Şekil 3.12'de verilmiştir. PGI üç ana bileşenden oluşur: (1) çıkarımlar yapan bir ana dal, (2) çıkarım için güvenilir gradyanlar sağlayan bir yardımcı tersinir dal ve (3) çok düzeyli Yardımcı bilgi: Ana dal öğrenimini kontrol etmek için çok düzeyli anlamsal bilgi öğrenilebilir. Şekil 3.12 d)'de görüldüğü gibi PGI, çıkarım sürecinde yalnızca ana dalı kullanır, dolayısıyla hiçbir ek maliyete yol açmaz. DÖ yöntemlerinde diğer iki bileşen, birçok önemli sorunun çözülmesine veya yavaşlatılmasına yardımcı olur. Sinir ağları derinleştikçe yardımcı tersinir dalın geliştirilmesini gerektiren sorunlarla karşılaşırlar. Bir bilgi darboğazı ağın derinleşmesine neden olur ve bu da kayıp fonksiyonu tarafından üretilen güvenilmez gradyanlara neden olur. Çoklu tahmin dallarının neden olduğu hata birikimi sorununu ve derin denetimin neden olduğu hafif mimariyi ele alacak şekilde tasarlanmıştır. GELAN'ın mimarisi Şekil 3.13'de verilmiştir. Videolardaki eylemlerin doğru ve verimli tespitini sağlayan ELAN ve evrişimli sinir ağlarının verimliliğini ve doğruluğunu artıran CSPNet ile GELAN'ı oluşturmak üzere birleştirilir. GELAN bu iki mimarinin güçlü yönlerini birleştiriyor.



a) b) c) d)

Şekil 3.12. PGI ağları için yöntemler ve mimariler a) PAN (Path Aggregation Network) , b) RevCol (Reversible Column Network) , c) geleneksel derin denetim ve d) PGI.



(a) (b) (c)

Şekil 3.13. GELAN mimarisi: (a) CSPNet [65], (b) ELAN [72] ve (c) GELAN [6].

Çizelge 3.1'de YOLO modellerinin kullandığı framework, backbone (omurga), Eğitilen Veri Kümesi ve AP (Ortalama Hassasiyet) karşılaştırılması yer almaktadır. Tabloda yer alan AP değerleri VOC-2007'yi temel alan YOLOv1 ve YOLOv2 hariç COCO veri setini temel almaktadır [73].

Çizelge 3.1. YOLO modellerinin karşılaştırılması [73].

Model	Framework	BackBone (Omurga)	Eğitilen Veri Kümesi	AP(%)
YOLOv1	Darknet	Darknet-24	VOC-2007	63,4
YOLOv2	Darknet	Darknet-24	VOC-2007	63,4
YOLOv3	Darknet	Darknet-53	COCO	36,2
YOLOv4	Darknet	CSPDarknet-53	COCO	43,5
YOLOv5	Pytorch	Modified CSPv7	COCO	55,8
YOLOv6	Pytorch	EfficientRep	COCO	52,5
YOLOv7	Pytorch	RepConvN	COCO	56,8
YOLOv8	Pytorch	YOLOv8	COCO	53,9
YOLOv9	Pytorch	GELSAN	COCO	55,6

3.2.8. Gri Kurt Optimizasyon

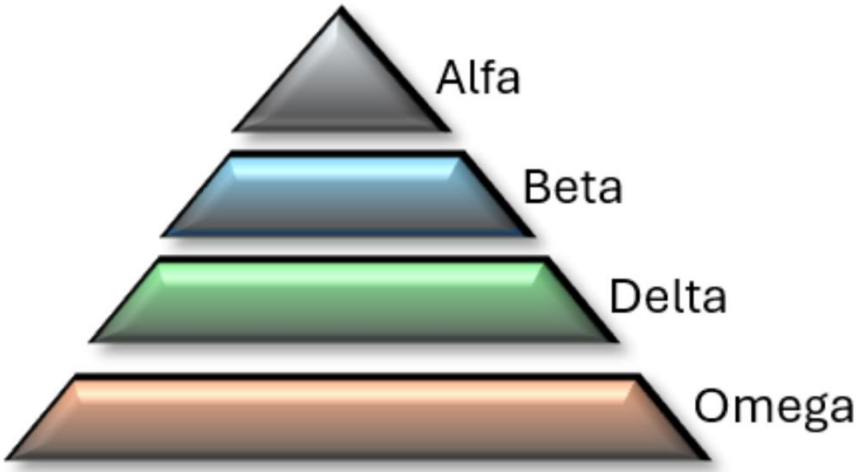
Gri kurt taktiklerini ve sosyal davranışları temel alan GKO, meta-sezgisel bir optimizasyon yöntemidir. GKO, bir gri kurt sürüsü tarafından sergilenen sosyal hiyerarşiyi ve avlanmayı taklit eder. Gri kurtla bir grup halinde yaşarlar. Bu grupta katı bir sosyal hiyerarşi vardır. Şekil 3.14'te gösterildiği gibi bir grup gri kurdun "Alfa" olarak bilinen liderleri, gruptaki diğer kurtlar adına karar vermekten sorumlu olan dişi ve erkek

kurtlardır. Genel olarak piramidin altındaki diğer kurtla Alfa kurtların verdiği kararlara uymak zorundalardır. Gri kurt sürülerinin en önemli özelliklerinden biri düzenli ve kendilerine hâkim olmalarıdır. Piramidin ikinci kademesindeki "Beta" kurtlarının görevi Alfaya yardımcı olmaktır. Beta, herhangi bir Alfa kurdu yaşlandığında ya da öldüğünde Alfanın yerine geçebilir. Beta, Alfa'nın emirlerini uygular, gruptaki herhangi bir hatalı bireyi cezalandırmaktan sorumludur. Ayrıca grup üyelerinin Alfaya yanıt vermesini sağlar. Piramidin dördüncü kısmında Omega kurtları yer alır. Bu seviyedeki kurtlar Alfa ve Beta kurtlarının emirlerine uymak zorundadır. Omega grupta en önemsiz kurtlar gibi görünse de Omega olmadan iç çatışma ve diğer sorunların varlığını tespit etmek zor olmaktadır. Bunun nedeni Omega'nın gruptaki diğer kurtların hoşnutsuzluğunu ve zulmü ortaya çıkarma sorumluluğu vardır. Alfa, Beta ve Omega dışında piramidin üçüncü kısmında yer alan diğer kurtlar ikincil (Delta) olarak adlandırılmaktadır. Delta kurtları Alfa ve Beta kurtlarına itaat eder ve Omega kurtlarına hükmeder. Grupta casus, bekçi, ihtiyar, avcı ve muhafız olarak görev yaparlar [74].

GKO algoritmasındaki gri kurtların sosyal hiyerarşisinde, en iyi çözüm Alfa α ile temsil edilir. İkinci en iyi çözüm Beta β, üçüncü en iyi çözüm Delta δ ile temsil edilir [74]. GKO algoritmasında, avlanma yani optimizasyon α, β ve δ kurtları tarafından yönlendirilirken, ω onların arkasından gider. Arama uzayındaki potansiyel çözümleri temsil etmek için kullanılan bu Alfa, Beta, Delta ve Omega kurtları ile tüm çözüm uzayını keşfetme ve yerel bölgelere odaklanma arasında bir denge elde edilir [7]. Kurt pozisyonları, bu süreçte, avını takip eden yırtıcı hayvanların hareketini takip ederek potansiyel çözümlere doğru dinamik olarak uyarlanır. Kurt pozisyonları algoritma sırasında yinelemeli olarak iyileştirilir ve her yinelemede optimal çözüme ulaşılmasına yol açar. Bu teknikleri kullanarak GKO algoritması, en uygun çözümü bulmak için arama alanını kademeli olarak daraltır. Sonuç olarak, çok çeşitli görevleri optimize etmek için kullanışlıdır. Kurtların avını çevreleme ve avlama adımları aşağıdaki denklemlerle matematiksel olarak ifade edilmiştir. Burada t mevcut yinelemeyi, $\vec{A}$ ve $\vec{C}$ katsayı vektörlerini, $\vec{X}_p$ av konumunu ve $\vec{X}$ ise gri kurt konumunu belirtmektedir [75]. $\vec{A}$ ve $\vec{C}$ vektörlerinin hesaplanması aşağıdaki gibidir:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \tag{3.1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{3.2}$$



Şekil 3.14. Gri kurt hiyerarşisi.

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{3.3}$$

$$\vec{C} = 2 \,.\vec{r}_2 \tag{3.4}$$

$\vec{A}$ bileşenleri yinelemeler boyunca doğrusal olarak 2'den 0'a azaltılırken, $\vec{r}_1$ ve $\vec{r}_2$ rastgele vektörlerdir [18]. Gri kurt, Denklem 3.1 ve Denklem 3.2 denklemlerini kullanarak avın etrafındaki herhangi bir rastgele konumun yakınındaki konumunu güncelleyebilir. Kurtlar avlarını tanıyabilir ve etrafını sarabilir. Avlanma, Alfa kurt tarafından yönetilen, Beta ve Deltanın da ara sıra katılımıyla gerçekleşen bir süreçtir. Ancak soyut bir arama uzayında avın nerede bulunacağını belirlememektedir. Gri kurt avlanma davranışını matematiksel olarak simüle etmek için Alfa (mümkün olan en iyi çözüm), Beta ve Deltanın avın nerede bulunabileceği hakkında daha iyi bilgiye sahip olduğu varsayılır.

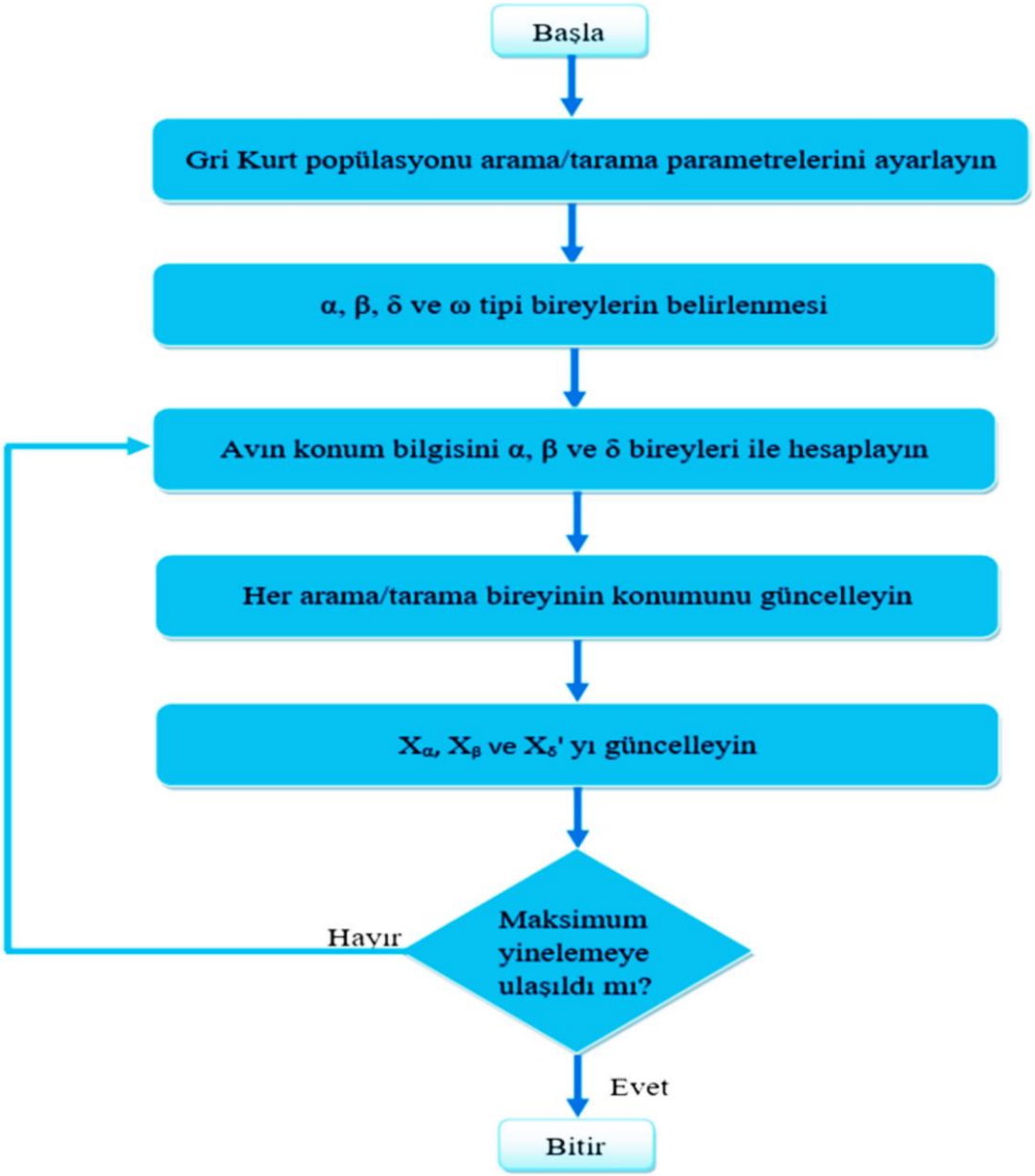
$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \cdot \vec{X}_\alpha - \vec{X}\right|, \quad \overrightarrow{D_\beta} = \left|\overrightarrow{C_2} \cdot \vec{X}_\beta - \vec{X}\right|, \quad \overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \cdot \vec{X}_\delta - \vec{X}\right| \tag{3.5}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \left(\overrightarrow{D_\alpha}\right), \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \left(\overrightarrow{D_\beta}\right), \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \left(\overrightarrow{D_\delta}\right) \tag{3.6}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{3.7}$$

Sorunu çözmek için, ilk üç en iyi çözüm kaydedilir ve Omegalar da dahil olmak üzere mevcut tüm arama aracıları, konumlarını en iyi çözüme göre güncellemeye zorlanır . Kurt α'nın liderliğindeki kurtlar β ve δ yavaş yavaş avlarına gizlice girerler. Denklem 3.5 ve Denklem 3.6 kullanarak α, β ve δ arasındaki mesafe hesaplanmakta ve ardından Denklem 3.7 ile gri kurdun bireysel olarak ava doğru nasıl hareket edebileceği belirlenmektedir [7]. Formülde, $\vec{X}_\alpha$ α kurdun konumudur, $\vec{X}_\beta$ β kurdun konumunu belirtir, $\vec{X}_\delta$ δ kurdun

konumudur ve C_1, C_2 ve C_3 rastgele başlatılan vektörlerdir. Denklem 3.4, $Ci\ (i\ = 1, 2, 3)$ formülüdür, dolayısıyla $Ci\ (i\ = 1, 2, 3)$ [0, 2] aralığına sahiptir [7]. Optimizasyon sorunlarının üstesinden gelmek için GKO algoritması içinde gri kurtların doğal sosyal hiyerarşisini ve avlanma taktiklerini taklit etmeye yönelik yinelemeli süreç, Şekil 3.15 akış şemasında gösterilmektedir. Her yineleme sırasında, arama alanındaki kurt konumları olarak sembolize edilen aday çözümler, şu ana kadar tanımlanan en iyi, ikinci en iyi ve üçüncü en iyi çözümleri temsil eden üç ana rolden etkilenen güncellemelerden geçer: Alfa, Beta ve Delta kurtları. Çizelge 3.2'de GKO'da kullanılan parametreler, açıklamaları ve değerleri verilmektedir.



Şekil 3.15. GKO algoritmasının akış şeması.

Çizelge 3.2. GKO parametreleri ve çalışma şartları.

Parametre	**Açıklama**	**Değer**
Npop	Popülasyon boyutu	1000
Niter	İterasyon sayısı	100
alpha	Crossover oranı	0,5 – 1,0 arası rastgele bir değer
beta	Mutasyon oranı	0 – 1 arası rastgele bir değer
gamma	Seçim parametresi	1 – 3 arası rastgele bir değer
Veri seti	DAWN - RTTS	-
Değerlendirme ölçütü	mAP	-
Donanım	Google COLAB V100	-

3.2.9. Yapay Tavşan Optimizasyon

YTO, 2022 yılında yeni bir meta-sezgisel optimizasyon algoritması olarak piyasaya sürüldü [8]. En iyi çözümü bulmak için doğadan ilham alan çeşitli optimizasyon algoritmaları geliştirilmiştir. YTO'da tavşanların çevresini yiyecek arama ve keşfetme süreci bu şekilde modellenmiş ve simüle edilmiştir. Tavşanlar doğası gereği tehlikeli ortamlardan kaçınır ve aynı zamanda en iyi besin kaynağını bulmak için yeni ortamları keşfederler. Yuvanın yırtıcı hayvanlar tarafından bulunmasını önlemek için tavşanlar deliklerin yakınındaki otları yemezler, yiyeceklerini daima yuvalarının uzağında ararlar. Tavşanlar oldukça geniş bir görüş alanına sahiptir. Böylece geniş bir alanda kolayca yiyecek bulabilirler. Tavşanların bu hayatta kalma stratejisi, popüler bir Çin deyimi olarak geniş çapta yayılmıştır: "Tavşanlar kendi yuvalarının yakınındaki otları yemezler." Tavşanların bir diğer hayatta kalma stratejisi de rastgele saklanmaktır. Tavşanlar, yırtıcı hayvanlardan veya avcıların izinden kaçmak için yuva yapmakta iyidirler; bir tavşan, yuvasının etrafına birçok yuva kazar ve yırtıcı hayvanlardan korunmak için rastgele bir yuva seçer. Ayrıca düşmanlarından kurtulabilmek için fiziksel bir avantaja sahiptirler. Tavşanların ön ayaklarının kısa, arka ayakları uzun olması ve daha hızlı koşmalarını sağlayan güçlü kas ve tendonlara sahip olması sayesinde tavşanlar koşarken aniden durabilir, keskin bir şekilde dönebilir veya kovalamacadan kaçmak için zikzaklı bir hareketle geri koşabilirler. Bu hayatta kalma stratejisi, düşmanlarının kafasını kolayca karıştırabilir ve takipten kaçmalarına yardımcı olabilir, bu da tavşanların hayatta kalma olasılığını etkili bir şekilde artırabilir. Tavşanlar besin zincirinin en altında olduğundan ve çok fazla yırtıcı hayvana sahip olduğundan, tavşanların tehlikeden kaçmak için hızlı koşması gerekir, bu onların enerjisini azaltır, dolayısıyla tavşanların enerjiye göre dolambaçlı yiyecek arama ve rastgele saklanma arasında uyarlanabilir bir şekilde geçiş yapması gerekir. YTO, gerçek tavşanların yiyecek arama ve saklanma stratejilerinin yanı

sıra enerjilerinin azalmasını da kullanarak her iki strateji arasında geçiş sağlar.

Yiyecek arama ve rastgele saklanma için ayrı matematik modelleri uygulanmaktadır. Tavşanlar, yiyecek ararken yeterli yiyecek almak için bir yiyecek kaynağının etrafında tedirginlik gösterebilirler. Bu nedenle, YTO'nun dolambaçlı yiyecek arama davranışı, her arama bireyinin sürüde rastgele seçilen diğer arama bireyine göre konumunu güncelleme ve bir tedirginlik ekleme eğiliminde olduğunu gösterir. Tavşanların dolambaçlı yiyecek aramasının matematiksel modeli önerilmiştir:

$$\vec{v}_i(t+1) = \vec{x}_j(t) + R\,.\left(\vec{x}_i(t) - \overrightarrow{x_j}(t)\right) + round\left(0{,}5.\,(0{,}05 + r_1)\right).n_1,$$

$$i, j = 1, \dots, n \; ve \; j \neq i \tag{3.8}$$

$$R = L\,.c \tag{3.9}$$

$$L = (e - e^{\left(\frac{t-1}{T}\right)^2}).\,sin(2\pi r_2) \tag{3.10}$$

$$c(k) = \begin{cases} 1, & if\ k == g(l) \\ 0, & else \end{cases} \quad k = 1, \dots, d \; ve \; l = 1, \dots, \lceil r_3\,.\,d \rceil \tag{3.11}$$

$$g = randperm(d) \tag{3.12}$$

$$n1 \sim N(0,1) \tag{3.13}$$

burada $\vec{v}_i(t+1)$ i'inci tavşanın t+1 anındaki aday konumudur, $\vec{x}_i(t)$ i'inci tavşanın t zamanındaki konumudur, n tavşan popülasyonunun büyüklüğüdür, d ise problemin boyutu, T maksimum yineleme sayısıdır, $\lceil \cdot \rceil$ tavan fonksiyonudur, round en yakın tamsayıya yuvarlamayı belirtir, randperm 1'den d'ye kadar olan tamsayıların rastgele bir permütasyonunu döndürür, $r1$, $r2$ ve $r3$ üç rastgeledir (0,1)'deki sayılar, L, dolambaçlı yiyecek aramayı gerçekleştirirken hareket hızını temsil eden koşu uzunluğudur ve $n1$, standart normal dağılıma tabidir.

Denklem 3.8'de, tedirginlik YTO'nun yerel aşırılıklardan kaçınmasına ve küresel arama yapmasına yardımcı olabilir. Denklem 3.10'a göre, çalışma uzunluğu L, ilk yinelemeler sırasında daha uzun bir adım oluşturabilir. Bu uzunluk daha sonraki yinelemelerde daha kısa bir adım oluşturabilir. Denklem 3.9'da c, algoritmanın yiyecek arama davranışında mutasyona uğratılacak arama bireylerinin rastgele sayıdaki öğelerini rastgele seçmesine yardımcı olabilecek bir haritalama vektörüdür. R, tavşanların koşma karakteristiğini simüle etmek için kullanılan koşma operatörünü temsil eder.

Denklem 3.8'de, arama yapan bireylerin birbirlerinin konumuna göre rastgele bir yiyecek araması yaptığını gösterir. Bu davranış, bir tavşanın kendi bölgesinden diğer tavşanların bölgelerine doğru hareket etmesini sağlar. Kendi yuvaları yerine başkalarının yuvalarını

ziyaret eden tavşanların bu özel yiyecek arama davranışı, keşfe önemli ölçüde katkıda bulunur ve YTO algoritmasının küresel arama yeteneğini garanti eder.

Yırtıcı hayvanlardan kaçmak için, bir tavşan genellikle saklanmak üzere yuvasının etrafına bazı farklı yuvalar kazar. YTO'da, her yinelemede, bir tavşan her zaman arama uzayının her boyutu boyunca etrafında d yuvalar üretir ve avlanma olasılığını azaltmak için saklanmak üzere her zaman tüm yuvalardan birini rastgele seçer. Bu konuyla ilgili aşağıdaki denklem verilmiştir. *i*'inci tavşanın *j*'inci yuvası şu şekilde oluşturulur:

$$\vec{b}_{i,j}(t) = \vec{x}_i(t) + H \cdot g \cdot \vec{x}_i(t)\,, i\ =\ 1,\ldots,n\ ve\ j\ =\ 1,\ldots,d \tag{3.14}$$

$$H\ =\ \frac{T-t+1}{T}\ .r_4 \tag{3.15}$$

$$n_2 \sim N(0,1) \tag{3.16}$$

$$g(k) = \begin{cases} 1, & if\ k == j \\ 0, & else \end{cases}\ k\ =\ 1,\ldots,d \tag{3.17}$$

Denklem 3.14'de, d yuvaları her boyut boyunca bir tavşanın mahallesinde oluşturulur. H, yinelemeler boyunca rastgele bir pertürbasyonla doğrusal olarak 1'den 1/T'ye azaltılan gizleme parametresidir. Bu parametreye göre başlangıçta bu yuvalar tavşanın daha büyük bir mahallesinde oluşturulur. Tekrarlar arttıkça bu komşuluk da azalıyor.

Yukarıda belirtildiği gibi, tavşanlar sıklıkla yırtıcı hayvanların takibine ve saldırılarına maruz kalır. Hayatta kalmak için tavşanların saklanacak güvenli bir yer bulması gerekir. Bu nedenle, yakalanmamak için sığınmak üzere yuvalarından rastgele bir yuva seçmeleri reddedilir. Bu rastgele gizleme stratejisini matematiksel olarak modellemek için denklemler önerilmektedir:

$$\vec{v}_i(t+1) = \vec{x}_i(t) + R\ .\left(r_4\ .\ \vec{b}_{i,r}(t) -\ \vec{x}_i(t)\right), i\ =\ 1,\ldots,n \tag{3.18}$$

$$g_r(k) = \begin{cases} 1, & if\ k == \lceil r_5\ .d \rceil \\ 0, & else \end{cases}\quad k\ =\ 1,\ldots,d \tag{3.19}$$

$$\vec{b}_{i,r}(t) = \vec{x}_i(t) + H \cdot g_r \cdot \vec{x}_i(t) \tag{3.20}$$

burada $\vec{b}_{i,r}$ d yuvalarından saklanmak için rastgele seçilmiş bir yuvayı temsil eder ve r_4 ve r_5 (0,1)'deki iki rastgele sayıdır. Denklem 3.18'e dayanarak, i'inci arama bireyi d yuvalarından rastgele seçilen yuvaya doğru konumunu güncellemeye çalışacaktır. Hem dolambaçlı yiyecek arama hem de rastgele saklanma yöntemlerinden biri başarıldıktan sonra, i. tavşanın konum güncellemesi şu şekildedir:

$$\vec{x}_i(t+1) = \begin{cases} \vec{x}_i(t) & f(\vec{x}_i(t)) \leq f(\vec{v}_i(t+1)) \\ \vec{v}_i(t+1) & f(\vec{x}_i(t)) > f(\vec{v}_i(t+1)) \end{cases} \quad (3.21)$$

Bu denklem, eğer *i*'inci tavşanın aday konumunun uygunluğu mevcut olanınkinden daha iyiyse, tavşanın mevcut konumunu terk edeceğini ve Denklem 3.8 veya Denklem 3.18 tarafından oluşturulan aday konumda kalacağını gösterir.

YTO'da tavşanlar her zaman yinelemelerin başlangıç aşamasında sıklıkla dolambaçlı yiyecek arama yapma eğilimindeyken, yinelemelerin sonraki aşamalarında sıklıkla rastgele saklanma gerçekleştirirler. Bu arama mekanizması, tavşanın zaman geçtikçe azalacak olan enerjisinden kaynaklanmaktadır. Bu nedenle, aramadan işletmeye geçişi modellemek için bir enerji faktörü tasarlanmıştır. ARO'daki enerji faktörü şu şekilde tanımlanır:

$$A(t) = 4(1 - \frac{t}{T}) ln\frac{1}{r} \quad (3.22)$$

burada r, (0,1)'deki rastgele sayıdır. YTO'da, enerji faktörü $A(t) > 1$ olduğunda, bir tavşan, keşif aşamasında yiyecek aramak için farklı tavşanların bölgelerini rastgele keşfetmeye eğilimlidir, dolayısıyla dolambaçlı yiyecek arama gerçekleşir. Enerji faktörü A(t)≤1 olduğunda, bir tavşan kullanım aşamasında kendi yuvalarını rastgele kullanma eğiliminde olur, dolayısıyla rastgele saklanma meydana gelir. A enerji faktörünün değerine bağlı olarak YTO, dolambaçlı yiyecek arama veya rastgele saklanma arasında geçiş yapabilir. Yani $A(t) > 1$ olduğunda keşif, $A(t) \leq 1$ olduğunda ise kullanım gerçekleşir.

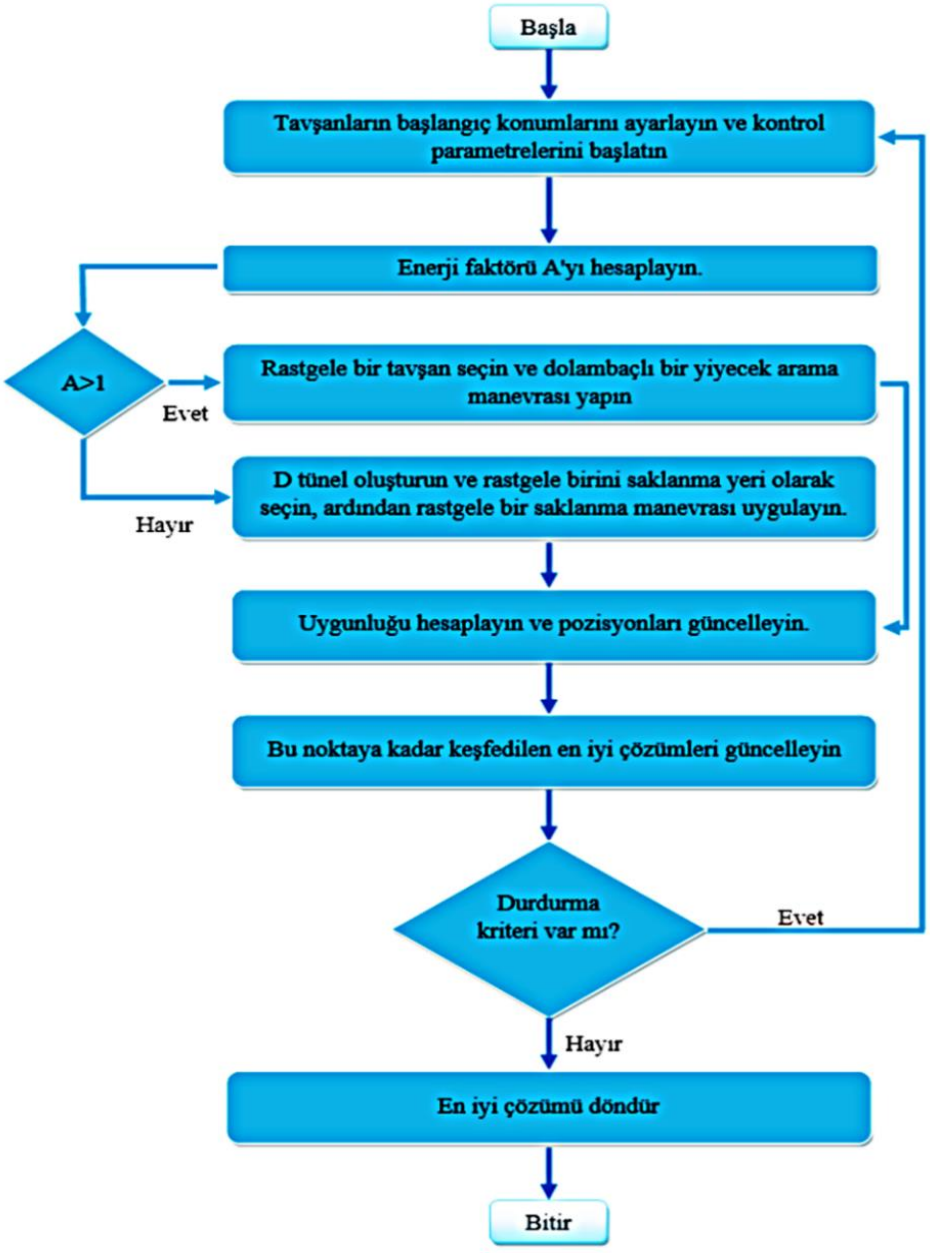
Enerji faktörünün algoritmanın arama davranışı üzerindeki etkisini araştırmak için $A > 1$ olasılığı hesaplanır. $\theta = 2 \cdot (1 - \frac{t}{T})$ olsun, o zaman $A(t) = 2 \cdot ln\frac{1}{r} \cdot \theta$ olsun, $A > 0$ olasılığı şu şekilde hesaplanır:

$$P\{A(t) > 1\} = \frac{\int_0^2 \int_0^{e^{-\frac{1}{2k}}} dr d\theta}{1.2} = \frac{1}{4} \cdot \int_{-\infty}^{-\frac{1}{4}} \frac{e^t}{t} dt + e^{-\frac{1}{4}} \approx 0{,}5177 \quad (3.23)$$

Dolayısıyla, yinelemeli süreçte dolambaçlı yiyecek arama olasılığı yaklaşık 0,5'tir. Başka bir deyişle, YTO algoritması, yinelemeli süreçte hem dolambaçlı yiyecek arama hem de rastgele gizleme işlemlerini hemen hemen aynı miktarda gerçekleştirir; bu da keşif ve kullanım arasında denge kurulmasına önemli ölçüde katkıda bulunur.

Şekil 3.16 YTO algoritmasının nasıl çalıştığını gösteren akış şemasıdır. Tavşanların pozisyon ve kontrol parametreleri yüklenir. Tavşanların enerji faktörü hesaplanır.

Rastgele bir osilatörün yineleme sayısı arttıkça, A enerji faktörünün boyutu arama davranışını belirler. Eğer A 1'den büyükse rastgele bir tavşan seçilir ve yiyecek arama manevrası gerçekleştirilir. A'nın 1'den küçük olması durumunda bir tünel oluşturulur, rastgele bir saklanma yeri seçilir ve rastgele bir saklanma manevrası gerçekleştirilir. İki durumdan biri gerçekleştikten sonra uygunluk hesaplanır ve konum güncellenir. Bu durum güncellenene kadar en iyi çözüm bulunmuştur. Bu adımlar durdurma kriteri sağlanana kadar tekrarlanır. Sonunda en iyi çözüm döndürülür. YTO, yinelemeli keşif ve kullanım süreci nedeniyle çeşitli uygulamalar için kullanılabilen bir optimizasyon yöntemidir. Çözüm uzayında etkili bir şekilde gezinilir ve optimum çözümlere ulaşılır.



Şekil 3.16. YTO algoritmasının akış şeması.

Çizelge 3.3'te YTO'da kullanılan parametreler, açıklamaları ve değerleri verilmektedir.

Çizelge 3.3. YTO parametleleri ve çalışma şartları.

Parametre	**Açıklama**	**Değer**
Popülasyon Boyutu	Algoritmanın çalıştırıldığı tavşan sayısı	50
Maksimum Iterasyon	Algoritmanın çalıştırılacağı maksimum döngü sayısı	1000
Öğrenme Oranı	Tavşanların öğrenme hızını belirleyen parametre	0.01-0.1
İnertsiya Ağırlığı	Tavşanların önceki hızlarını ne kadar koruyacakları	0.5-1.0
Başlangıç Koşulları	Tavşanların başlangıç konumları	Rastgele veya Heuristik
Veri seti	DAWN - RTTS	-
Değerlendirme ölçütü	mAP	-
Donanım	Google COLAB V100	-

3.2.10. Şempanze Lider Seçme Optimizasyon

ŞLSO, Kasım 2022'de piyasaya sürülen en son Meta-sezgisel optimizasyon algoritmalarından biridir. Şempanze ailesinin liderlerini nasıl seçtiğinden ilham almaktadır. Lider Alfa erkek olmalıdır. Ancak seçilen bu liderin diğer erkek ve dişilerle olan ilişkilerinin ŞLSO algoritmasında önemli rol oynadığı görülmektedir. Şempanzelerin ezberleme yeteneği, yaşla birlikte ters U şeklinde bir performans artışı göstermektedir. Denklem 3.24'te bu performans basit bir şekilde matematiksel olarak modellenebilir [9].

$$y = ax^2 + bx + c \tag{3.24}$$

y sembolü bağımlı değişkendir, *x* bağımsız değişkendir, a ve b katsayılardır ve c bir sabittir. Ayrıca, alanlar $x \in [0,1]$ ve $y \in [0,1]$ arasında ters bir U grafiği elde etmek için Denklem 3.24'ü kullanabilir, yani (0,0), (0.5,1) ve (1,0), böylece *a*, *b* ve *c* değerlerini Denklem 3.25 – 3.27'ye göre yazabilir. *c* değerini elde etmek için, nokta (0,0)'ı Denklem 3.25 olarak kullanabilir.

$$(0,0) \Rightarrow 0 = a(0)^2 + b(0) + c \;,\; . \; . \; c = 0 \tag{3.25}$$

Bu durumda, şempanzelerin yaş sınırı değeri ve bilişsel performansına yönelik bir yaklaşım, şempanzelerin yaşının 0 ile 1 arasında değiştiği yerde kullanılabilir. Bu nedenle, *a* ve *b* değerlerini elde etmek için, *c* değeri Denklem 3.26'da kullanır.

$$(1,0) \Rightarrow 0 = a(1)^2 + b(1) \therefore 0 \ ,. \ .a = -b \qquad (3.26)$$

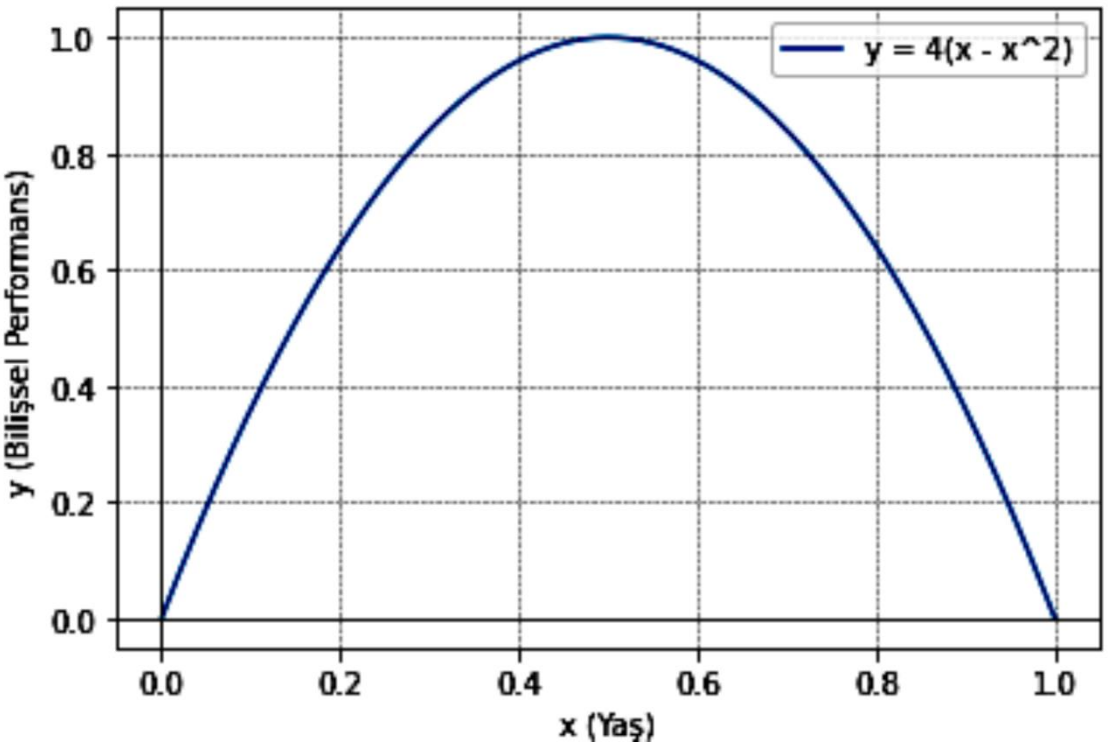
Denklem 3.25 ve 3.26'dan sırasıyla $c = 0$ ve $a = -b$ değerleri elde edilmiştir. Şempanzelerin optimum bilişsel performans modeli, $x = 0{,}5$'te $y = 1$'lik bir maksimum değere sahiptir. Daha sonra b değeri Denklem 3.27 olarak elde edilir.

$$(0{,}5, 1) \Rightarrow 1 = -b(0{,}5)^2 + b(0{,}5) \therefore 0 \ ,. \ . \ b = 4 \qquad (3.27)$$

Şempanzenin bilişsel durum modelinin denkleminin Denklem 3.28'deki gibi elde edilebilmesi için $a = -4$, $b = 4$ ve $c = 0$ değerlerinin elde edilebileceği sonucuna varılabilir [9].

$$y = 4(x - x^2) \qquad (3.28)$$

Denklem 3.28 $x \in [0,1]$ değeri için çizilirse Şekil 3.17'de gösterildiği gibi bir grafik elde edilir.



Şekil 3.17. Şempanze için x(yaş) ve y(bilişsel performans) arasındaki ilişkinin model grafiği.

Şempanzenin bilişsel durum yaklaşım modeli, Yaş ∈ [0,1] ve Bilişsel Performans ∈ [0,1] gereksinimlerini karşıladığı ve ters U şeklinde olduğu sürece diğer modelleri kullanabilir.

Şempanzelerin yetenek performans güncelleme denklemi Denklem (6) olarak modellenebilir.

$$X^{+} = r_{age}X + 4(r_{age} - r_{age}^{2})(X_{best} - X) \quad (3.29)$$

Sembol X^{+}, şempanzenin bireysel yetenek performansını ve X değerini sürekli olarak güncelleyecek yinelemeyi temsil eder. Değişken r_{age} rastgele bir değerdir, $r_{age} \in [0,1]$, X_{best} ise X için yerel en iyi yetenek performansıdır.

Tüm bireysel şempanzeler yeteneklerini güncelledikten sonraki adım, Alfa erkek şempanzeler olmak için yetenek performanslarını güncellemektir. Bu performans durumu, bir topluluk lideri olmak için hayati öneme sahiptir. Erkek şempanzelerin topluluktaki erkek ve dişi şempanzelerle ilişki kurma olasılığı daha yüksektir, bu nedenle durumlarını X^{*}güncellemek, erkek (X_M) ve dişi (X_F) eşlerle bir dereceye kadar ilişki gerektirir. Ayrıca, topluluktaki küresel en iyi şempanze (X_{Gbest}) ile ilişkileri güncellemek de gereklidir. Bu durumda denklem modeli, Denklem 3.30'a göre gösterilmiştir.

$$X^{*} = X + r_1(X_M - X) + r_2(X_F - X) + r_3(X_{Gbest} - X) \quad (3.30)$$

Erkek şempanzenin özelliği, doğduğu topluluğu asla terk etmemesidir, bu nedenle sınır koşullarının dışına düşen değerler tanımlanmalıdır. X değerinin üst sınır değerinden (X_{UB}) fazla olduğu varsayılsın. Bu durumda, (X_{UB}) değeri üst sınır değeriyle aynı olacaktır. (X_{UB}) değeri alt sınır (X_{LB}) değerinden küçükse, (X_{UB}) değeri alt sınır değeriyle aynı olacaktır. Bu sınır değer gereksinimi sırasıyla Denklem 3.31 ve Denklem 3.32 denklemleri olarak yazılabilir [9].

$$X > X_{UB} \Rightarrow X = X_{UB} \quad (3.31)$$

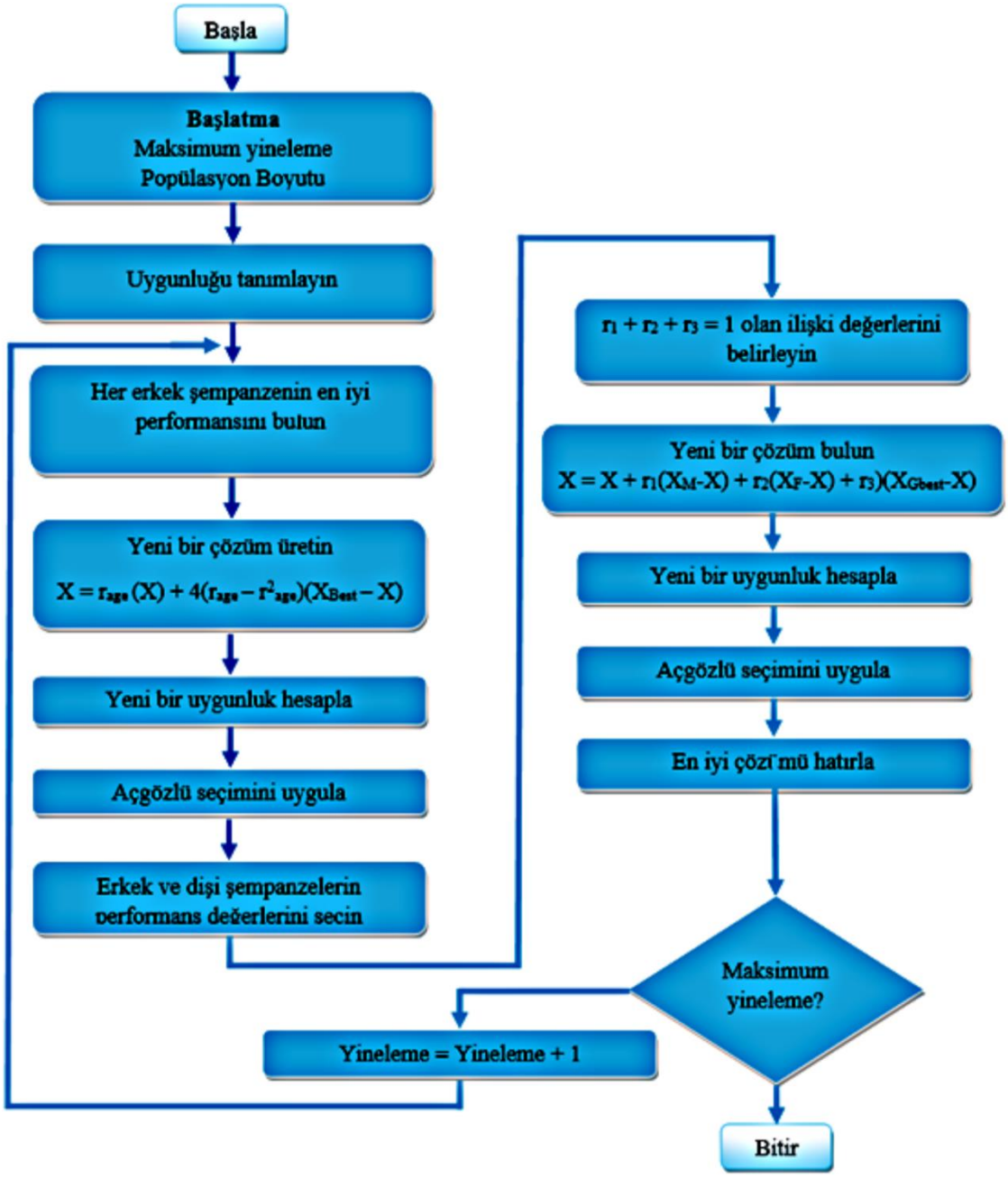
$$X < X_{LB} \Rightarrow X = X_{LB} \quad (3.32)$$

Rastgele değerler r_1, r_2 ve r_3, değerler toplanırsa, Denklem 3.33'e göre 1 üretecektir.

$$r_1 + r_2 + r_3 = 1 \quad (3.33)$$

Şekil 3.18'de bu matematiksel olarak bahsedilen ŞLSO algoritmik adımlarını gösteren akış diyagramı gösterilmektedir. ŞLSO algoritması popülasyondaki en uygun çözümü bulmak için yinelemeli bir yaklaşım kullanır. Algoritma her yinelemede her erkek ve dişi şempanze için uygunluk değerini hesaplar, en yüksek uygunluk değerine sahip erkek ve dişi şempanzeyi seçer, seçilen şempanzelerin uygunluk değerlerini kullanarak yeni bir çözüm oluşturur, yeni çözümün uygunluk değerini hesaplar. Yeni çözümü mevcut

popülasyondaki en az optimal çözümle değiştirir ve mevcut en iyi çözümü saklar. Maksimum yineleme sayısına ulaşıldığında veya en iyi çözüm belirli bir kriteri karşıladığında algoritma durur.



Şekil 3.18. ŞLSO algoritmasının akış şeması.

Çizelge 3.4'te ŞLSO'da kullanılan parametreler, açıklamaları ve değerleri verilmektedir. Tezde kullanılan algoritmanın popülasyon boyutunun değeri maksimum iterasyon değeri tabloya eklenmiştir. Bounds yani sınırlar parametresinde YOLO algoritmasındaki hiper

parametrelerin, ilk on iki hiperpametresinin alt değerlerinin en küçük değeri üst değerlerin en büyük değeri girilmiştir.

Çizelge 3.4. ŞLSO parametre ve çalışma şartları

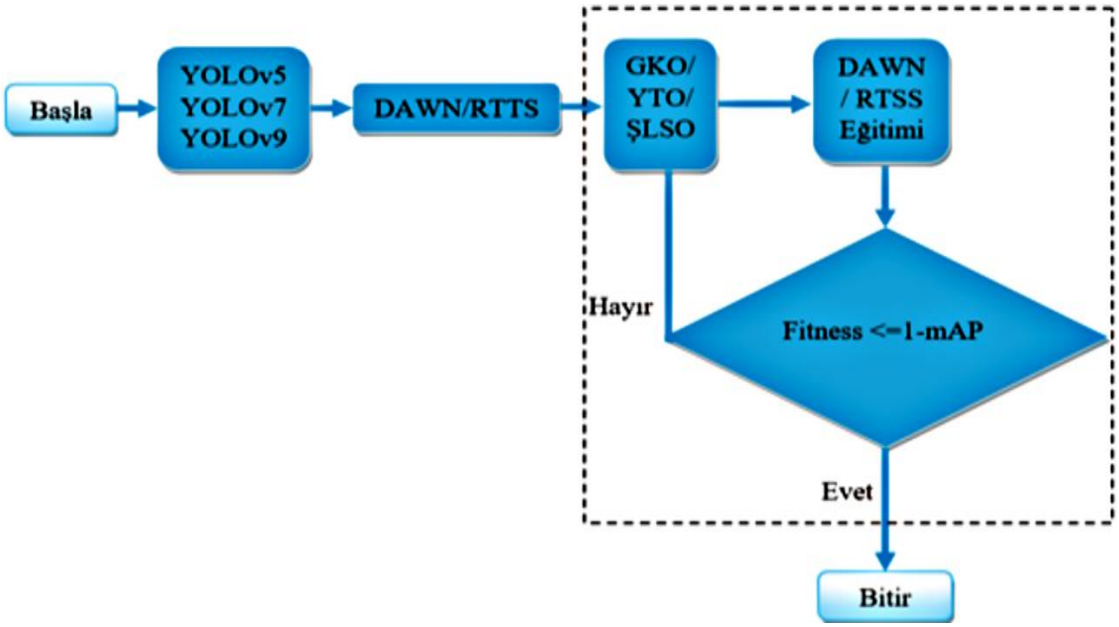
Parametre	**Açıklama**	**Değer**
Popülasyon Boyutu	Gruptaki şempanze sayısı	50
Maksimum Iterasyon	Algoritmanın çalıştırılacağı maksimum döngü sayısı	100
Bounds	Sınırlar	0,01-5
Yaş Dağılımı	Şempanzelerin yaş aralığı	0 - 50
Sosyal Bağlar	Şempanzeler arasındaki ilişkilerin gücü	0 - 1
Saldırganlık Seviyesi	Agresif etkileşimlerin sıklığı	0 - 1
Veri seti	DAWN - RTTS	-
Değerlendirme ölçütü	mAP	-
Donanım	Google COLAB V100	-

4. ÖNERİLEN YÖNTEMLER

Bu tez çalışmasında, Bölüm 3'de bahsedilen YOLOv5, YOLOv7 ve YOLOv9 modellerinin OHK'deki nesne algılama performanslarını iyileştirmek için hiperparametrelerinin optimize edilmesi önerilmektedir. Çizelge 4.1'de bahsedilen YOLO modellerinin çeşitli eğitim ayarları için kullanılan yaklaşık 30 hiperparametresi vardır. Bunlar /data/hyps dizinindeki *.yaml dosyalarında tanımlanır. Daha iyi başlangıç tahminleri daha iyi nihai sonuçlar üretecektir; bu nedenle, gelişmeden önce bu değerlerin uygun şekilde başlatılması önemlidir. YOLO modellerinde yer alan bu hiperparametrelerin ilk değerleri COCO veri setinin eğitimi için optimize edilmiş varsayılan değerlerdir. COCO veri seti birçok nesneyi içeren ve genelde temiz görüntüler içeren bir veri setidir. Bu veri setine göre ayarlanmış hiperparametre değerleri OHK altında nesne tespit konusunda aynı performansı gösterememektedir [76]. Bu sorunu çözmek için, YOLO modellerinin hiperparametrelerinin optimize edilerek OHK'de daha iyi performans gösterebilmesi hedeflenmektedir. Hiperparametrelerin optimizasyonu için farklı yaklaşımlar söz konusudur. Örneğin Izgara Arama, Rastgele Arama (RS) ve Bayesian Optimizasyon, hiper parametreleri yapılandırmak için popüler olsa da DÖ modeli daha karmaşık ve derinleştikçe veya parametre sayısı ve problemin karmaşıklığı yüksek olduğunda pratik değildirler [77]. Meta-sezgisel optimizasyon algoritmaları ise farklı problemlere uygulanabilirlik, küresel optimum bulma yeteneği, basitlik ve esneklik gibi yapısal mühendislik, yapay zekâ ve birçok alanda karmaşık optimizasyon problemlerinin çözümünde yaygın olarak kullanılmaktadır. Ayrıca daha hızlı yakınsama elde etmek ve optimuma yakın konfigürasyonu makul bir sürede bulmak için de meta-sezgisel algoritmalar kullanılmaktadır. Her yinelemede popülasyonu geliştirerek hiper parametre uzayını sistematik olarak ararlar. Bu sebeple YOLO modellerinin hiperparametrelerinin optimizasyonları için meta-sezgisel optimizasyon algoritmalarından otonom sürüşlerde daha önce tercih edilmiş GKO algoritması ve 2022 yılında çıkmış ve diğer optimizasyon algoritmalarına üstünlüğünü göstermiş olan YTO ve ŞLSO algoritmaları önerilmiştir. Hiperparametrelerin optimizasyonu YOLO modellerinin eğitiminden alınan mAP (Genel Hassasiyet Ortalaması) değerine göre hesaplandığı için her bir yeni hiperparametre değerlerinde YOLO eğitimi tekrar başlamaktadır. Bu durum optimizasyon işlemlerinin süresini oldukça uzatmaktadır. Çizelge 4.1'de 30 hiperparametre bulunmaktadır. Her birinin alt değeri ve üst değeri

arasında belli bir değer farkı vardır. Bu hiperparametrelerinin hepsinin optimize edilmeye çalışılması, YOLO modellerinin eğitim süresi de göz önünde tutulduğunda neredeyse imkânsız hale gelmektedir. Bu durum göz önünde tutularak, modelin eğitimi sırasında optimizasyon süresini kontrol eden lr0, lrf, momentum, weight_decay, warmup_epochs, warmup_momentum ve warmup_bias_lr optimizasyon hiperparametreleri ile modelin öğrenme sürecini, performansını ve aşırı uyum (overfitting) ve eksik uyum (Underfitting)'u önlemeye yarayan box, cls, cls_pw, obj ve obj_pw kayıp fonksiyonu hiperparametreleri tercih edilmiştir. Bu ilk on iki hiperparametrenin optimizasyonu [51] numaralı çalışmada da YOLOv5'in hiperparametrelerinin optimizasyonu için kullanılmıştır. Geri kalan diğer hiperparametreler yani tespit hiperparametreleri ve veri çoğaltma hiperparametreleri modelin öğrenme sürecinde kullanılan hiperparametreler olmadığı için de tercih edilmemiştir.

Modelin öğrenme süreci, duyarlılığı, dağılımı ve genel performansın etkisinden önemli ölçüde etkilenir. Bu parametreler öğrenme hızı, momentum, ağırlık azaltma, kutu sayısı, sınıf sayısı ve kayıp ağırlıkları gibi temel ayarları içerir. Bu hiperparametreler, modelin çeşitli yönlerini etkiler ve performansını maksimize etmek için optimize edilmesi gereklidir. Örneğin, öğrenme oranı (lr0) ve momentum, modelin eğitiminin ne kadar hızlı ve stabil olduğunu etkilerken, box ve cls gibi parametreler, modelin tahminlerinin doğruluğunu doğrudan etkiler. Ayrıca sadece en önemli hiperparametreleri optimize etmek, hesaplama maliyetlerini ve sürelerini düşürür. Tüm olası hiperparametrelerin optimize edilmesi yerine, en çok etkisi olanların seçilmesi, daha verimli bir optimizasyon süreci sağlar. YOLOv5, YOLOv7 ve YOLOv9 aynı altyapıyı kullandığından hiperparametreler ve değer aralıkları aynıdır. Şekil 4.1'de bu tez çalışmasının genel iş akışı süreci gösterilmektedir. Çalışmada GKO, YTO ve ŞLSO kullanarak optimizasyon için YOLOv5, YOLOv7 ve YOLOv9'un ilk on iki hiperparametresi kullanılmaktadır.



Şekil 4.1. Bu tez çalışmasında önerilen iş akışı.

Çizelge 4.1. YOLOv5, YOLOv7 ve YOLOv9'daki hiperparametreler.

Hiperparametre	Tanım	Alt sınır	Üst sınır
lr0	başlangıç öğrenme oranı (SGD=1E-2, Adam=1E-3)	0,0001	0,1
lrf	son OneCycleLR öğrenme oranı (lr0 * lrf)	0,01	0,5
momentum	SGD momentum/Adam beta1	0,6	0,99
weight_decay	optimizer ağırlık azalması 5e-4	0,00001	0,01
warmup_epochs	ısınma dönemleri (kesirler tamam)	0,0	5,0
warmup_momentum	ısınma başlangıç momentumu	0,0	0,95
warmup_bias_lr	ısınma başlangıç önyargısı lr	0,0	0,2
box	kutu kaybı kazancı	0,02	4,0
cls	cls kayıp kazancı	0,2	4,0
cls_pw	cls BCELoss pozitif_ağırlık	0,5	2,0
obj	obj kayıp kazancı (piksellerle ölçekleme)	0,2	4,0
obj_pw	obj BCELoss pozitif_ağırlık	0,5	2,0
iou_t	IoU eğitim eşiği	0,1	0,7
anchor_t	anchor-multiple eşiği	2,0	8,0
fl_gamma	odak kaybı gama (efficientDet varsayılan gama=1,5)	0,0	2,0
hsv_h	görüntü HSV-Ton artırma (kesir)	0,0	0,1
hsv_s	görüntü HSV-Doygunluk artırma (kesir)	0,0	0,9
hsv_v	görüntü HSV-Değer artırma (kesir)	0,0	0,9
degrees	görüntü döndürme (+/- derece)	0,0	45,0
translate	görüntü çevirme (+/- kesir)	0,0	0,3
scale	görüntü ölçeği (+/- kazanç)	0,5	1,5
shear	görüntü kayması (+/- derece)	0,0	10,0
perspective	görüntü perspektifi (+/- kesir)	0,0	0,001
flipud	görüntü yukarı-aşağı çevirme (olasılık)	0,0	1,0
fliplr	görüntü sola-sağa çevirme (olasılık)	0,0	1,0
mosaic	görüntü mozaiği (olasılık)	0,0	1,0
mixup	görüntü karıştırma (olasılık)	0,0	1,0
copy_paste	segment kopyala-yapıştır (olasılık)	0,0	1,0

4.1. EĞİTİM VE DEĞERLENDİRME

İlk olarak YOLOv5, YOLOv7 ve YOLOv9, DAWN ve RTTS veri kümelerindeki varsayılan parametreler kullanılarak eğitilir. Nesne algılama performansını karşılaştırmak için YOLOv5, YOLOv7 ve YOLOv9, GKO, YTO ve ŞLSO ile optimize edilmiş hiperparametreler kullanılarak eğitilir.

Şekil 4.1'de ki akış şemasında başlangıçta YOLOv5, YOLOv7 ve YOLOv9 algoritmalarında varsayılan hiperparametrelerle 50 devirlik (epoch) bir eğitim gerçekleştirilir. Daha sonra GKO ile her 50 devirde bir yenilenen hiperparametreler ile eğitim gerçekleştirilir. Bu işlemler YTO kullanılarak aynı adımlarla nesne tespit sonuçları elde edilir. Aynı şekilde ŞLSO kullanılarak aynı adımlarla nesne tespit sonuçları elde edilir. Döngü 1-mAP değeri belli bir uygunluk eşiği (fitness_threshold) altına düşmeden devam etmektedir. Uygunluk eşiğine son üç eğitimde çok yaklaştığında yani benzer mAP değerleri ürettildiğinde durmaktaır. Bu eğitim OHK'de araç tespitinin doğruluğu ve verimliliğine ilişkin bir sonuç vermektedir. Bu eğitim sonuçları, OHK'yi içeren ancak bazı farklılıklara sahip olan DAWN ve RTTS veri kümelerini kullanan YOLOv5, YOLOv7 veya YOLOv9'un nesne algılama performansının, veri kümesine ve DÖ algoritmalarına bağlı olarak değiştiğini göstermektedir.

4.2. PERFORMANS ÖLÇÜMLERİ

YOLOv5, YOLOv7 ve YOLOv9'daki nesne algılama modelinin verimliliğini değerlendirmek amacıyla bu çalışmada performans ölçümlerinde mAP değeri kullanılmıştır. mAP değeri, otonom sürüşte anlık nesne tespitinde performans ölçümü için tercih edilen metriklerden biridir. Ayrıca etiketli verilerde dengesizliklerin olduğu veri setlerinde nesne tespiti için F1-Score değerini de dikkate alan çalışmalar bulunmaktadır. F1-Score, kesinlik ve geri çağırma değerlerinin harmonik ortalaması olarak belirlenir. Çalışmamızda kullanılan açık erişimli veri setlerinde veriler dengesiz olduğundan bir diğer performans ölçütü olan "doğruluk" yanıltıcı olabileceğinden performans ölçümünde dikkate alınmamaktadır.

Algoritmalar nesneleri tespit ederken 4 farklı durum ortaya çıkıyor. Nesne doğru olarak tahmin edildiğinde Doğru Pozitif (*TP*), mevcut olmayan bir nesne yok olarak tahmin edildiğinde Doğru Negatif (*TN*), var olmayan bir nesne varmış gibi tahmin edildiğinde Yanlış Pozitif (*FP*) ve son olarak nesne mevcut olduğu halde tespit edilemediğinde Yanlış

Negatif (*FN*) olarak ifade edilmektedir. Bu değerler Hassasiyet, Geri Çağırma ve dolayısıyla F1-Score'un hesaplanmasına olanak sağlar.

Kesinlik (*Precision*) : Alınan numunelerin doğru tahmin edilmesinin alınan tüm numunelere oranına Kesinlik denir. Matematiksel olarak Kesinlik Denklem 4.1'de verilen denklemle ifade edilir.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

Geri Çağırma (Hassasiyet) (*Recall*) : TP tahminlerinin ilgili tüm örneklere oranına Geri Çağırma denir. Matematiksel olarak Geri Çağırma Denklem 4.2'de verilen denklemle ifade edilir.

$$Recall = \frac{TP}{TP+FN} \tag{4.2}$$

F1 Skoru: F1 Skoru, hassasiyet ve hatırlamanın harmonik ortalamasıdır ve hem yanlış pozitifleri hem de yanlış negatifleri hesaba katarken modelin performansının dengeli bir değerlendirmesini sağlar. Matematiksel olarak F1-Score Denklem 4.3'te verilen denklem ile ifade edilir.

$$F1\ Score = \ 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.3}$$

mAP : Özellikle araç ve yaya tespiti için nesne algılama algoritmalarının performansını ölçmek ve karşılaştırmak için en önemli parametre mAP'dir. Modelin farklı nesneleri tespit etmedeki başarısı önemlidir. mAP, modelin tüm sınıflardaki ortalama hassasiyet performansını belirler. Bu veriler modelin genel nesne algılama performansını gösterir. Matematiksel olarak mAP Denklem 4.5'te verilen denklemle ifade edilir.

$$AP = \frac{1}{m} \sum_{\substack{r=0.0 \\ step\ 0.1}}^{1.0} p(r) \tag{4.4}$$

$$mAP = \frac{1}{k} \sum_{i}^{k} AP_i \tag{4.5}$$

4.3. UYGULAMA ORTAMI

Modeli eğitmek için Google Colab'ın güçlü GPU'larından ve yüksek RAM'inden V100 GPU seçildi. Görseller eğitime dahil edilirken YOLOv7 standardı için 640 x 640 olarak

yeniden boyutlandırıldı. Toplu iş boyutu 8'e ayarlandı. Varsayılan hiperparametrelerle 50 devirlik eğitim 1 saat 7 dakika sürdü. GKO ile optimize edilmiş hiperparametrelerle eğitim, en iyi sonuçlar bulunana kadar devam etti. Benzer şekilde YTO ile eğitime en iyi sonuçlar elde edilene kadar devam edildi. Tüm algoritmaların kod uygulamaları Python'da geliştirildi. YOLOv5, YOLOv7, YOLOv9, GKO, YTO ve ŞLSO algoritmalarının Python kodları resmi depolardan alınmıştır. Bu, sonuçların güvenilirliğini ve uygulanabilirliğini sağlamıştır.

Bu çalışmaları gerçekleştirmek için kullanılan bilgisayar sisteminde Core i7 10. Nesil, Nvidia RTX 3060 6 GB GDDR6, 32 GB DDR4 3,2 GHz RAM ve 1 TB SSD bulunmaktadır.

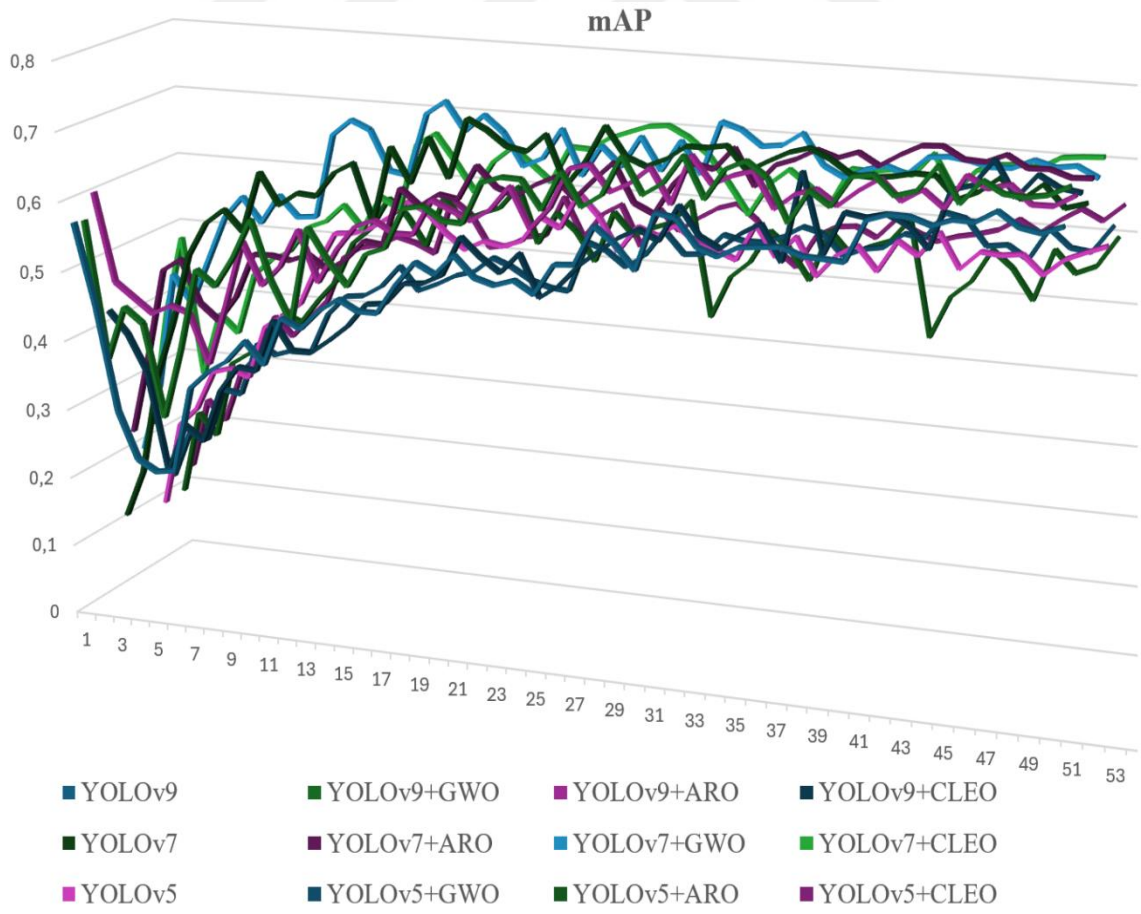
5. SONUÇLAR VE TARTIŞMA

5.1. YOLO MODELLERİNİN DAWN ÜZERİNDEKİ PERFORMANSI

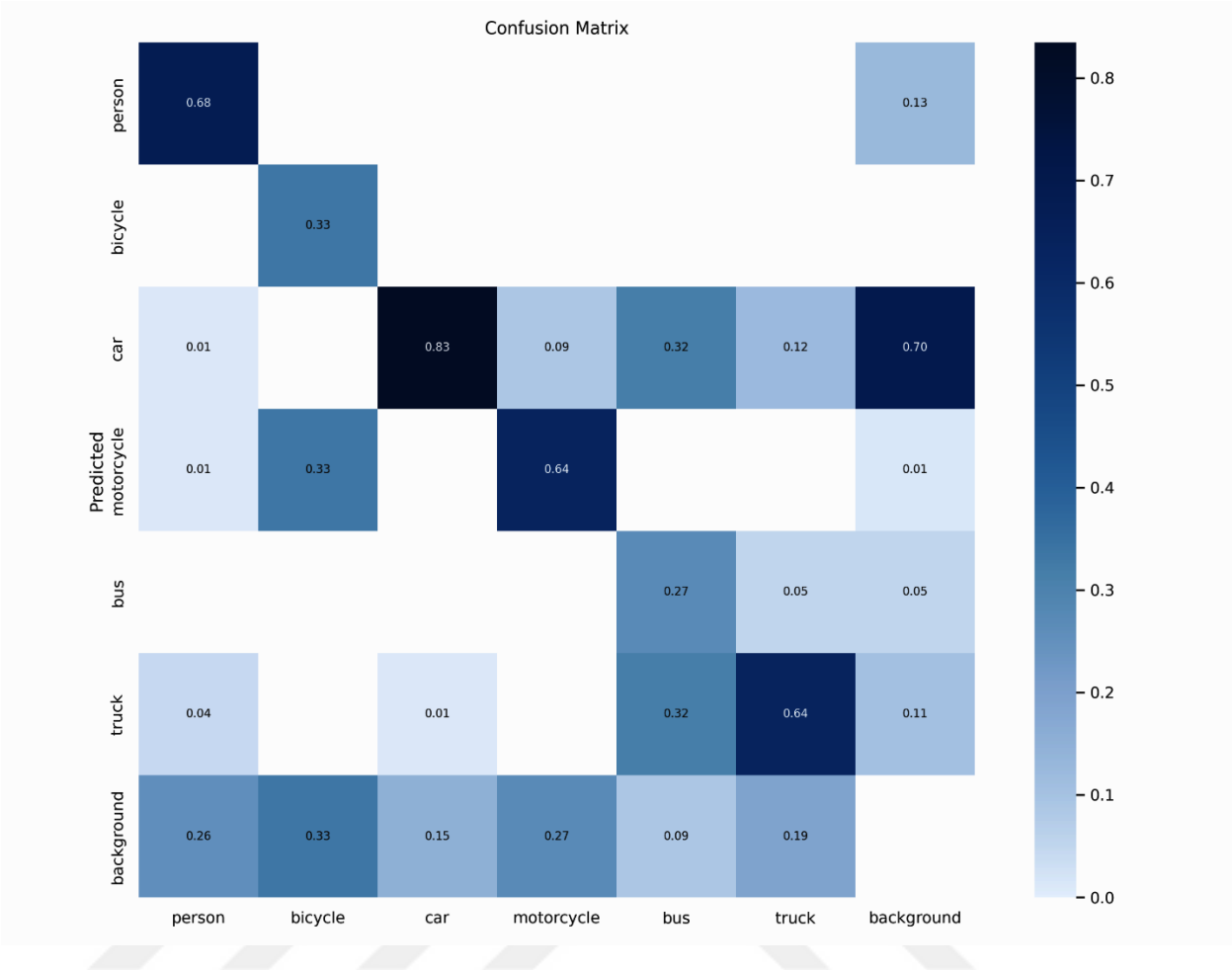
Çizelge 5.1'de DAWN veri kümesi kullanılarak mAP tipinde orijinal hiperparametreler, GKO, YTO ve ŞLSO optimizasyon algoritmaları ile YOLOv5, YOLOv7 ve YOLOv9 modellerinin eğitim sonuçları gösterilmektedir. Şekil 5.1, 50 devir sonrasında DAWN veri setini kullanan tüm eğitimin grafiksel sonuçlarını göstermektedir. Makalede bahsedilen DAWN veri setini kullanan diğer çalışmaların sonuçlarına da yer verilmiştir. [49]'daki makalede orijinal DAWN veri seti kullanılarak eğitim gerçekleştirilmiştir. Sisli, yağmurlu, karlı ve kumlu hava durumlarında ayrı ayrı eğitilerek elde edilen mAP sonuçlarında bile en fazla mAP değeri %55,32 olarak tespit edilmiştir. Tüm hava durumlarına göre ortalama mAP değeri %47,79 elde edilmiştir. [32]'deki makalede transfer öğrenme yöntemi kullanarak ince ayar yaptıkları YOLOv5 modeliyle DAWN veri seti üzerine eğitim gerçekleştirdiklerinde mAP sonuçları tüm hava koşullarının ortalama değeri %34,50'dir. En iyi mAP değeri yağmur altındaki görüntülerde elde edilmektedir. Yağmur altındaki mAP değeri %41,21'dir. [76]'daki çalışmada renk artırma ile güncelledikleri DAWN veri seti RetinaResnet50, YOLOv3 + RetinaResnet50, RetinaResnet50 + SSD ve YOLOv3 + Retinaresnet50 + SSD modelleriyle eğitilmiştir. Elde edilen mAP sonuçlar Çizelge 5.1'de görüldüğü gibidir. En iyi sonuç %32,75 mAP değeri ile RetinaResnet50 modeli tek başına kullanıldığında çıkmıştır. Çizelge 5.1 incelendiğinde orijinal hiperparametrelere sahip YOLOv5, YOLOv7 ve YOLOv9 modellerinin eğitim sonuçlarının bile diğer çalışmalardaki sonuçlardan daha iyi bir sonuca sahip oldukları görülmektedir. Sırasıyla YOLOv5 %60,20, YOLOv7 %68,50 ve YOLOv9 %68,00 olduğu görülmektedir. Bu sonuçlara göre yeni çıkan YOLO modellerinin nesne tespit işlemlerinden oldukça başarılı olduğu sonucunu göstermektedir. YOLO modellerinin hiperparametrelerinin optimizasyon algoritmaları ile optimize edilmesi sonrasındaki eğitim sonuçları incelendiğinde ise farklı sonuçlar görülmektedir. Örneğin ŞLSO, YOLOv5 ve YOLOv7'de en iyi performans artışını sağlarken, GKO YOLOv9'da en iyisidir. YOLOv5 + ŞLSO, %6,146 performans artışıyla %63,90 mAP değerine sahiptir. YOLOv7 + ŞLSO, %6,277 performans artışıyla %72,80 mAP'ye sahiptir. YOLOv9 + GKO, %6,764 performans artışıyla %72,60 mAP'ye sahiptir. Çizelge 5.1'e göre YOLOv9 algoritmasında en iyi mAP değeri YOLOv7 + ŞLSO iken en iyi performans artışı YOLOv9 + GKO ile gerçekleştirilmektedir.

Çizelge 5.1. DAWN veri setindeki mAP (%) karşılaştırması.

Model	**DAWN (mAP (%))**
YOLOv3 + AWBLP [49]	47,79
pre-trained YOLO-v5 [32]	34,50
RetinaResnet50 [78]	32,75
YOLOv3 + RetinaResnet50 [78]	25,68
RetinaResnet50 + SSD [78]	26,20
YOLOv3 + RetinaResnet50 + SSD [78]	25,03
YOLOv5	60,20
YOLOv5 + GKO	**62,30**
YOLOv5 + YTO	**60,10**
YOLOv5 + ŞLSO	**63,90**
YOLOv7	68,50
YOLOv7 + GKO	**70,60**
YOLOv7 + YTO	**71,20**
YOLOv7 + ŞLSO	**72,80**
YOLOv9	68,00
YOLOv9 + GKO	**72,60**
YOLOv9 + YTO	**70,80**
YOLOv9 + ŞLSO	**70.50**



Şekil 5.1. DAWN veri setindeki 50 devir için mAP sonuçları.

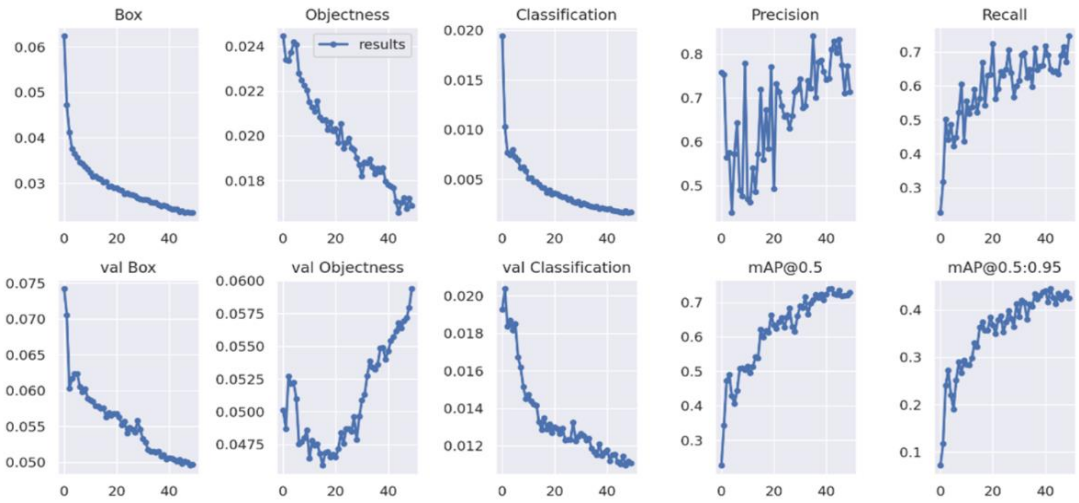


Şekil 5.2. Karışıklık matrisi.

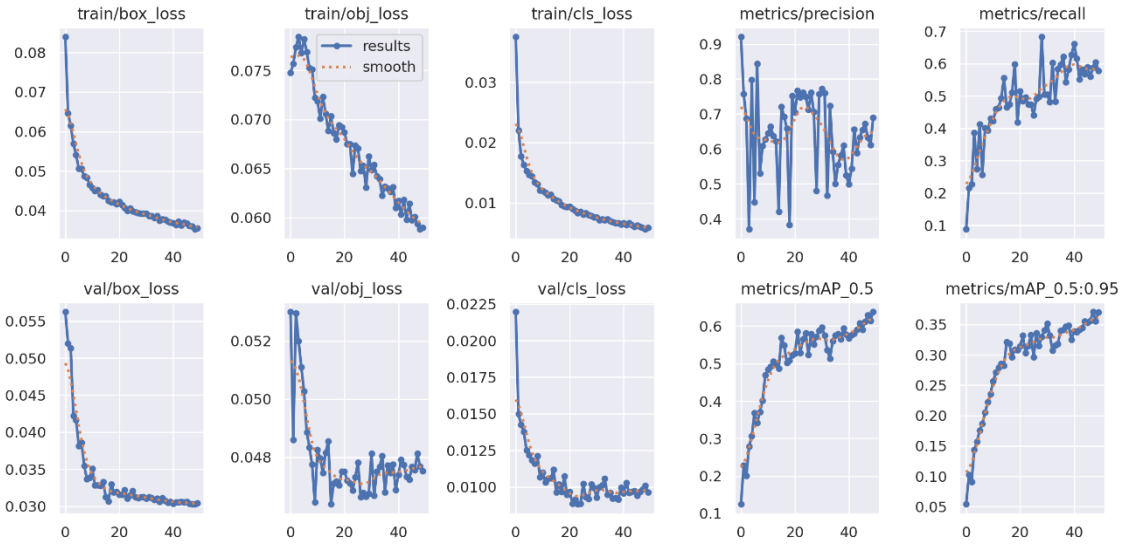
Şekil 5.2 yalnızca bir algoritmanın eğitilmesinden kaynaklanan bir Karışıklık Matrisi değeridir. Diğer tüm Karışıklık Matrisleri Ek 1'de yer almaktadır. Çizelge 5.2'deki değerler 12 adet Karışıklık Matrisi değerinin tek tek hesaplanmasıyla oluşturulmuştur. Bu değerler DAWN veri setindeki 6 nesne için ayrı ayrı hesaplanmakta ve uygulanan her model için farklı sonuçlar elde edilmektedir. Eğitim boyunca meydana gelen tüm değişiklikleri gösteren grafiksel veri örneği Şekil 5.3 ve Şekil 5.4'de gösterilmektedir. Şekil 5.3 YOLOv9 için ŞLSO Parametreleriyle Karışıklık Matrisi Diyagramı. Çizelge 5.2'deki değerler tüm nesnelerin ortalamasına göre analiz edildiğinde F1-Score için en iyi model YOLOv7 + YTO'dur. Otonom sürüşte nesnenin tespiti ve konumlandırılması daha önemli olduğundan öncelikle mAP değerleri dikkate alınır. Dolayısıyla YOLOv7 + ŞLSO ve YOLOv9 + GKO modellerinin nesne tespiti ve lokalizasyonu konusunda diğer modellere göre üstün olduğu görülmektedir.

Çizelge 5.2. DAWN veri seti ile sonuçlar (%).

		Person	Bicycle	Car	Motorcycle	Bus	Truck
YOLOv5	**Precision**	88,70	0,00	49,69	95,74	94,73	91,80
	Recall	57,89	0,00	82,82	62,50	72,00	73,68
	F1-Score	70,05	0,00	62,11	75,62	81,81	81,74
YOLOv5 + GKO	**Precision**	81,42	1,00	53,94	94,82	91,83	84,21
	Recall	61,29	50,00	82,82	67,07	76,27	77,10
	F1-Score	69,93	66,66	65,33	78,56	83,32	80,49
YOLOv5 + YTO	**Precision**	86,76	0,00	51,87	96,96	88,88	88,23
	Recall	62,76	0,00	83,83	78,04	64,00	80,00
	F1-Score	72,83	0,00	64,08	86,47	74,41	83,91
YOLOv5 + ŞLSO	**Precision**	85,33	89,47	51,57	94,11	95,34	92,85
	Recall	67,36	34,00	82,82	70,32	69,49	75,58
	F1-Score	75,28	49,27	63,56	80,49	80,38	83,32
YOLOv7	**Precision**	90,66	1,00	51,86	97,33	96,49	87,01
	Recall	68,68	1,00	84,69	80,22	78,57	77,01
	F1-Score	78,15	1,00	64,34	87,95	86,61	81,71
YOLOv7 + GKO	**Precision**	87,65	99,00	54,38	95,52	92,85	88,15
	Recall	71,71	1,00	87,87	70,33	91,23	81,70
	F1-Score	78,88	99,49	67,18	81,01	92,03	84,80
YOLOv7 + YTO	**Precision**	88,09	99,00	55,00	95,34	92,15	87,84
	Recall	74,74	1,00	89,79	82,00	79,66	84,42
	F1-Score	80,86	99,49	68,21	88,16	85,45	86,09
YOLOv7 + ŞLSO	**Precision**	88,75	99,00	53,12	95,89	96,34	87,83
	Recall	71,00	1,00	86,73	70,00	84,94	80,24
	F1-Score	78,88	99,49	65,88	80,92	90,28	83,86
YOLOv9	**Precision**	82,43	1,00	51,57	96,96	96,15	91,17
	Recall	64,89	79,76	83,67	70,32	78,12	72,09
	F1-Score	72,61	88,74	63,81	81,51	86,20	80,51
YOLOv9 + GKO	**Precision**	87,67	1,00	51,85	97,33	93,75	90,90
	Recall	67,36	79,76	84,84	73,00	76,27	80,45
	F1-Score	76,18	88,74	64,36	83,42	84,11	85,35
YOLOv9 + YTO	**Precision**	91,17	1,00	49,41	98,64	96,15	95,89
	Recall	65,26	50,00	85,85	80,21	78,12	80,45
	F1-Score	76,06	66,66	62,72	88,47	86,20	87,49
YOLOv9 + ŞLSO	**Precision**	83,95	1,00	54,24	98,46	84,37	85,33
	Recall	72,34	50,00	84,69	70,32	75,00	77,10
	F1-Score	77,71	66,66	66,12	82,04	79,40	81,00



Şekil 5.3. YOLOv7 + ŞLSO sonuçları.



Şekil 5.4. YOLOv5 +ŞLSO sonuçları.

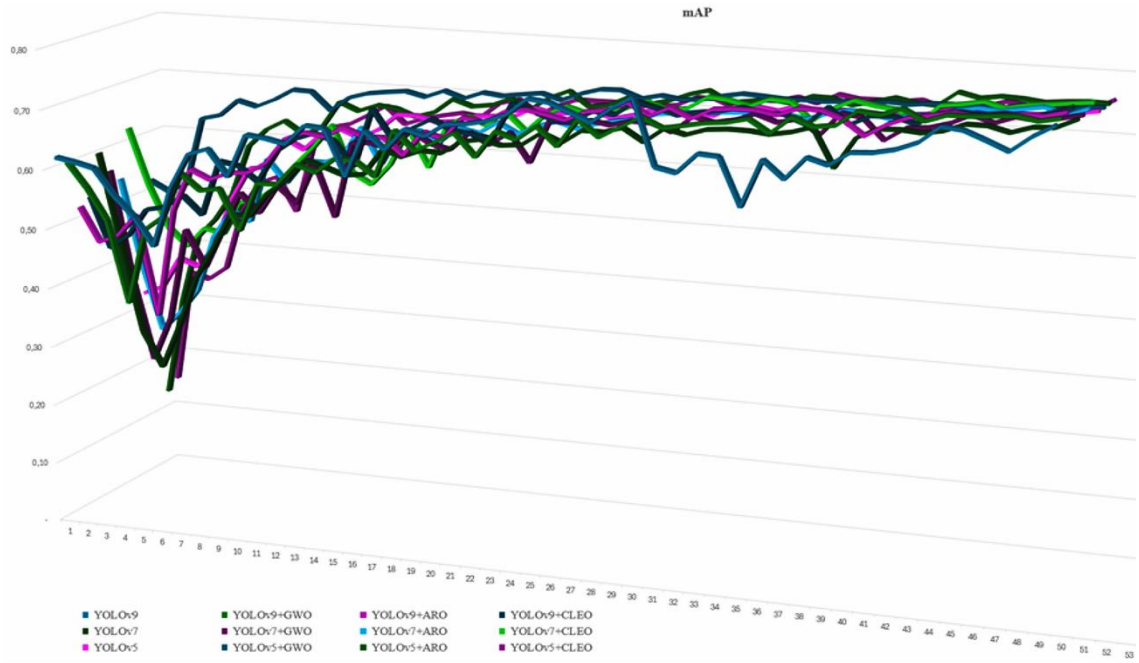
5.2. YOLO MODELLERİNİN RTTS ÜZERİNDEKİ PERFORMANSI

Çizelge 5.3, GKO, YTO ve ŞLSO optimizasyon algoritmalarıyla orijinal ve optimize edilmiş hiper parametrelerle RTTS veri kümesini kullanan YOLOv5, YOLOv7 ve YOLOv9 modellerinin eğitim mAP sonuçlarını göstermektedir. Çizelge 5.3'te ayrıca makalede bahsedilen RTTS veri setini kullanan diğer çalışmaların sonuçları da gösterilmektedir. Sonuçlar analiz edildiğinde [25]'deki çalışmada kötü hava koşulları düzelterek gizli bilgileri ortaya çıkaran farklılaşmış bir görüntü işleme modülü ortaya çıkarmıştır. IA-YOLO modeli geliştirilerek RTTS veri seti üzerinde eğitim gerçekleştirildiğinde alınan mAP sonuc %52 olmuştur. [19]'da TogetherNet adı verilen, OHK'de nesneleri ayırt etmek için öğrenmeyi dinamik olarak geliştirerek bu iki alt görevi birbirine bağlayan etkili ancak birleştirilmiş bir algılama paradigması önerilmektedir. RTTS veri seti kullanılarak yapılan eğitim sonucunda alınan mAP değeri %61,55 olmuştur. [79]'de geliştirilmiş YOLOv5'e dayanan çalışma, sisli sürüş sahneleri için çoklu nesne algılama ağı sağlar. Geliştirilmiş YOLOv5'e dayanan sisli sürüş için bir algılama ağı sunulmaktadır. Yapısal yeniden parametrelendirmeyle değiştirilen ResNeXt modeli, modelin omurgasını oluşturmaktadır. RTTS veri seti üzerinde eğitim yapan bu YOLOv5-Fog modelinin mAP değeri tüm YOLOv5 ve YOLOv7 modellerinden daha yüksek olduğu görülmektedir. Tüm Çizelge 5.3'teki değerler incelendiğinde ise en iyi mAP değerine sahip modeller YOLOv9 modelleridir.

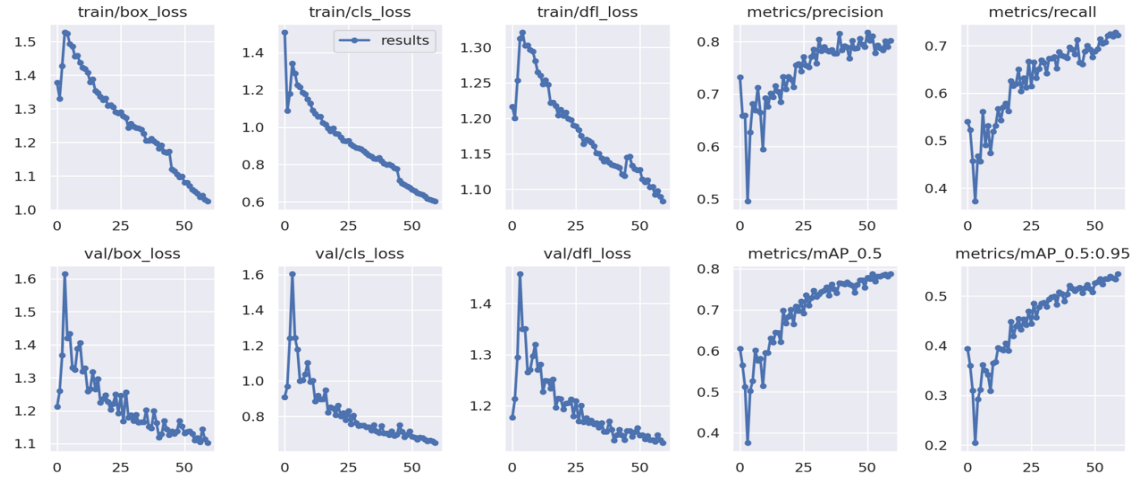
Çizelge 5.3. RTTS veri setindeki mAP (%) karşılaştırması.

Model	**RTTS (mAP)**
IA-YOLO [25]	52,00
TogetherNet [19]	61,55
Faster-RCNN [79]	51,30
RetinaNet [78]	53,10
YOLOv5-Fog [79]	77,80
YOLOv5	75,60
YOLOv5 + GKO	**75,60**
YOLOv5 + YTO	**75,60**
YOLOv5 + ŞLSO	**75,59**
YOLOv7	76,80
YOLOv7 + GKO	**76,90**
YOLOv7 + YTO	**77,30**
YOLOv7 + ŞLSO	**77,60**
YOLOv9	78,40
YOLOv9 + GKO	**78,80**
YOLOv9 + YTO	**79,20**
YOLOv9 + ŞLSO	**79,30**

Çizelge 5.3., RTTS veri seti kullanılarak YOLOv5, YOLOv7 ve YOLOv9'un orijinal hiperparametreleri ile eğitildiğinde mAP sonuçlarının sırasıyla %75,60, %76,80 ve %78,40 olduğunu göstermektedir. YOLOv5 ile kullanılan optimizasyon algoritmalarının YOLOv5'teki mAP sonucunu hiçbir şekilde etkilemediği görülmektedir. Orijinal YOLOv7 modeline kıyasla YOLOv7 + GKO modeli %0,13 performans artışı, YOLOv7 + YTO modeli %0,65 performans artışı ve YOLOv7 + ŞLSO modeli %1,04 performans artışı gerçekleştirmektedir. YOLOv9 + GKO modeli, orijinal YOLOv7 modeline göre %0,51, YOLOv9 + YTO modeli %1,02, YOLOv9 + ŞLSO modeli ise orijinal YOLOv9 modellerine göre %1,14 performans artışı gerçekleştirmekte, orijinal YOLOv9 modeline göre YOLOv9 + GKO modeli %0,51, YOLOv9 + YTO modeli %1,02, YOLOv9 + ŞLSO modeli ise %1,14 performans artışı gerçekleştirmektedir. Ayrıca Şekil 5.5'de RTTS veri setini kullanan tüm eğitimlerin 50 devir sonrasındaki grafiksel sonuçları gösterilmektedir. Eğitim boyunca meydana gelen tüm değişiklikleri gösteren grafiksel verilere bir örnek Şekil 5.6 ve Şekil 5.7'de gösterilmektedir.



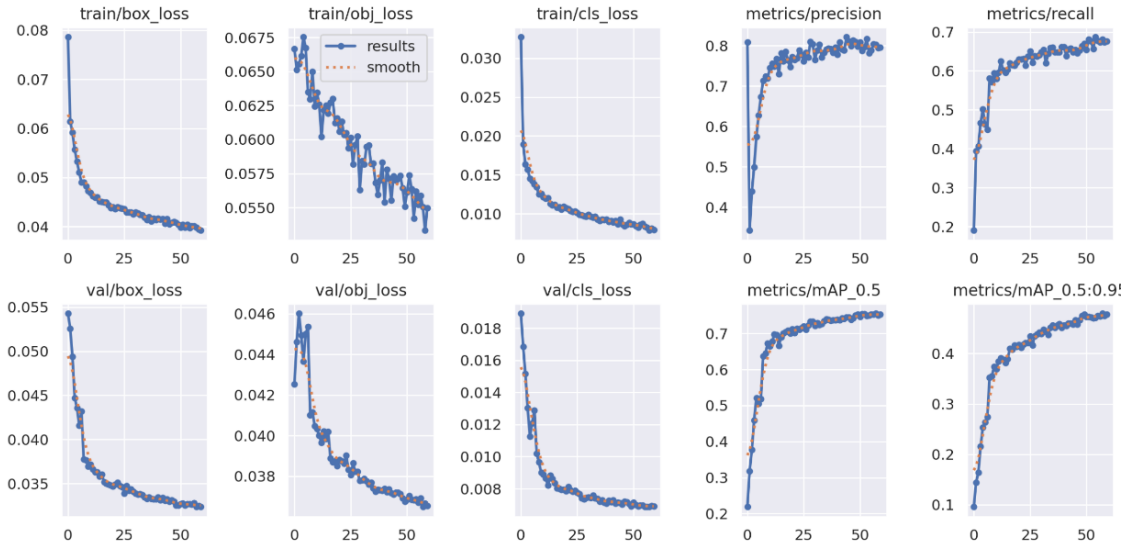
Şekil 5.5. RTTS veri kümesinde 50 devir için mAP sonuçları.



Şekil 5.6. YOLOv9 + GKO sonuçları.

Çalışmamızda hiper parametre optimizasyonu ile nesne tespit performansının artması, özellikle OHK'nin ve sadece trafik yollarının dahil edilmesinde amaçlanan hedefe ulaşıldığını göstermektedir. Aradaki fark, yalnızca OHK üzerinde eğitim almamız ve hiperparametre optimizasyonu için GKO, YTO ve ŞLSO'yu kullanmamızdır. [80]'de YOLOv5'in on iki hiper parametresi, DBOA optimizasyon algoritması kullanılarak nesne tespiti için kullanılmıştır. Kullanılan KITTI [81] veri seti genellikle normal, temiz görüntüler içerdiğinden OHK'de nasıl performans göstereceği bilinmemektedir. Bu nedenle normal hava koşullarına sahip veri kümesini kullanır ve OHK'yi ele almaz.

Ayrıca nesne tespit modeli olarak, yalnızca YOLOv5'i kullanır.



Şekil 5.7. YOLOv5 + GKO sonuçları.

DAWN veri setini varsayılan parametrelerle eğittikten sonra, Şekil 5.8'de test için sisteme sunulan bazı görüntüler gösterilmektedir. Birinci sıra ve üçüncü sütunda bazı araçlar algılanmıyor. İkinci sıranın ilk sütununda karşıdan gelen araç algılanmıyor. Diğer örneklerde ilk satır, dördüncü sütunda olmayan nesneleri algılar. GKO optimizasyonlu hiperparametrelerle eğitilmiş DAWN veri setinden test amaçlı olarak sisteme girilen görsellerden bazıları Şekil 5.9'da gösterilmektedir. Şekil 5.8 ile karşılaştırıldığında ilk sıra ve dördüncü sütunda yer alan kamyon daha yüksek oranda tahmin edilmekte ve tespit edilmektedir. ve birinci sırada ve dördüncü sütunda olmayan bir araç, araç olarak etiketlenmez. İkinci sıra ve ilk sütunda karşıdan gelen araç etiketlenir. YTO için optimize edilmiş hiperparametrelerle eğitilmiş DAWN veri setinden elde edilen bir sonraki test veri görüntüleri seti Şekil 5.10'da gösterilmektedir. Şekil 5.8. ile karşılaştırıldığında, ilk satır ve üçüncü sütundaki kum fırtınası görüntüsünde tüm arabalar algılanır ve etiketlenir. İkinci sıra ve birinci sütunda karşıdan gelen araç ile dördüncü sıra ve dördüncü sütunda yer alan kamyon Şekil 5.8'deki gibi etiketlenmemiştir. ŞLSO için optimize edilmiş hiperparametrelerle eğitilmiş DAWN veri setinden sonraki test veri görüntüleri Şekil 5.11'de gösterilmektedir. Şekil 5.8'de, ilk satır ve üçüncü sütundaki kum fırtınası görüntüsünde tüm arabalar algılanır ve etiketlenir. Aynı görüntüde, bulunmayan bir araç etiketlenmiştir. Çizelge 5.5'deki mAP değerlerine bakıldığında özellikle YOLOv5 ve YOLOv9 için en küçük mAP değeri %68,00, en büyük değeri ise %72,60'dır. Şekillerde etiketlenen nesnelere bakıldığında algılama işlemlerinin mAP değerlerine uygun olarak

yapıldığı görülmektedir. Çizelge 5.4, tüm OHK'yi kapsadığı için özellikle çalışmamız olmak üzere DAWN veri setini kullanan tüm eğitimler için uygulanan yöntemlerin yineleme oranını, eğitim süresini, avantajlarını ve dezavantajlarını göstermektedir.

Çizelge 5.4. Nesne algılama performansı ve verimlilik karşılaştırmaları.

Metot	**mAP(%)**	**Devir Hızı (it/s)**	**50 Devir**	**Avantaj**	**Dezavantaj**
YOLOv5	60,20	1,97	0,661	Gerçek zamanlı uygulamalar için daha hızlı yanıt sağlar.	Nesne tespitinde düşük doğruluk
YOLOv5+GKO	**62,30**	2	0,58	Yüksek doğruluk gerektiren uygulamalar için idealdir.	Daha fazla işlem gücü ve enerji tüketimi
YOLOv5+YTO	**60,10**	1,06	0,63	Düşük eğitim süresi	Nesne tespitinde düşük doğruluk
YOLOv5+ŞLSO	**63,90**	1,88	0,61	Gerçek zamanlı uygulamalar için daha hızlı yanıt sağlar.	Nesne Algılama Doğruluğu yetersiz
YOLOv7	68,50	1,71	0,63	Hem hız hem de eğitim süresi açısından verimli, dengeli bir model.	Diğer YOLOv7 varyantlarına kıyasla daha düşük F1 Puanı
YOLOv7+GKO	**70,60**	1,55	0,63	İyi bir F1-Score ile yüksek doğruluk sağlar.	Nispeten düşük hız. Bu, gerçek zamanlı uygulamalarda bazı gecikmelere neden olabilir.
YOLOv7+YTO	**71,20**	1,38	0,64	yüksek doğruluk gerektiren uygulamalar için idealdir.	Nesne tespitinde düşük doğruluk
YOLOv7+ŞLSO	**72,80**	1,22	0,65	Hem hız hem de antrenman için iyi bir denge sağlar.	Ara F1 Skoru en yüksek doğruluğu sağlamaz.
YOLOv9	68,00	1,06	0,65	Düşük eğitim süresi	F1-Score ortalaması diğer YOLOv9 varyantlarına göre daha düşüktür.
YOLOv9+GKO	**72,60**	1,11	0,66	İyi bir F1 Puanı ve orta hız	Eğitim süresi diğer YOLOv9 varyantlarından daha uzun olabilir.
YOLOv9+YTO	**70,80**	1,42	0,67	Yüksek hız ve yüksek doğruluk	Eğitim süresi biraz daha uzun
YOLOv9+ŞLSO	**70.50**	1,05	0,66	Düşük eğitim süresi ve iyi bir denge sunar.	Düşük hız, bazı uygulamalarda performansın düşmesine neden olabilir.

Şekil 5.8. Orijinal hiperparametrelerle test sonucu.

Şekil 5.9. GKO optimize edici hiperparametreleriyle test sonucu.

Sonuçlara genel olarak bakıldığında en yüksek mAP değerine sahip olan YOLOv7 + ŞLSO veya YOLOv9 + GKO ile optimize edilen modeller yüksek doğruluk gerektiren uygulamalarda kullanılabilir. Kullanılan modeller hava şartlarına göre değişiklik gösterebilir. Ancak YOLOv7 veya YOLOv9 modelleri kullanılarak hava koşullarından bağımsız olarak en iyi mAP sonuçları elde edilebilmektedir. Tahmin görüntüleri detaylı

olarak incelendiğinde GKO, YTO ve ŞLSO ile ayrı ayrı optimize edilen YOLO modellerinin özellikle tozlu hava koşullarında OHK veri kümeleri kullanıldığında araçların tespiti. Her üç eğitim konfigürasyonu da (GKO, YTO, ŞLSO ve Orijinal), tozlu

Şekil 5.10. YTO optimize edici hiperparametrelerle test sonuçları.

Şekil 5.11. ŞLSO optimize edici hiperparametrelerle test sonuçları.

havaların ötesindeki senaryolar da dahil olmak üzere, hava koşullarından bağımsız olarak nesne tespitinde başarı göstermektedir.

Çizelge 5.5. YOLOV5 YOLOv7 ve YOLOv9'da optimize edilmiş ve orijinal ilk 6 hiperparametre.

	lr0	**lrf**	**momentum**	**weight_decay**	**warmup_ epochs**
YOLOv5	0,0100	0,0100	0,9370	0,0005	3,0000
YOLOv5 + GKO	**0,0124**	**0,0129**	**0,9781**	**0,0008**	**3,2113**
YOLOv5 + YTO	**0,0100**	**0,0100**	**0,937**	**0,0005**	**3,0000**
YOLOv5 + ŞLSO	**0,0115**	**0,0109**	**0,9556**	**0,0007**	**3,2030**
YOLOv7	0,0100	0,1000	0,937	0,0005	3,0000
YOLOv7 + GKO	**0,0123**	**0,1286**	**0,9781**	**0,0008**	**3,2113**
YOLOv7 + YTO	**0,0120**	**0,1176**	**0,9406**	**0,0006**	**3,0755**
YOLOv7 + ŞLSO	**0,0114**	**0,0125**	**0,9638**	**0,0006**	**3,0110**
YOLOv9	0,0100	0,0100	0,9370	0,0005	3,0000
YOLOv9 + GKO	**0,0106**	**0,0107**	**0,9472**	**0,0006**	**3,0526**
YOLOv9 + YTO	**0,0118**	**0,0129**	**0,9389**	**0,0006**	**3,1323**
YOLOv9 + ŞLSO	**0,0116**	**0,0129**	**0,9737**	**0,0006**	**3,0202**

Bu bulgular, YOLOv7 ve YOLOv9 algoritmalarının OHK'deki nesneleri tespit etmedeki etkinliğini vurguluyor ve GKO, YTO veya ŞLSO optimizasyon algoritmalarıyla birleştirildiğinde daha da iyi performans gösteriyor. Ek 2'de geri kalan nesne tespit görüntüleri yer almaktadır. Meta-sezgisel optimizasyon algoritmaları tarafından optimize edilen YOLO sürüm 5, 7 ve 9 modellerinin ilk 12 hiper parametresinin sonuçları ve orijinal değerleri, Çizelge 5.5 ve Çizelge 5.6'daki iki tabloda gösterilmektedir.

Çizelge 5.6. YOLOv5, YOLOv7 ve YOLOv9'da optimize edilmiş ve orijinal 7 - 12 hiperparametre.

	Warmup_momentum	**warmup_ bias_lr**	**box**	**cls**	**cls_pw**	**obj**
YOLOv5	0,8000	0,1000	0,0500	0,5000	1,0000	1,0000
YOLOv5 + GKO	**0,8956**	**0,0991**	**0,0496**	**0,5956**	**1,1816**	**1,2008**
YOLOv5 + YTO	**0,8000**	**0,1000**	**0,0500**	**0,5000**	**1,0000**	**1,0000**
YOLOv5 + ŞLSO	**0,8658**	**0,0972**	**0,0425**	**0,5692**	**1,0823**	**1,0925**
YOLOv7	0,8000	0,1000	0,0500	0,3000	1,0000	0,0700
YOLOv7 + GKO	**0,0991**	**0,8956**	**0,7956**	**0,3956**	**1,1816**	**0,0495**
YOLOv7 + YTO	**0,0960**	**0,8387**	**0,7303**	**0,3704**	**1,1298**	**0,0472**
YOLOv7 + ŞLSO	**0,8008**	**0,8100**	**0,0512**	**0,4000**	**1,1280**	**0,0470**
YOLOv9	0,8000	0,1000	7,5000	0,5000	1,0000	0,7000
YOLOv9 + GKO	**0,8238**	**0,0848**	**0,0424**	**0,5238**	**1,0452**	**1,0500**
YOLOv9 + YTO	**0,8118**	**0,0852**	**0,0467**	**0,5975**	**1,1680**	**1,0945**
YOLOv9 + ŞLSO	**0,8208**	**0,0802**	**0,0497**	**0,5680**	**1,1760**	**1,1497**

6. SONUÇ VE GELECEK ÇALIŞMALAR

Bu çalışmada, YOLO modellerinin nesne algılama performansları, özellikle OHK'de otonom sürüşte nesne algılama için kullanılan YOLO modellerinin hiperparametreleri optimize edilmiştir. Veri setleri karşılaştırıldığında, özellikle sadece trafik görselleri ve sadece OHK'li görsellerin yer aldığı DAWN veri setinde, YOLO modellerinin optimizasyon algoritmaları ile optimize edilmesiyle performansta %6'nın üzerinde bir artış gözlemlenmektedir. RTTS veri kümesinde YOLO modellerinin optimize edilmesi, gözle görülür bir performans artışı göstermemektedir. RTTS veri setinde genellikle tek bir OHK'nn (puslu/sisli) bulunması ve sadece trafik verilerini içermemesi, RTTS veri setini DAWN veri setinden ayırmaktadır. DAWN veri setinde YOLOv5 modeli GKO, YTO veya ŞLSO ile optimize edilmiş olsa bile daha yeni YOLOv7 ve YOLOv9'dan daha iyi performans göstermez. Ancak YOLOv7 modeli ve YOLOv9 modeli ile tüm eğitim sonuçlarının mAP değerleri birbirine yakındı. Bu sonuçlara göre OHK içeren YOLO modelleri otonom araçlarda nesne tespitinde iyi performans göstermektedir. YOLO, OHK'de optimizasyon algoritmalarını kullanarak optimize edilmiş hiperparametreler ile çok daha iyi performans gösterir. Veri setlerinde etiketlenen nesneler dengeli olmadığından bazı nesnelerin tespitinde zorluklar yaşanmaktadır. Bu sorunu çözmek için OHK'de bisiklet, motosiklet, otobüs gibi nesnelerin yer aldığı görsellerin bulunması ve bu nesnelerin etiketlenmesi gerekmektedir. Gelecek çalışmalarda farklı optimizasyon algoritmalarını birleştirmeyi ve YOLO modelinin aktivasyon fonksiyonlarında değişiklikler yaparak YOLO modellerinin OHK'deki nesne algılama performansını daha iyi seviyelere çıkarmayı hedeflenmektedir.

7. KAYNAKLAR

[1] C. Ozan, Ö. Başkan, S. Haldenbilen, and E. Derici, "Trafik kazalarının tehlike indeksi metodu ile analizi: Denizli örneği," *Mühendislik Bilimleri Dergisi*, 2010.

[2] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[3] N. Alam, (2024, 30 Mart). [Online]. Erişim: https://medium.com/@nahidalam/understanding-yolov7-neural-network-343889e32e4e

[4] H. Nadeem *et al.*, "Road feature detection for advance driver assistance system using deep learning," *Sensors*, vol. 23, no. 9, p. 4466, 2023.

[5] A. Lincy, G. Dhanarajan, S. S. Kumar, and B. Gobinath, "Road Pothole Detection System," in *ITM Web of Conferences*, EDP Sciences, 2023.

[6] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," *arXiv preprint arXiv:2402.13616*, 2024.

[7] J. Liu, X. Wei, and H. Huang, "An improved grey wolf optimization algorithm and its application in path planning," *IEEE Access*, vol. 9, pp. 121944–121956, 2021.

[8] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems," *Eng Appl Artif Intell*, vol. 114, p. 105082, 2022.

[9] F. W. Wibowo, E. Sediyono, and H. D. Purnomo, "Chimpanzee leader election optimization," *Math Comput Simul*, vol. 201, pp. 68–95, 2022, doi: https://doi.org/10.1016/j.matcom.2022.05.007.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified,

real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[11] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[13] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 6, pp. 1137–1149, 2016.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[17] B. Mahaur, N. Singh, and K. K. Mishra, "Road object detection: a comparative study of deep learning-based algorithms," *Multimed Tools Appl*, vol. 81, no. 10, pp. 14247–14282, 2022.

[18] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[19] Y. Wang *et al.*, "TogetherNet: Bridging Image Restoration and Object Detection Together via Dynamic Enhancement Learning," in *Computer Graphics Forum*, Wiley Online Library, 2022, pp. 465–476.

[20] A. S. Talaat and S. El-Sappagh, "Enhanced aerial vehicle system techniques for detection and tracking in fog, sandstorm, and snow conditions," *J Supercomput*, pp. 1–26, 2023.

[21] J. Li, Y. Feng, Y. Shao, and F. Liu, "IDP-YOLOV9: Improvement of Object

Detection Model in Severe Weather Scenarios from Drone Perspective," *Applied Sciences*, vol. 14, no. 12, p. 5277, 2024.

[22] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[23] R. Wang *et al.*, "Real-time vehicle target detection in inclement weather conditions based on YOLOv4," *Front Neurorobot*, vol. 17, p. 1058723, 2023.

[24] F. Yu *et al.*, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[25] W. Liu, G. Ren, R. Yu, S. Guo, J. Zhu, and L. Zhang, "Image-adaptive YOLO for object detection in adverse weather conditions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, pp. 1792–1800.

[26] H. Zhang, R. Sehab, S. Azouigui, and M. Boukhnifer, "Application and Comparison of Deep Learning Methods to Detect Night-Time Road Surface Conditions for Autonomous Vehicles," *Electronics (Basel)*, vol. 11, no. 5, p. 786, 2022.

[27] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[31] W. Li, "Vehicle detection in foggy weather based on an enhanced YOLO method," in *Journal of Physics: Conference Series*, IOP Publishing, 2022, p. 012015.

[32] A. Farid, F. Hussain, K. Khan, M. Shahzad, U. Khan, and Z. Mahmood, "A Fast and Accurate Real-Time Vehicle Detection Method Using Deep Learning for

Unconstrained Environments," *Applied Sciences*, vol. 13, no. 5, p. 3059, 2023.

[33] C. Li, Q. Zhong, D. Xie, and S. Pu, "Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation," *arXiv preprint arXiv:1804.06055*, 2018.

[34] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.

[35] M. A. Kenk and M. Hassaballah, "Dawn: vehicle detection in adverse weather nature dataset," *arXiv preprint arXiv:2008.05402*, 2020.

[36] I. Ashraf, S. Hur, G. Kim, and Y. Park, "Analyzing performance of YOLOx for detecting vehicles in bad weather conditions," *Sensors*, vol. 24, no. 2, p. 522, 2024.

[37] D. Kumar and N. Muhammad, "Object detection in adverse weather for autonomous driving through data merging and YOLOv8," *Sensors*, vol. 23, no. 20, p. 8471, 2023.

[38] C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10765–10775.

[39] M. Humayun, F. Ashfaq, N. Z. Jhanjhi, and M. K. Alsadun, "Traffic management: Multi-scale vehicle detection in varying weather conditions using yolov4 and spatial pyramid pooling network," *Electronics (Basel)*, vol. 11, no. 17, p. 2748, 2022.

[40] A. Vellaidurai and M. Rathinam, "An Optimal Feature Selection and Mrblstm-based Autonomous Vehicle Detection System in Challenging Weather Conditions," *Preprint*, 2023. https://doi.org/10.21203/rs.3.rs-3080554/v1

[41] S. Wu, Y. Yan, and W. Wang, "CF-YOLOX: an autonomous driving detection model for multi-scale object detection," *Sensors*, vol. 23, no. 8, p. 3794, 2023.

[42] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3354–3361.

[43] F. Yu *et al.*, “Bdd100k: A diverse driving dataset for heterogeneous multitask learning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[44] M. A. Berwo *et al.*, “Deep learning techniques for vehicle detection and classification from images/videos: A survey,” *Sensors*, vol. 23, no. 10, p. 4832, 2023.

[45] J. Liu, X. Liu, Q. Chen, and S. Niu, “A Traffic Parameter Extraction Model Using Small Vehicle Detection and Tracking in Low-Brightness Aerial Images,” *Sustainability*, vol. 15, no. 11, p. 8505, 2023.

[46] N. Aloufi, A. Alnori, V. Thayananthan, and A. Basuhail, “Object detection performance evaluation for autonomous vehicles in sandy weather environments,” *Applied Sciences*, vol. 13, no. 18, p. 10249, 2023.

[47] R. Ghosh, “On-road vehicle detection in varying weather conditions using faster R-CNN with several region proposal networks,” *Multimed Tools Appl*, vol. 80, no. 17, pp. 25985–25999, 2021.

[48] X. Lai *et al.*, “Lisa: Reasoning segmentation via large language model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9579–9589.

[49] M. Hassaballah, M. A. Kenk, K. Muhammad, and S. Minaee, “Vehicle detection and tracking in adverse weather using a deep learning framework,” *IEEE transactions on intelligent transportation systems*, vol. 22, no. 7, pp. 4230–4242, 2020.

[50] Y. Qiu, Y. Lu, Y. Wang, and H. Jiang, “IDOD-YOLOV7: Image-dehazing YOLOV7 for object detection in low-light foggy traffic environments,” *Sensors*, vol. 23, no. 3, p. 1347, 2023.

[51] L. Xu, W. Yan, and J. Ji, “The research of a novel WOG-YOLO algorithm forautonomous driving object detection,” *Scientific Reports*, 2022.

[52] S. Yoon and J. Cho, “Deep multimodal detection in reduced visibility using thermal depth estimation for autonomous driving,” *Sensors*, vol. 22, no. 14, p. 5084, 2022.

[53] A. Karaman *et al.*, “Robust real-time polyp detection system design based on YOLO algorithms by optimizing activation functions and hyper-parameters with

artificial bee colony (ABC)," *Expert Syst Appl*, vol. 221, p. 119741, 2023.

[54] B. Li *et al.*, "Benchmarking single-image dehazing and beyond," *IEEE Transactions on Image Processing*, vol. 28, no. 1, pp. 492–505, 2018.

[55] C. Li *et al.*, "Detection-friendly dehazing: Object detection in real-world hazy scenes," *IEEE Trans Pattern Anal Mach Intell*, 2023.

[56] Anonim, (2023, 10 Aralık). [Online]. Erişim: https://universe.roboflow.com/test-mdnu9/rtts

[57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[58] H. Yang, D. Lin, G. Zhang, H. Zhang, J. Wang, and S. Zhang, "Research on detection of rice pests and diseases based on improved YOLOV5 algorithm," *Applied Sciences*, vol. 13, no. 18, p. 10188, 2023.

[59] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[60] S. Seong, J. Song, D. Yoon, J. Kim, and J. Choi, "Determination of vehicle trajectory through optimization of vehicle bounding boxes using a convolutional neural network," *Sensors*, vol. 19, no. 19, p. 4263, 2019.

[61] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[62] S. Shin, H. Han, and S. H. Lee, "Improved YOLOv3 with duplex FPN for object detection based on deep learning," *The International Journal of Electrical Engineering & Education*, p. 0020720920983524, 2021.

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

[64] Y. Liu, B. Lu, J. Peng, and Z. Zhang, "Research on the use of YOLOv5 object detection algorithm in mask wearing recognition," *World Sci. Res. J*, vol. 6, no. 11, pp. 276–284, 2020.

[65] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[66] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "Panet: Few-shot image semantic segmentation with prototype alignment," in *proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9197–9206.

[67] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, no. 2, p. 217, 2021.

[68] H. Liu, F. Sun, J. Gu, and L. Deng, "Sf-yolov5: A lightweight small object detection algorithm based on improved feature fusion mode," *Sensors*, vol. 22, no. 15, p. 5817, 2022.

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

[70] M. J. A. Soeb *et al.*, "Tea leaf disease detection and identification based on YOLOv7 (YOLO-T)," *Sci Rep*, vol. 13, no. 1, Dec. 2023, doi: 10.1038/s41598-023-33270-4.

[71] K. Li, Y. Wang, and Z. Hu, "Improved YOLOv7 for small object detection algorithm based on attention and dynamic convolution," *Applied Sciences*, vol. 13, no. 16, p. 9316, 2023.

[72] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis," *arXiv preprint arXiv:2211.04800*, 2022.

[73] M. Hussain, "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, p. 677, 2023.

[74] E. Dada, S. Joseph, D. Oyewola, A. A. Fadele, and H. Chiroma, "Application of grey wolf optimization algorithm: recent trends, issues, and possible horizons," *Gazi University Journal of Science*, vol. 35, no. 2, pp. 485–504, 2022.

[75] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[76] Y. Ding and X. Luo, “SDNIA-YOLO: A Robust Object Detection Model for Extreme Weather Conditions,” *arXiv preprint arXiv:2406.12395*, 2024.

[77] B. Akay, D. Karaboga, and R. Akay, “A comprehensive survey on optimizing deep learning models by metaheuristics,” *Artif Intell Rev*, vol. 55, no. 2, pp. 829–894, 2022.

[78] R. Walambe, A. Marathe, K. Kotecha, and G. Ghinea, “Lightweight object detection ensemble framework for autonomous vehicles in challenging weather conditions,” *Comput Intell Neurosci*, vol. 2021, 2021.

[79] H. Wang *et al.*, “YOLOv5-Fog: A multiobjective visual detection algorithm for fog driving scenes based on improved YOLOv5,” *IEEE Trans Instrum Meas*, vol. 71, pp. 1–12, 2022.

[80] L. Xu, W. Yan, and J. Ji, “The research of a novel WOG-YOLO algorithm for autonomous driving object detection,” *Sci Rep*, vol. 13, no. 1, p. 3699, 2023.

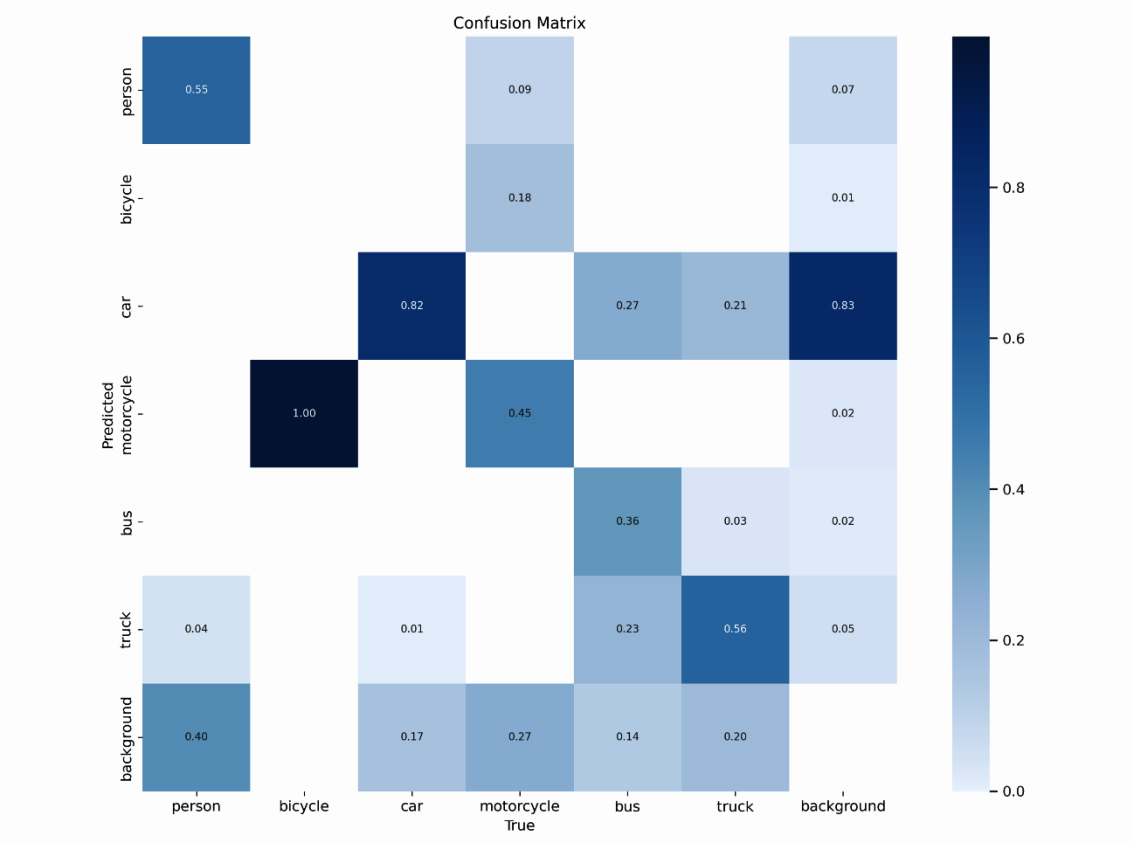
[81] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3354–3361.

8. EKLER

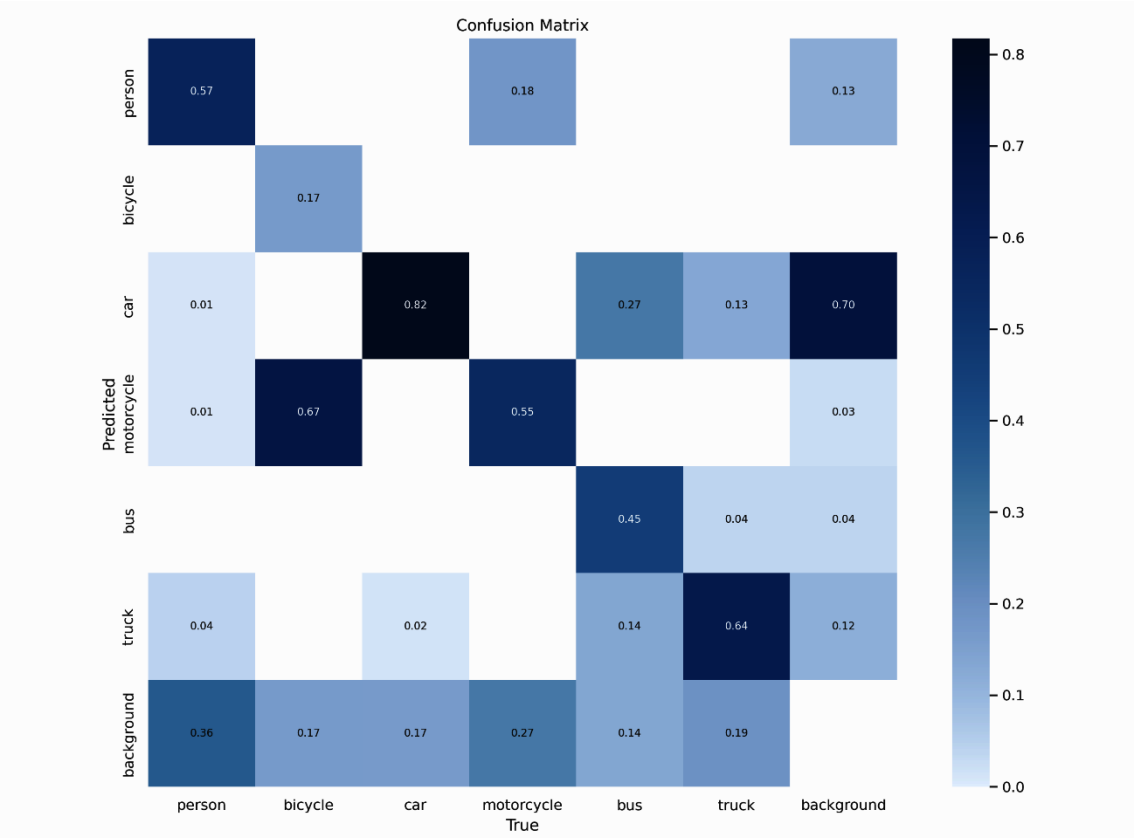
8.1. EK 1: KARIŞIKLIK MATRİSİ GÖRÜNTÜLERİ

DAWN ve RTTS veri seti kullanılarak, YOLOv5, YOLOv7 ve YOLOv9 modellerinin GKO, YTO ve ŞLSO algoritmalarıyla optimize edilmiş hallerinin her birinin Karışıklık Matrisi sonuçları aşağıda yer almaktadır.

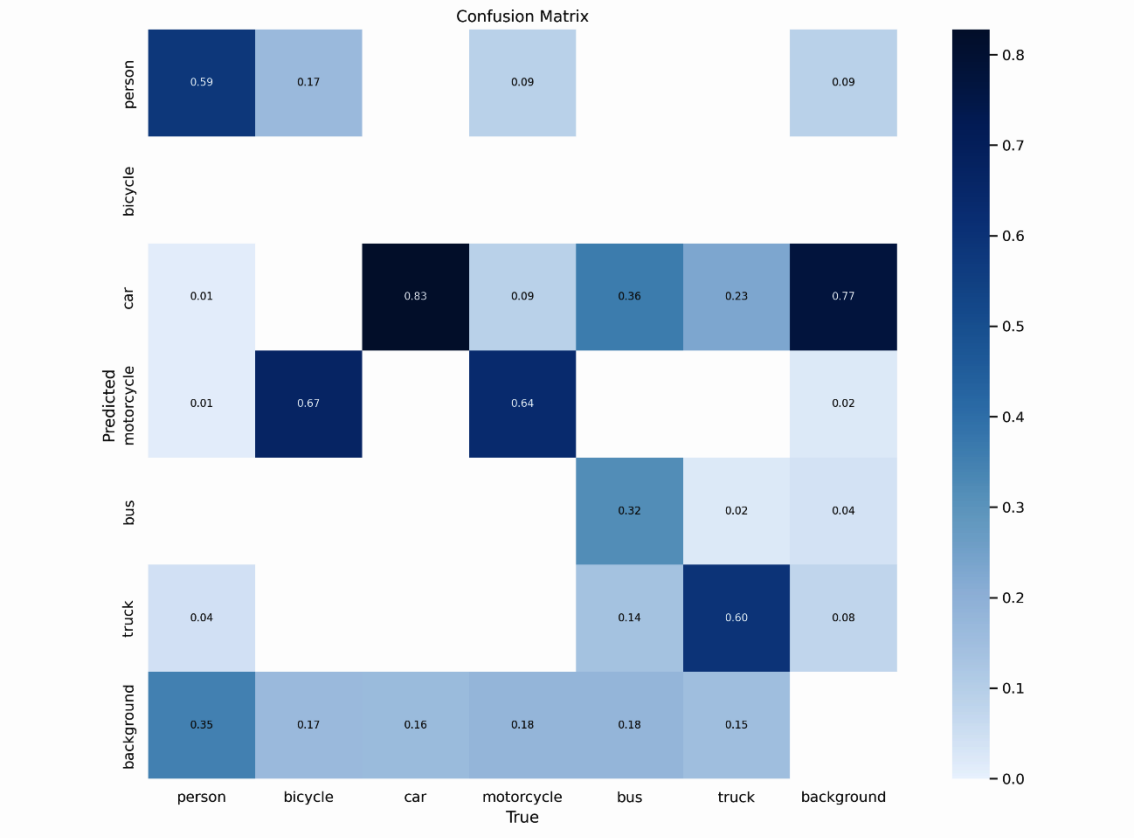
DAWN veri seti kullanılarak yapılan eğitim sonucu oluşan Karışıklık Matrisleri:



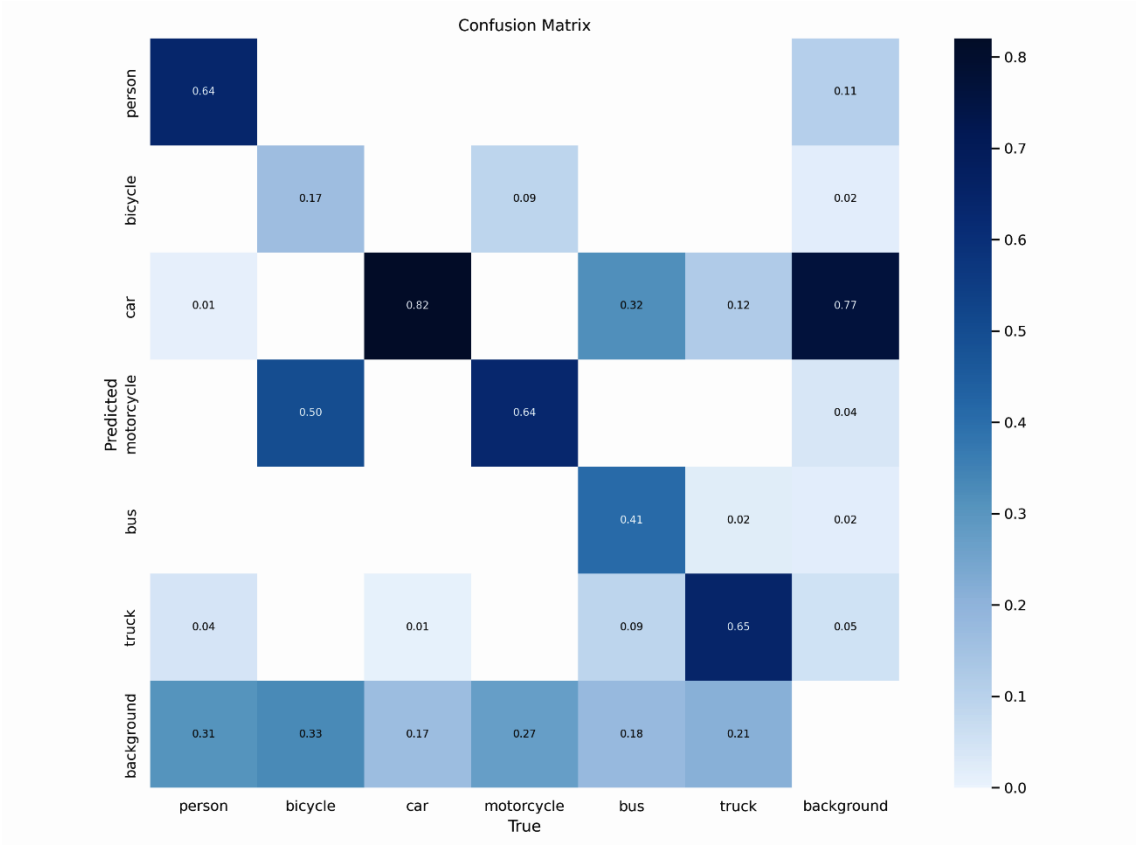
Şekil 8.1. YOLOv5 orijinal.



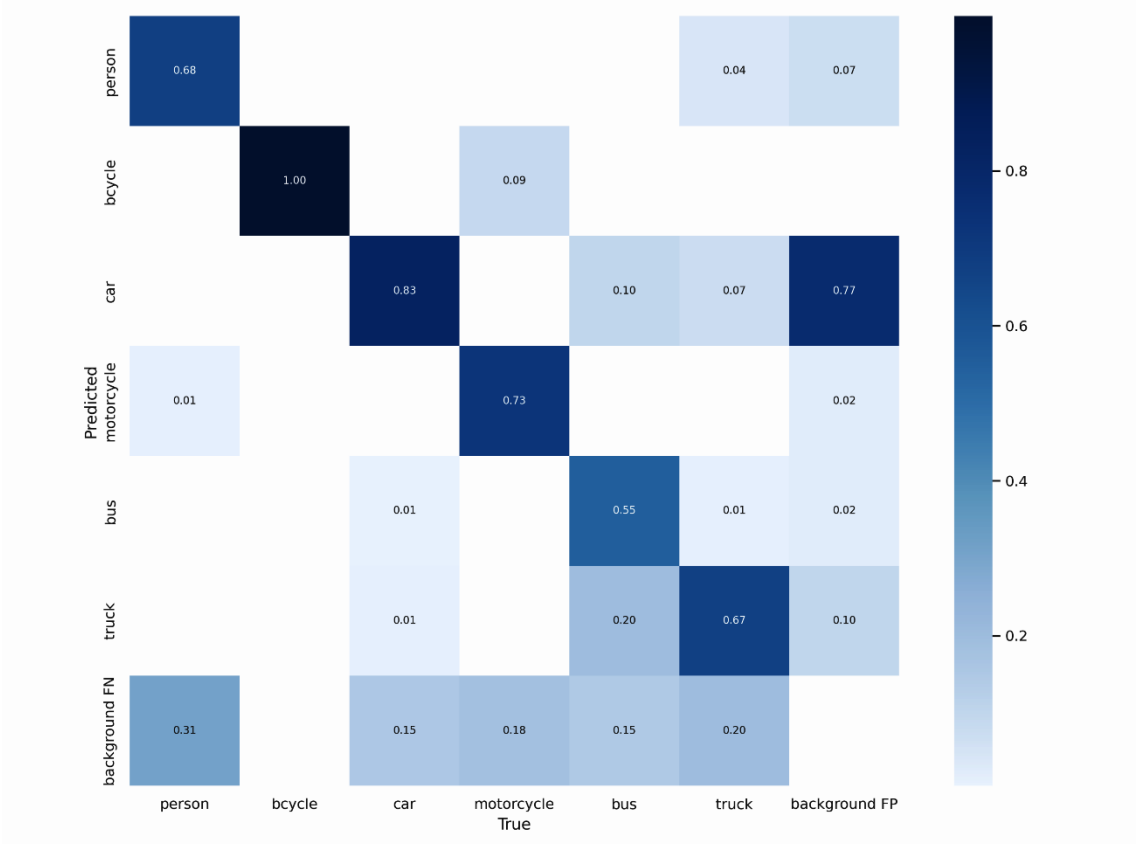
Şekil 8.2. YOLOv5 + GKO.



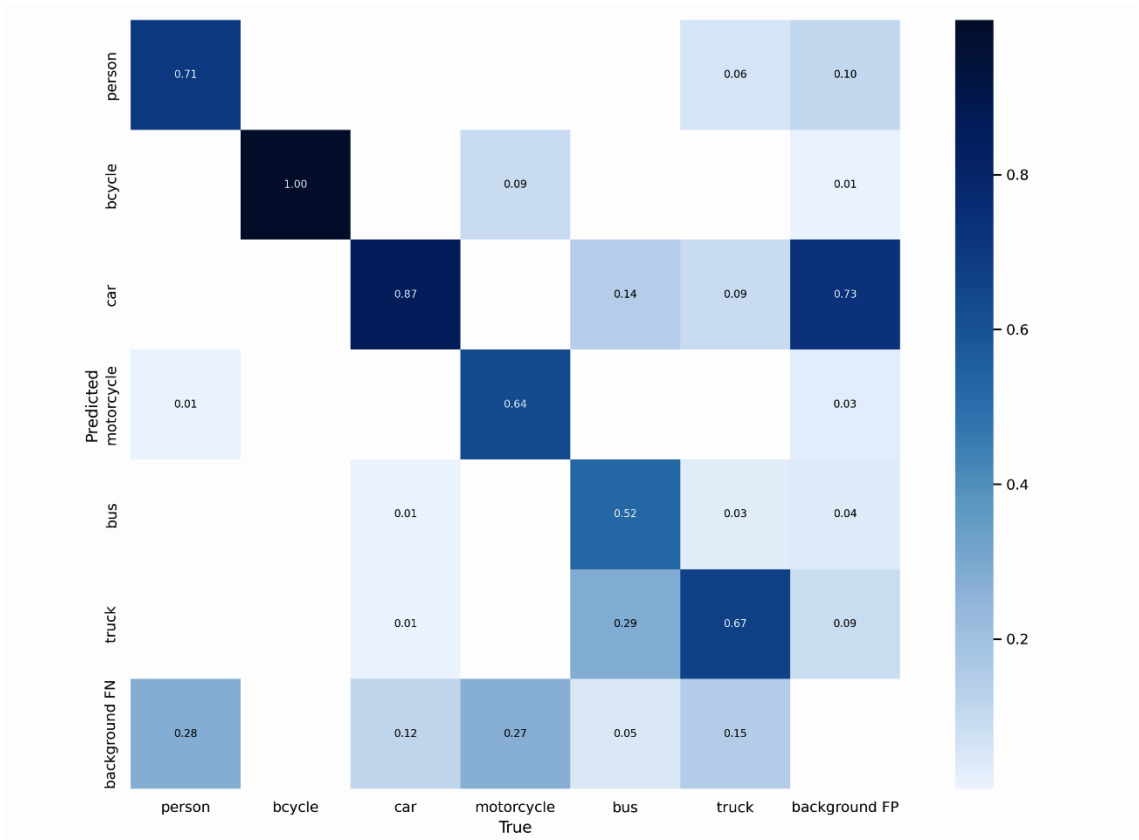
Şekil 8.3. YOLOv5 + YTO.



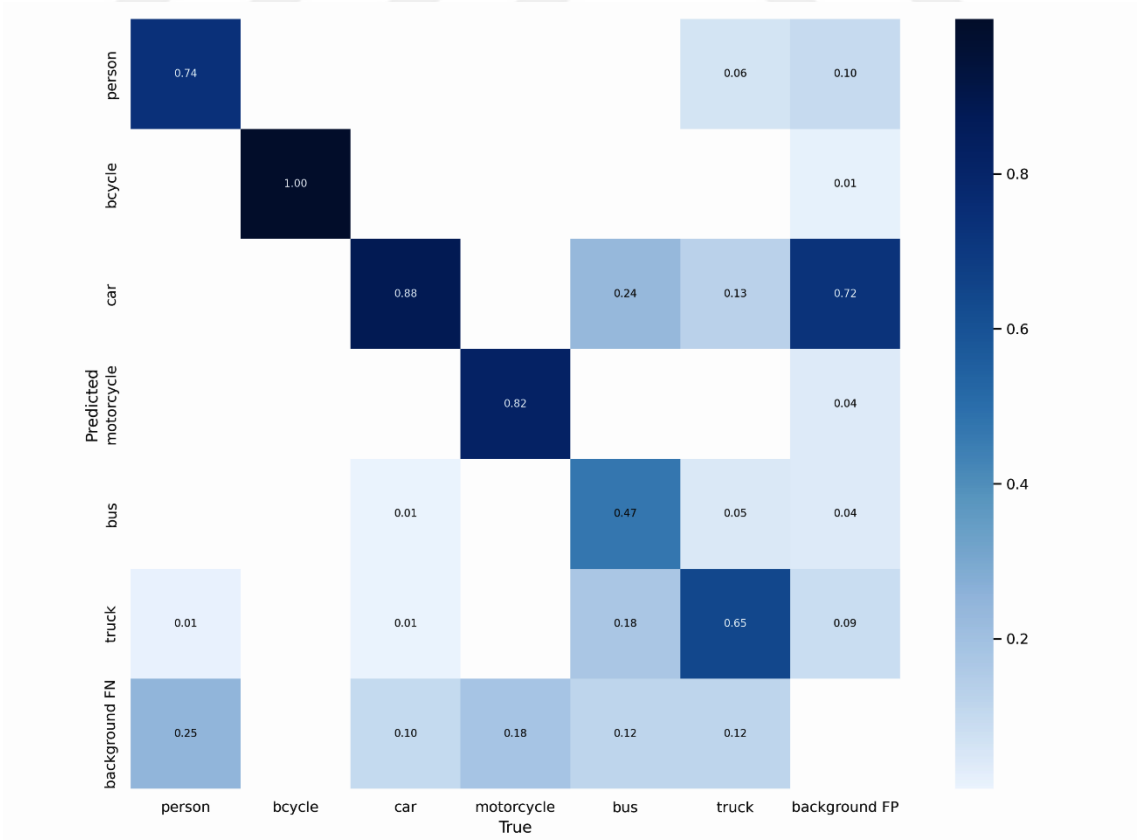
Şekil 8.4. YOLOv5 + ŞLSO.



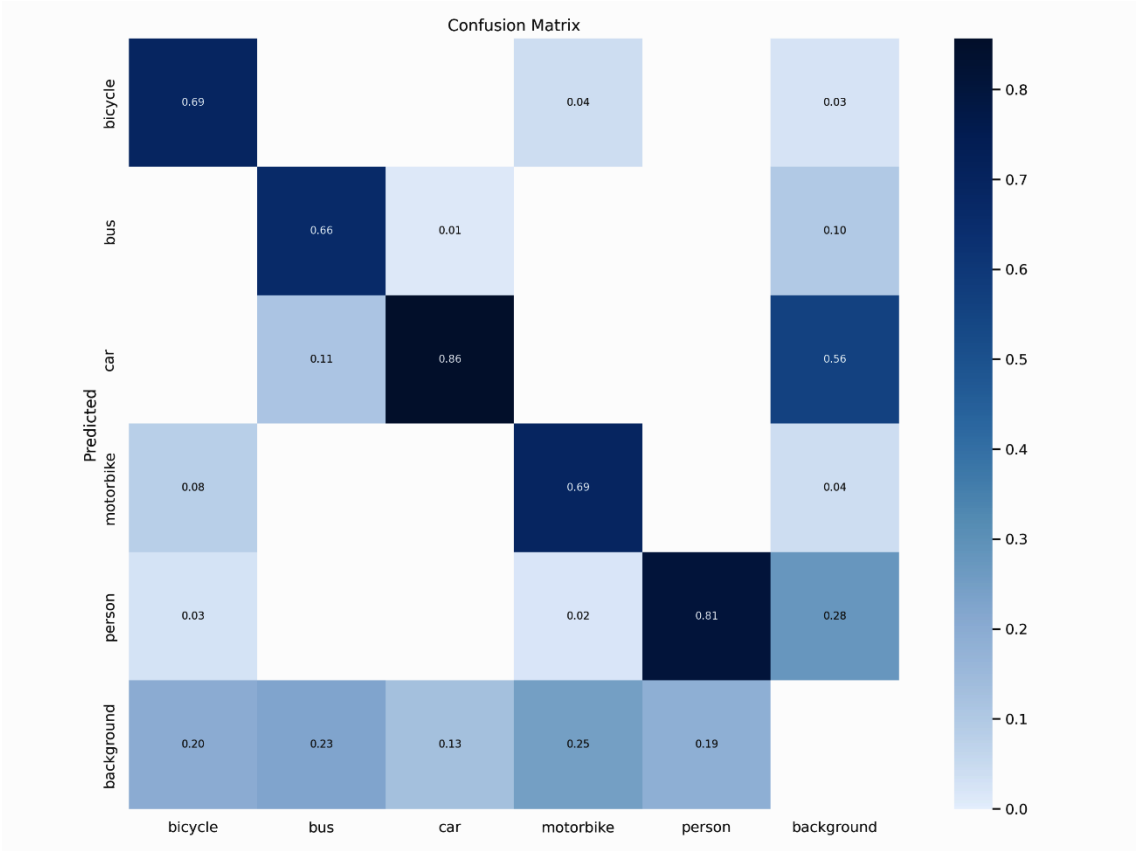
Şekil 8.5. YOLOv7 orijinal.



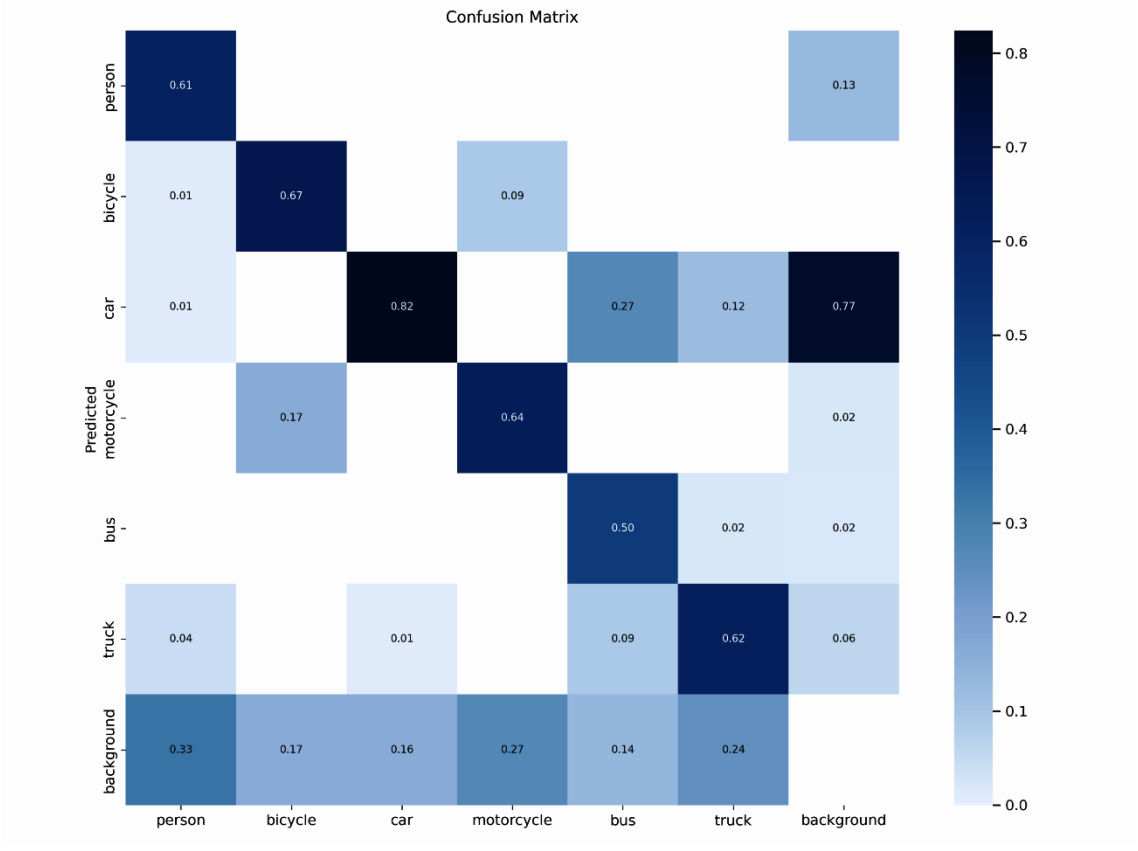
Şekil 8.6. YOLOv7 + GKO.



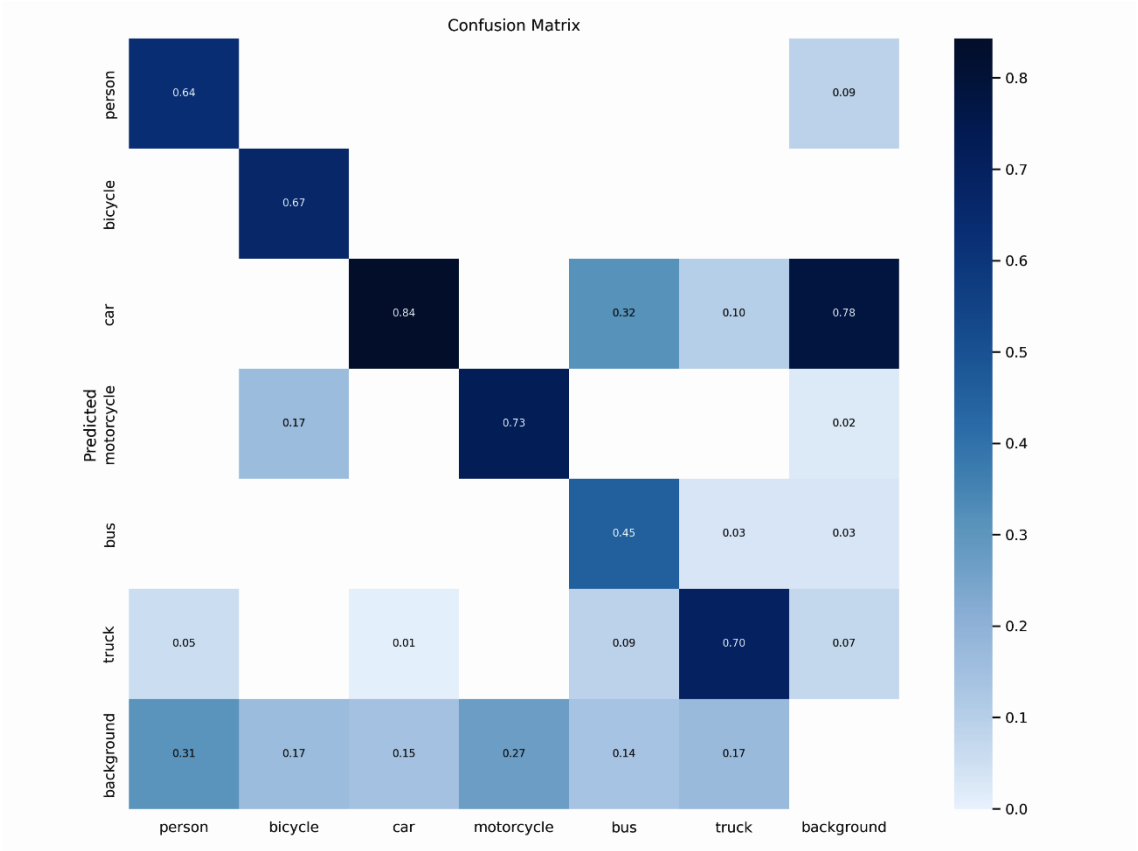
Şekil 8.7. YOLOv7 + YTO.



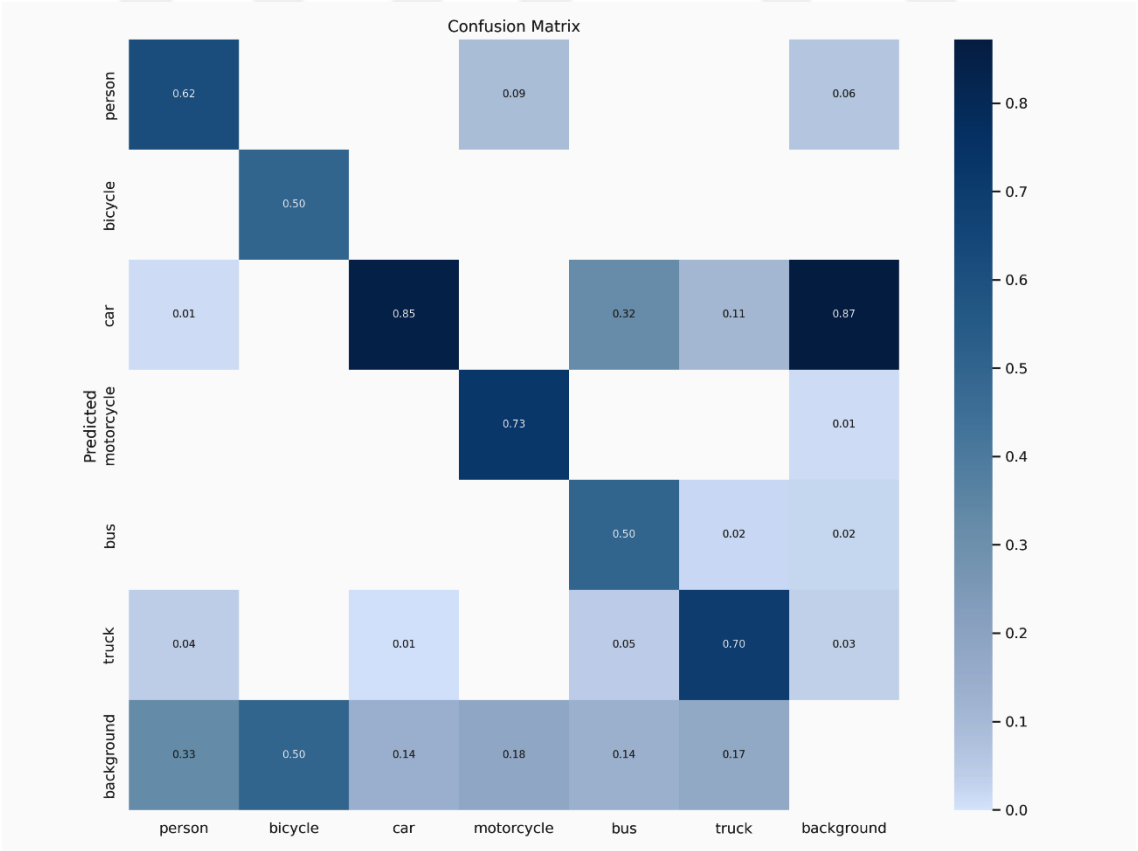
Şekil 8.8. YOLOv7 + ŞLSO.



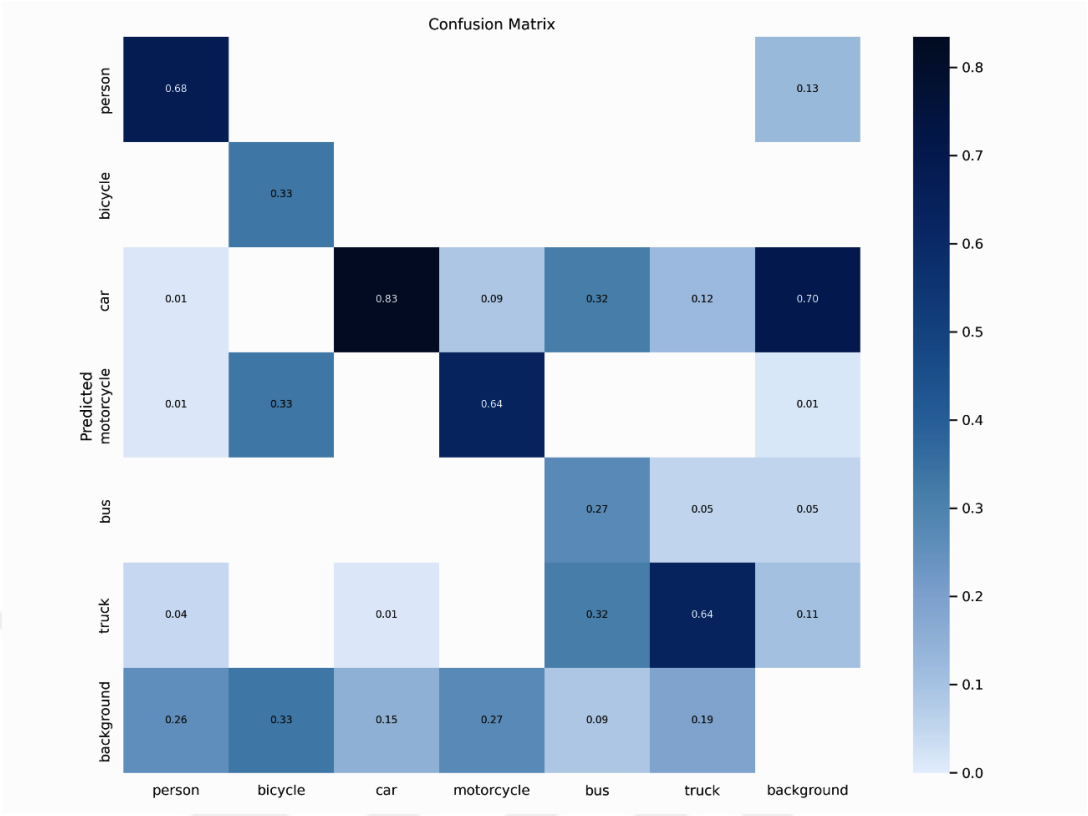
Şekil 8.9. YOLOv9 orijinal.



Şekil 8.10. YOLOv9 + GKO.

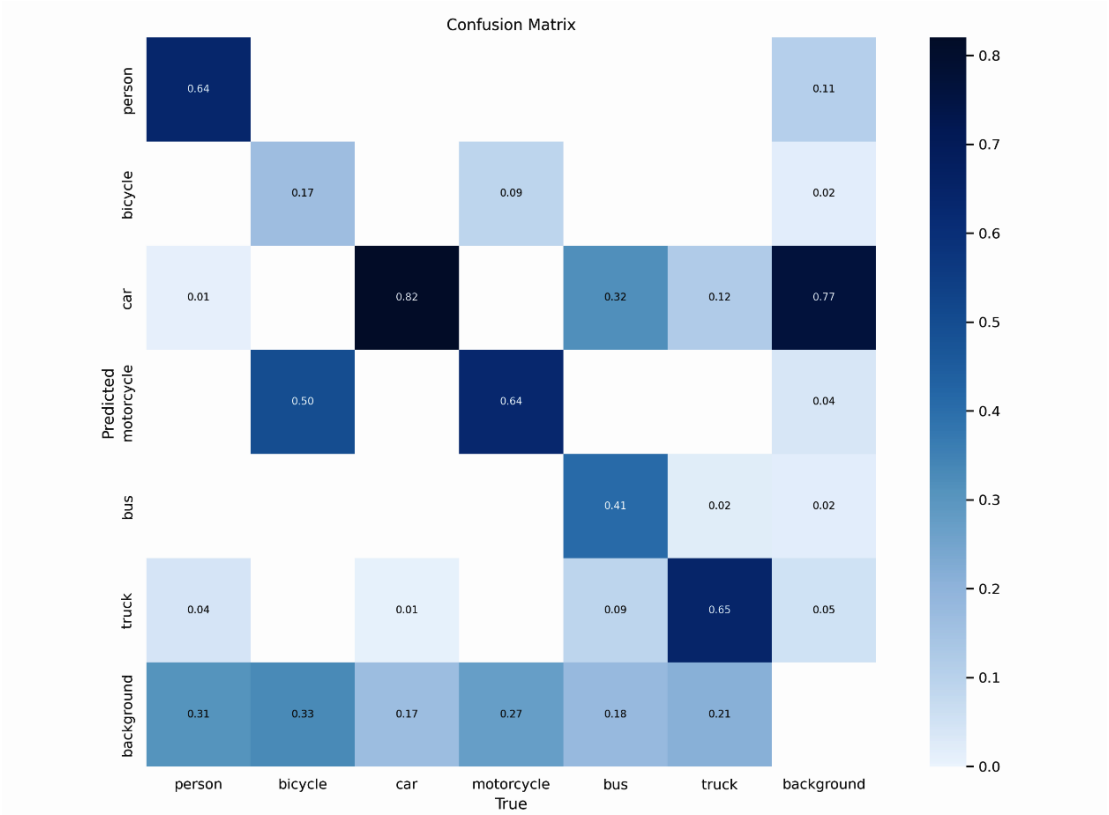


Şekil 8.11. YOLOv9 + YTO.

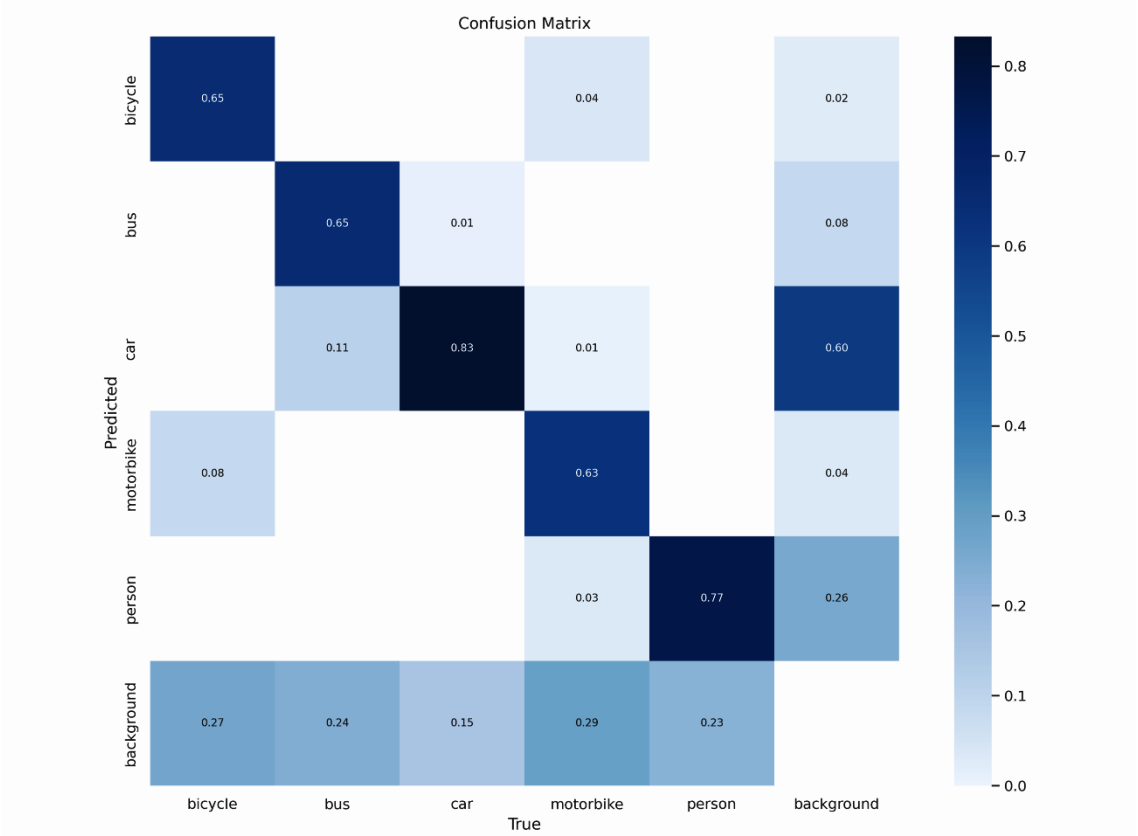


Şekil 8.12. YOLOv9 + ŞLSO.

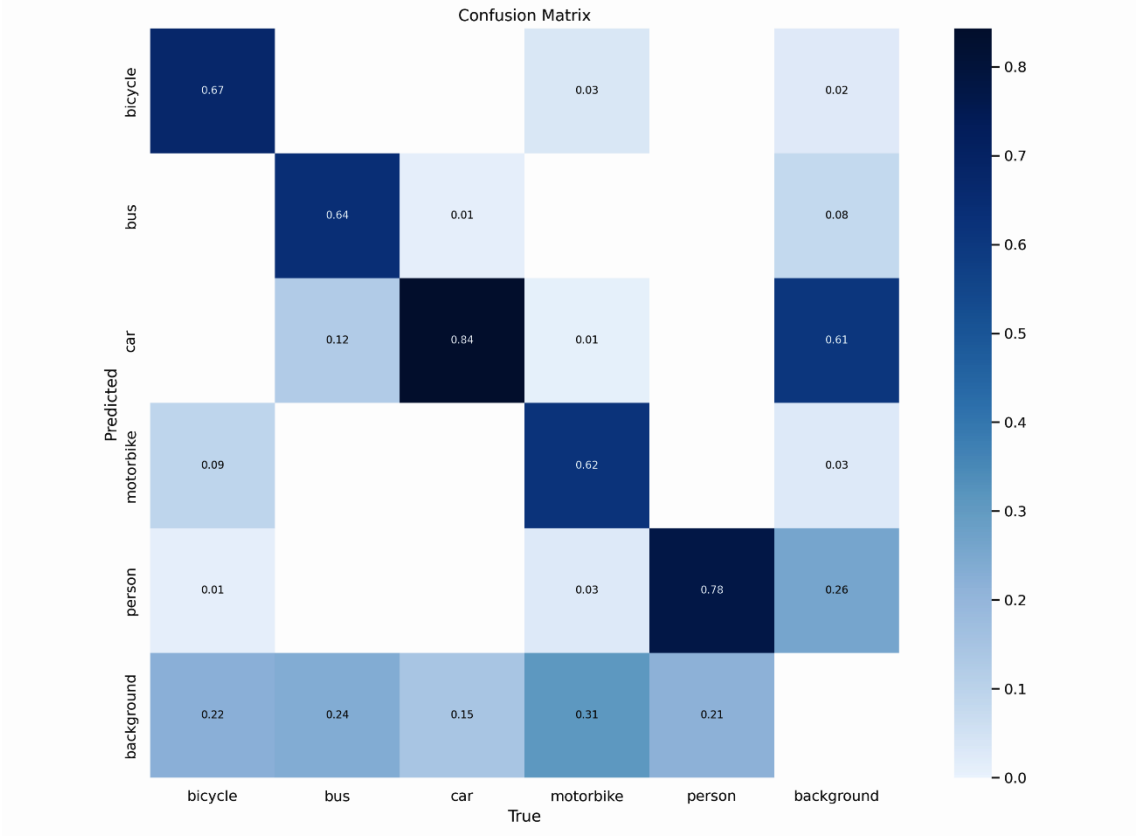
RTTS veri seti kullanılarak yapılan eğitim sonucu oluşan Karışıklık Matrisleri:



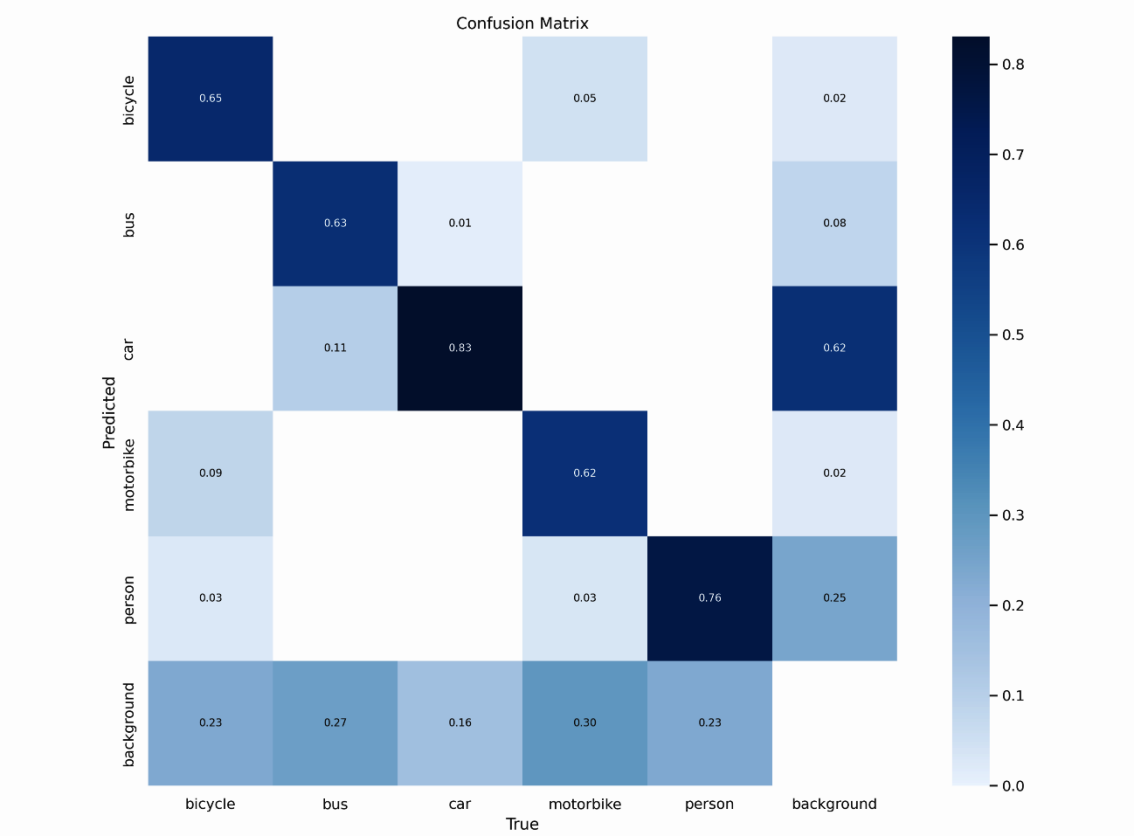
Şekil 8.13. YOLOv5 orijinal.



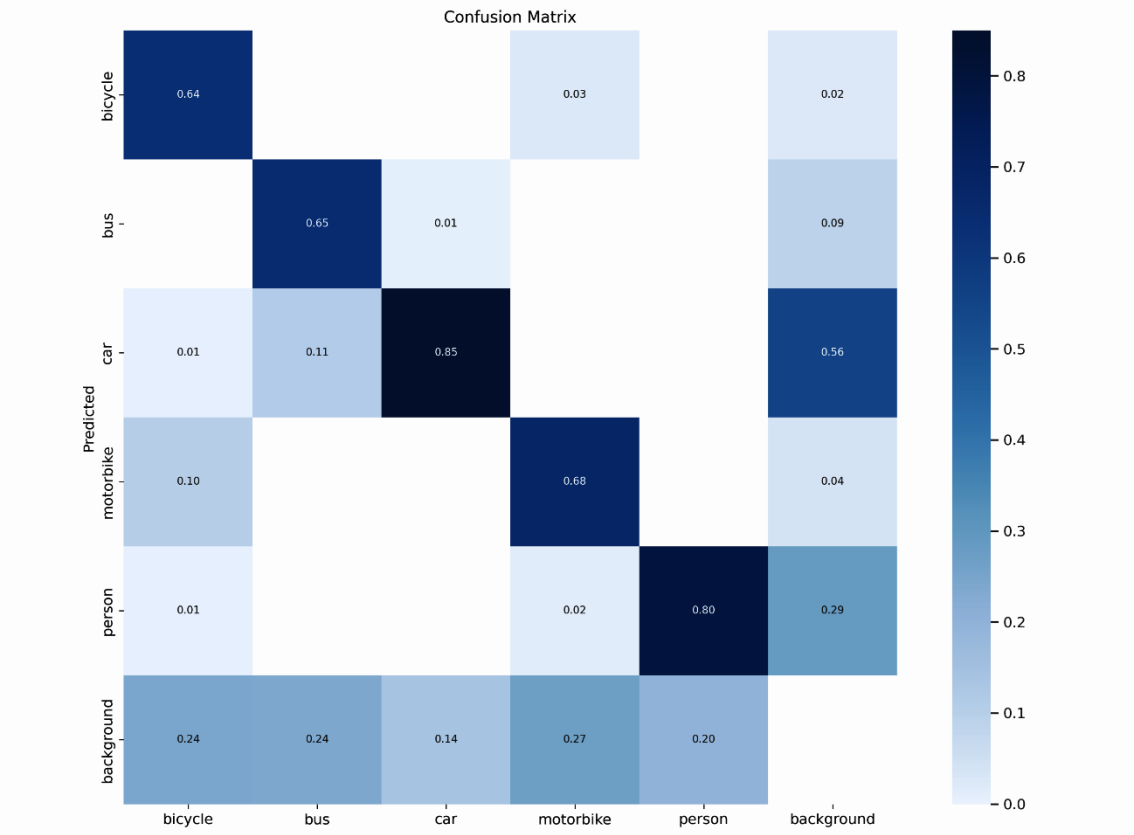
Şekil 8.14. YOLOv5 + GKO.



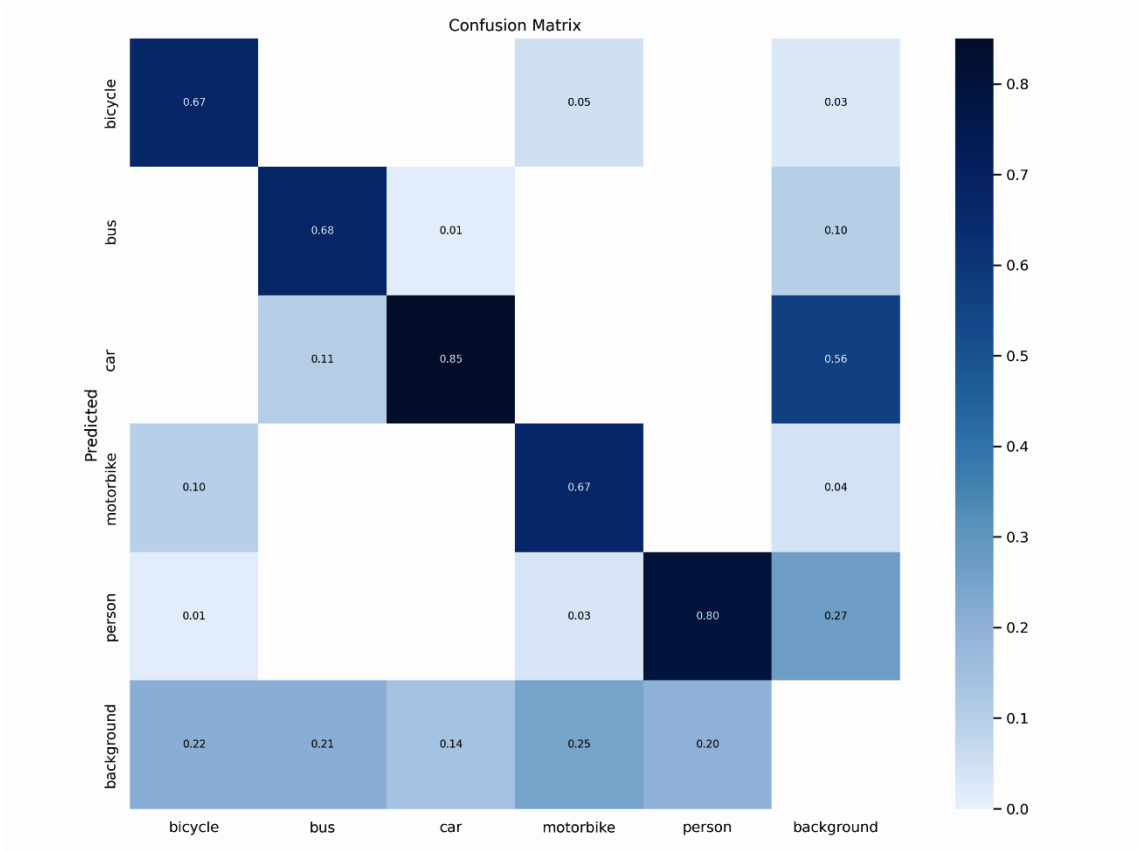
Şekil 8.15. YOLOv5 + YTO.



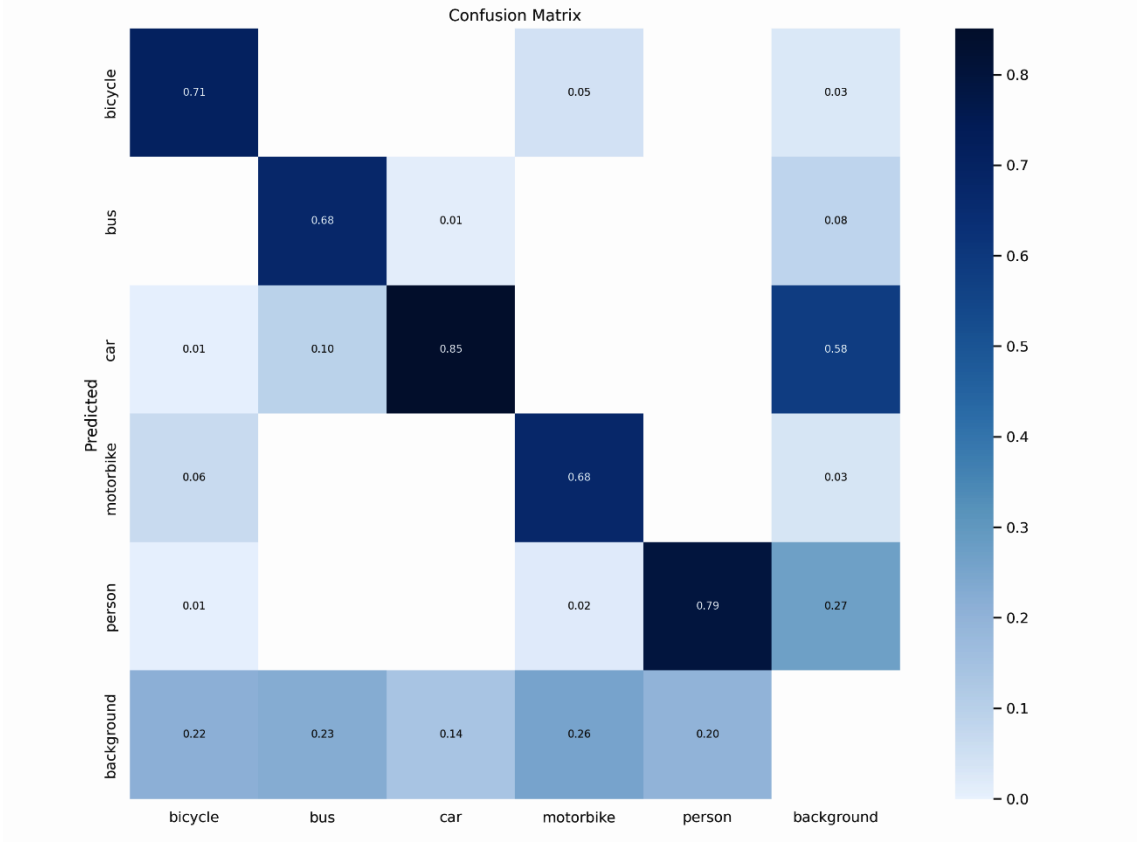
Şekil 8.16. YOLOv5 + ŞLSO.



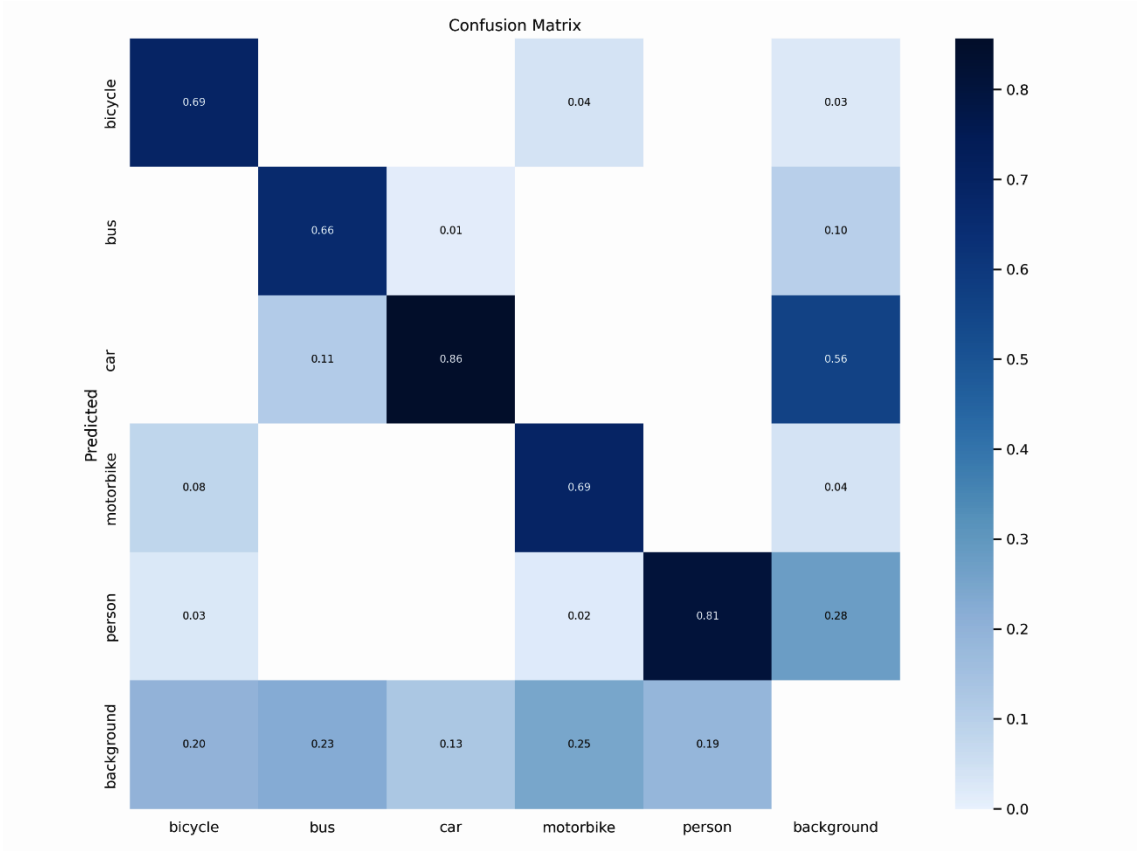
Şekil 8.17. YOLOv7 orijinal.



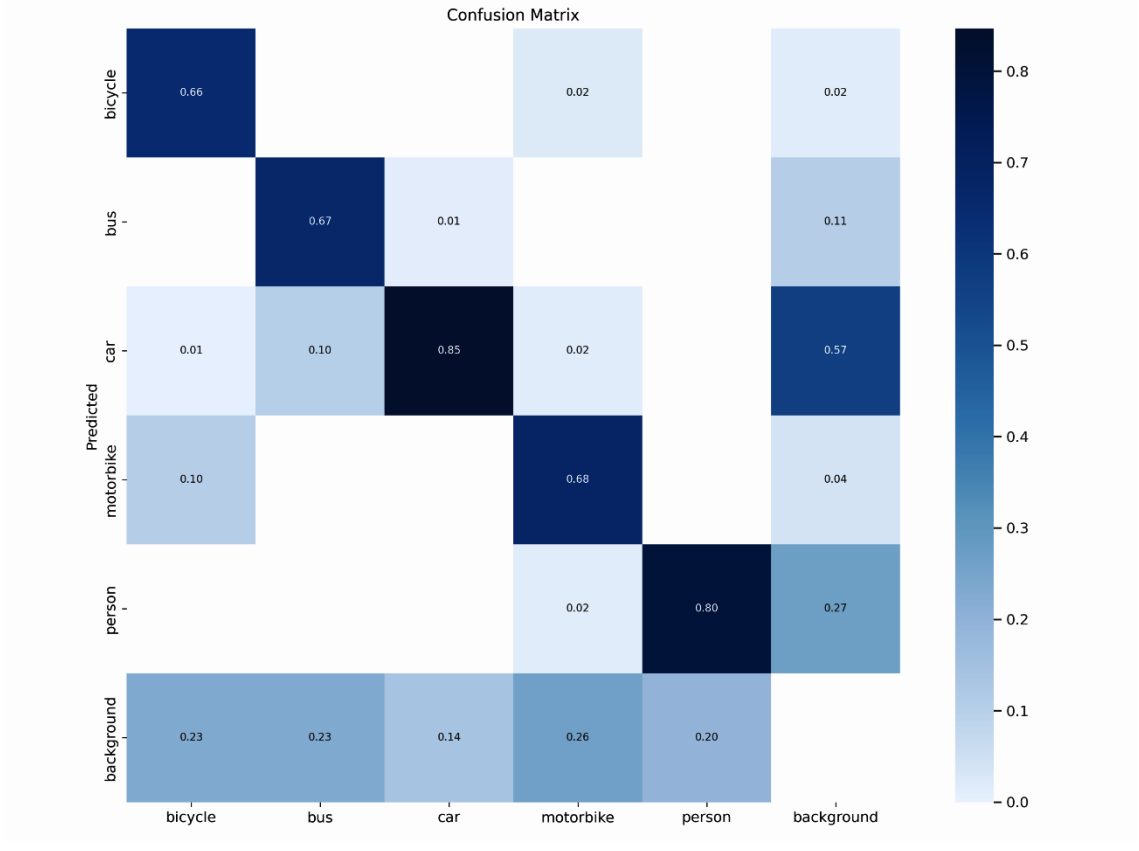
Şekil 8.18. YOLOv7 + GKO.



Şekil 8.19. YOLOv7 + YTO.

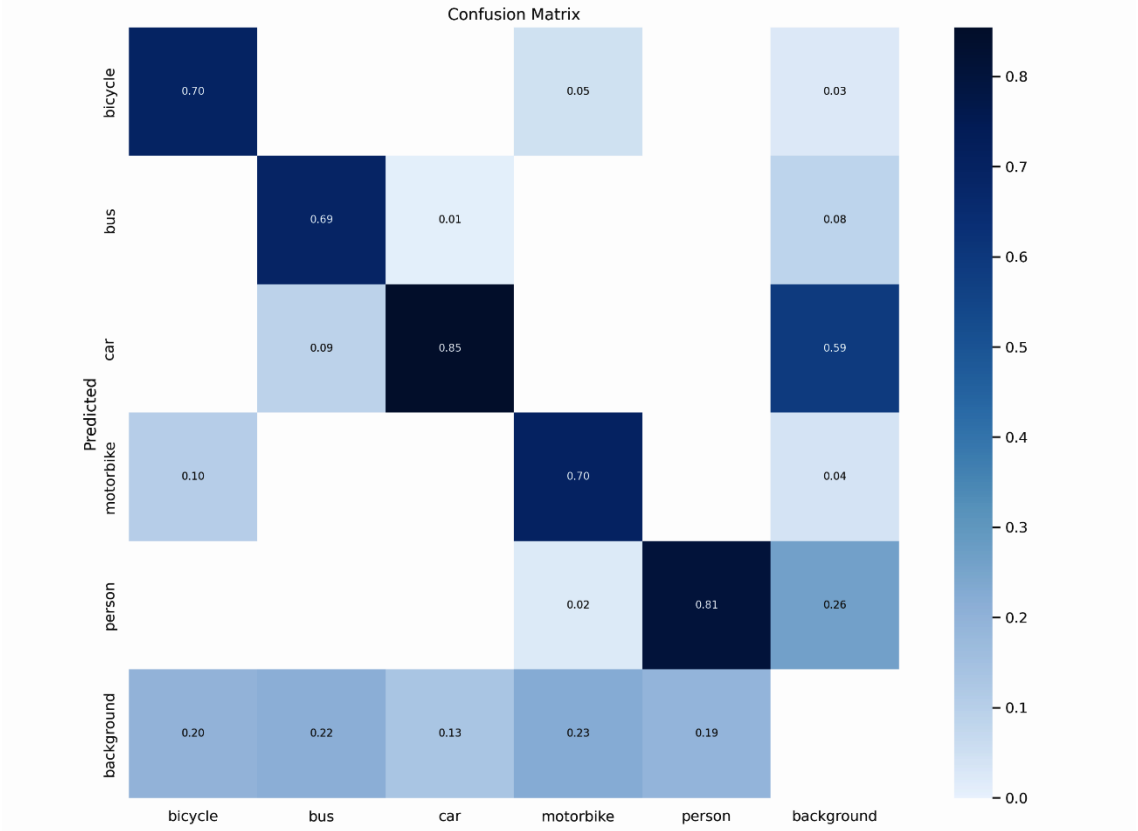


Şekil 8.20. YOLOv7 + ŞLSO.



Şekil 8.21. YOLOv9 orijinal.

Şekil 8.22. YOLOv9 + GKO.



Şekil 8.23. YOLOv9 + YTO.

Şekil 8.24. YOLOv9 + ŞLSO.

8.2. EK 2: EĞİTİM SONUNDA NESNE TAHMİNİ GÖRÜNTÜLERİ

Eğitimler tamamlandıktan sonra sisteme bazı görüntüler verilip görüntüler içindeki nesnelerin tahmin edilmesi beklenmektedir. Her bir algoritmalardan elde edilen görüntülerin bazıları aşağıda yer almaktadır.

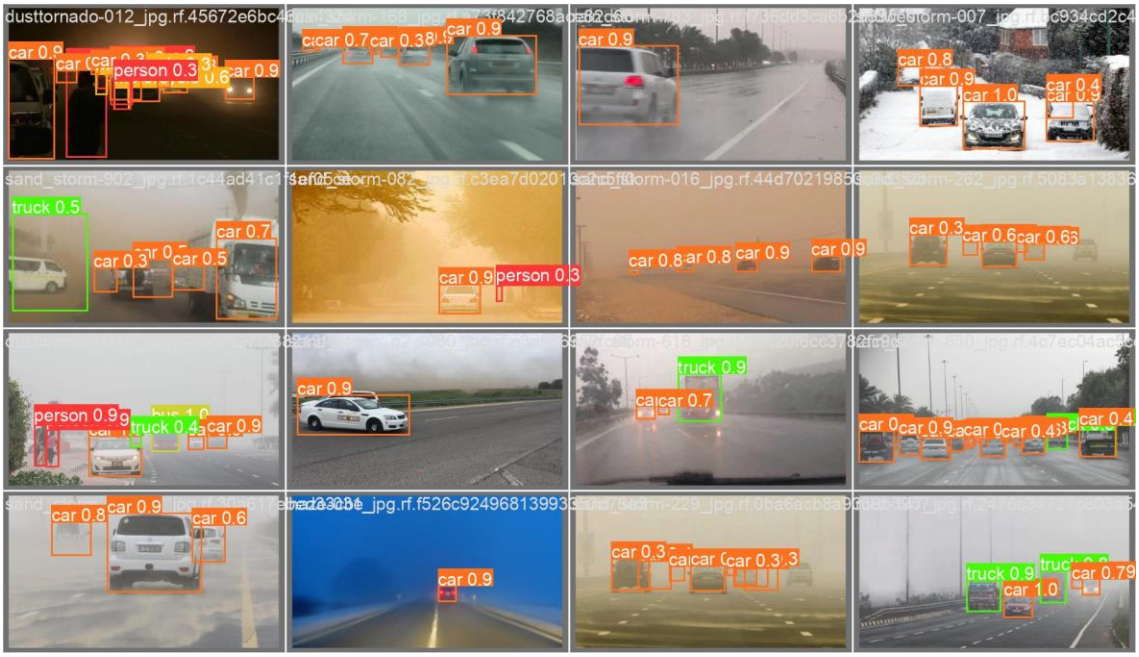
DAWN veri seti üzerinde elde edilen görüntüler:

Şekil 8.25. YOLOv5 + GKO.

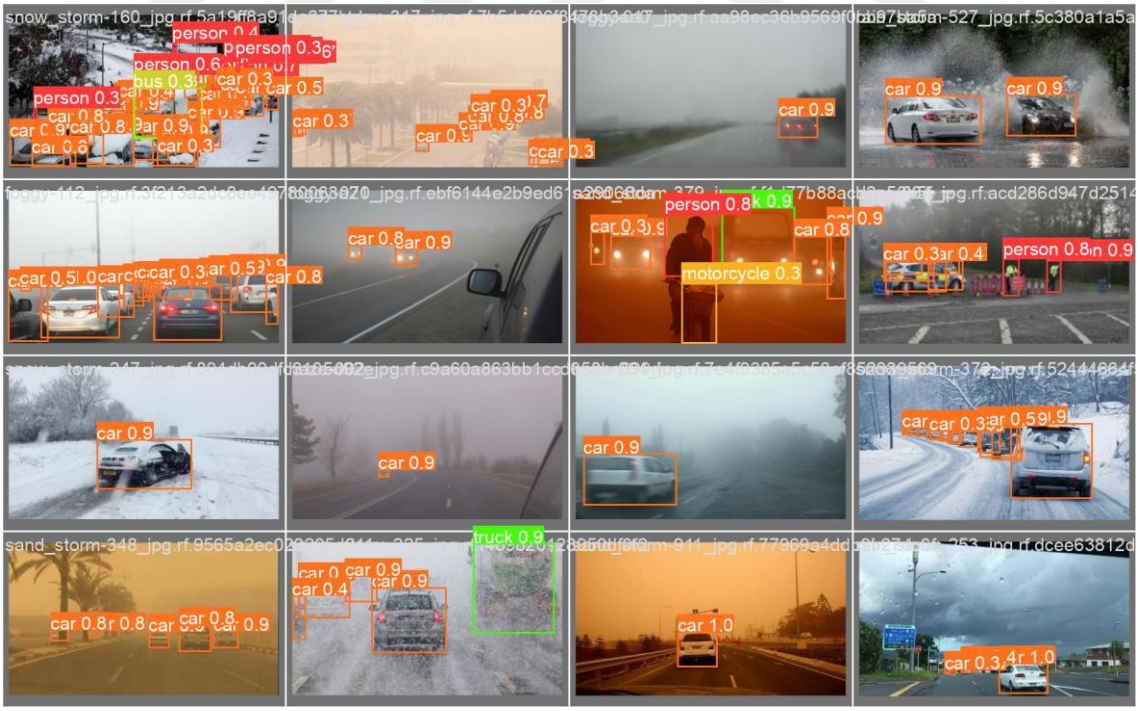
Şekil 8.26. YOLOv5 + GKO.

Şekil 8.27. YOLOv5 + GKO.

Şekil 8.28. YOLOv5 + YTO.



Şekil 8.29. YOLOv5 + YTO.



Şekil 8.30. YOLOv5 + YTO.

Şekil 8.31. YOLOv5 + ŞLSO.

Şekil 8.32. YOLOv5 + ŞLSO.

Şekil 8.33. YOLOv5 + ŞLSO.

Şekil 8.34. YOLOv7 + GKO.

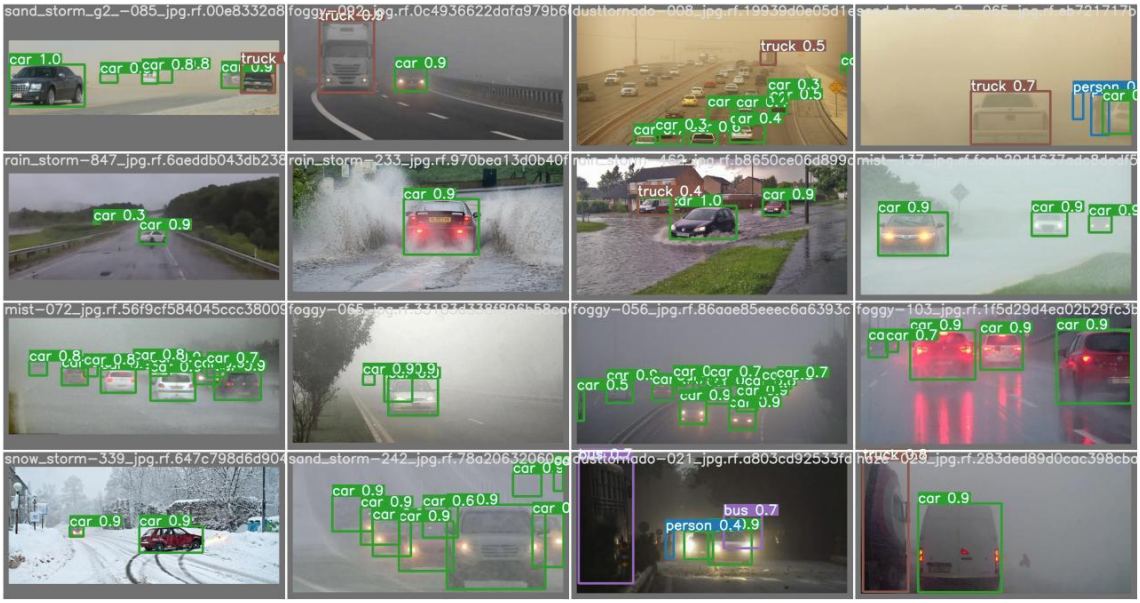
Şekil 8.35. YOLOv7 + GKO.

Şekil 8.36. YOLOv7 + GKO.

Şekil 8.37. YOLOv7 + YTO.

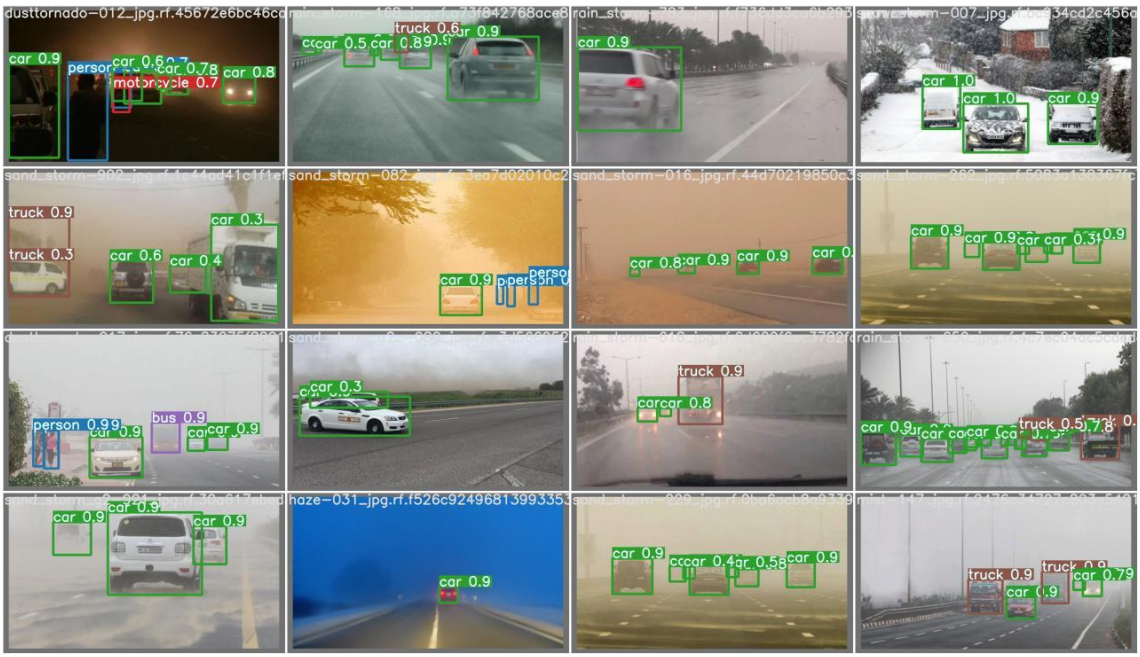
Şekil 8.38. YOLOv7 + YTO.

Şekil 8.39. YOLOv7 + YTO.



Şekil 8.40. YOLOv7 + ŞLSO.

Şekil 8.41. YOLOv7 + ŞLSO.



Şekil 8.42. YOLOv7 + ŞLSO.

Şekil 8.43. YOLOv9 + GKO.

Şekil 8.44. YOLOv9 + GKO.

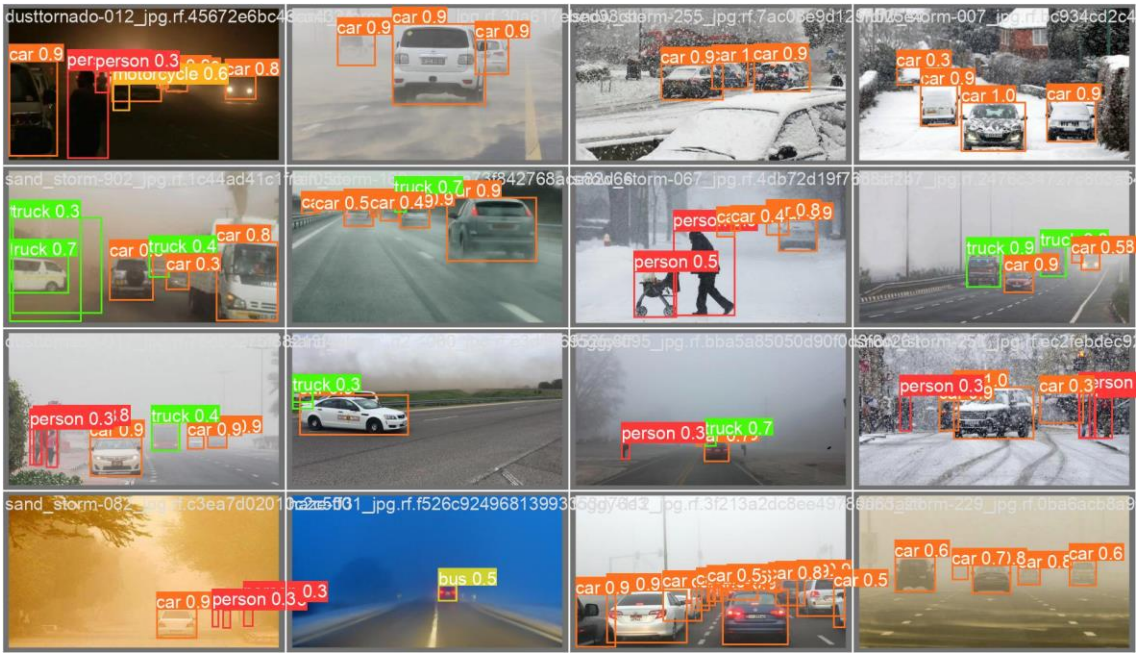
Şekil 8.45. YOLOv9 + GKO.

Şekil 8.46. YOLOv9 + YTO.

Şekil 8.47. YOLOv9 + YTO.

Şekil 8.48. YOLOv9 + YTO.

Şekil 8.49. YOLOv9 + ŞLSO.



Şekil 8.50. YOLOv9 + ŞLSO.

Şekil 8.51. YOLOv9 + ŞLSO.

RTTS veri seti üzerinde elde edilen görüntüler:

Şekil 8.52. YOLOv5 + GKO.

Şekil 8.53. YOLOv5 + GKO.

Şekil 8.54. YOLOv5 + GKO.

Şekil 8.55. YOLOv5 + YTO.

Şekil 8.56. YOLOv5 + YTO.

Şekil 8.57. YOLOv5 + YTO.

Şekil 8.58. YOLOv5 + ŞLSO.

.Şekil 8.59. YOLOv5 + ŞLSO.

Şekil 8.60. YOLOv5 + ŞLSO.

Şekil 8.61. YOLOv7 + GKO.

Şekil 8.62. YOLOv7 + GKO.

Şekil 8.63. YOLOv7 + GKO.

Şekil 8.64. YOLOv7 + YTO.

Şekil 8.65. YOLOv7 + YTO.

Şekil 8.66. YOLOv7 + YTO.

Şekil 8.67. YOLOv7 + ŞLSO.

Şekil 8.68. YOLOv7 + ŞLSO.

Şekil 8.69. YOLOv7 + ŞLSO.

Şekil 8.70. YOLOv9 + GKO.

Şekil 8.71. YOLOv9 + GKO.

Şekil 8.72. YOLOv9 + GKO.

Şekil 8.73. YOLOv9 + YTO.

Şekil 8.74. YOLOv9 + YTO.

Şekil 8.75. YOLOv9 + YTO.

Şekil 8.76. YOLOv9 + ŞLSO.

Şekil 8.77. YOLOv9 + ŞLSO.

Şekil 8.78. YOLOv9 + ŞLSO.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : İbrahim ÖZCAN

Yabancı Dili : İngilizce

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Doktora	Elektrik-Elektronik ve Bilgisayar Müh.	Düzce Üniversitesi	2024
Y. Lisans	Elektrik Elektronik Müh.	Balıkesir Üniversitesi	2010
Lisans	Bilgisayar Mühendisliği	Kocaeli Üniversitesi	2007
Lise	Fen	Tavşanlı Atatürk Lisesi	2002

TEZDEN ÇIKAN YAYIN

İ. Özcan, Y. Altun, and C. Parlak, "Improving YOLO Detection Performance of Autonomous Vehicles in Adverse Weather Conditions Using Metaheuristic Algorithms," *Applied Sciences*, vol. 14, no. 13, p. 5841, Jul. 2024, doi: 10.3390/app14135841.