T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YAPAY ARI KOLONİSİ VE PARÇACIK SÜRÜ OPTİMİZASYONU İLE YOLO ALGORİTMASININ HİPERPARAMETRELERİNİN BELİRLENMESİ

Yahya GÜNER

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Şubat-2024
KONYA

TEZ KABUL VE ONAYI

Yahya GÜNER tarafından hazırlanan “Yapay Arı Kolonisi ve Parçacık Sürü Optimizasyonu ile YOLO Algoritmasının Hiperparametrelerinin Belirlenmesi” adlı tez çalışması 01/02/2024 tarihinde aşağıdaki jüri tarafından oy birliği / oy çokluğu ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı’nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. İsmail BABAOĞLU Konya Teknik Üniversitesi	……..
Danışman Prof. Dr. Mustafa Servet KIRAN Konya Teknik Üniversitesi	……..
Üye Doç. Dr. Hüseyin HAKLI Necmettin Erbakan Üniversitesi	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Mevlüt UYAN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Yahya Güner

Tarih:

ÖZET

YÜKSEK LİSANS TEZİ

YAPAY ARI KOLONİSİ VE PARÇACIK SÜRÜ OPTİMİZASYONU İLE YOLO ALGORİTMASININ HİPERPARAMETRELERİNİN BELİRLENMESİ

Yahya GÜNER

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Mustafa Servet KIRAN

2024, 72 Sayfa

Jüri
Prof. Dr. Mustafa Servet KIRAN
Prof. Dr. İsmail BABAOĞLU
Doç. Dr. Hüseyin HAKLI

Bu tez çalışmasında YOLO algoritmasının beşinci sürümü olan YOLOv5 algoritmasında bulunan hiperparametrelerin optimize edilmesi sürecinde Yapay Arı Kolonisi (ABC) ve Parçacık Sürü Optimizasyonu (PSO) algoritmalarının kullanımı ve etkinliği incelenmiştir. YOLO algoritması nesne tespiti alanında önemli ilerlemeler sağlamış olmasına karşın beraberinde getirdiği hiperparametrelerin uzman bilgisi ve deneme yanılmalar gerektiren ayarlamalara ihtiyaç duyması algoritmanın uygulanmasını karmaşık bir süreç haline getirmektedir. Bu çalışmada algoritmanın hiperparametrelerinden dokuz tanesi seçilerek ABC ve PSO algoritmaları ile optimize edilmiştir. Oxford-IIIT evcil hayvan veri kümesindeki görseller kullanılarak elde edilen modellerin eğitim performansları kıyaslandığında PSO ile optimize edilen modelin zor ve nadir bulunan verileri tespit etme eğiliminde olduğu ve hızlı yakınsama ile eğitildiği görülmüştür. Buna karşılık ABC ile optimize edilen modelin tüm verileri dengeli bir şekilde öğrenerek düşük öğrenme oranlarıyla yavaş ve istikrarlı bir yakınsama yaptığı görülmüştür. Test aşamasında ABC algoritması ile optimize edilen modelin %81,7 mAP skoru aldığı, bu sonucun eğitim sonuçlarına benzer olduğu görülmüşken PSO ile optimize edilen modelin %76,8 mAP skoru aldığı ve eğitim esnasında aşırı uymaya girdiği, test verilerinde başarılı bir performans vermediği görülmüştür. Bu sonuçlar YOLO algoritmasının hiperparametre optimizasyonu probleminde PSO'ya göre daha çok sonuç çeşitliliği üretebilen ABC algoritmasının başarılı sonuçlar elde etmek için daha elverişli olduğunu göstermiştir.

Anahtar Kelimeler: ABC algoritması, hiperparametre optimizasyonu, nesne tespiti, PSO algoritması, YOLO algoritması

ABSTRACT

MS THESIS

DETERMINING THE HYPERPARAMETERS OF YOLO ALGORITHM USING ARTIFICIAL BEE COLONY AND PARTICLE SWARM OPTIMIZATION

Yahya GÜNER

Konya Technical University
Institute of Graduate Studies
Department of Computer Engineering

Advisor: Prof. Dr. Mustafa Servet KIRAN

2024, 72 Pages

Jury
Prof. Dr. Mustafa Servet KIRAN
Prof. Dr. İsmail BABAOĞLU
Assoc. Prof. Dr. Hüseyin HAKLI

This thesis investigates the application and effectiveness of Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO) algorithms in the optimization process of hyperparameters found in YOLOv5, the fifth iteration of the YOLO algorithm. Despite the significant advancements that YOLO has brought to the field of object detection, the requirement for expert knowledge and trial-and-error in tuning its numerous hyperparameters adds complexity to its application. In this study, nine hyperparameters of the algorithm were selected and optimized using ABC and PSO algorithms. When the training performance of the resulting models obtained using images from the Oxford-IIIT pet dataset was compared, it was observed that the model optimized with PSO tended to detect rare and challenging data and was trained with rapid convergence. In contrast, the model optimized with ABC learned all data in a balanced manner, achieving slow and steady convergence with low learning rates. During testing, the ABC-optimized model produced %81,7 mAP score which similar to its training outcomes while the PSO-optimized model produced %76,8 mAP score and exhibited overfitting and failed to perform well on the test data. These findings indicate that in the hyperparameter optimization problem of the YOLO algorithm, the ABC algorithm, which can produce a greater variety of results compared to PSO, is more conducive to achieving successful outcomes.

Keywords: ABC algorithm, hyperparameter optimization, object detection, PSO algorithm, YOLO algorithm

ÖNSÖZ

Görüntü işleme ve bilgisayarlı görü alanında yaşanan hızlı gelişmeler nesne tespiti problemi için sunulan tekniklerin durmaksızın yenisinin duyurulması ve mevcut tekniklerin güncellenmesi ile yapay zekâ alanında çalışan kesimlerde heyecan yaratmaktadır. Son dönemlerin en popüler ve hızlı ilerleyen nesne tespiti algoritmalarından biri olan YOLO algoritması tespit başarısı ve birim zamandaki tespit hızı ile dikkatleri üstüne çekmiştir. Bu durum YOLO algoritmasının akademik çevrelerce araştırmalara konu edilmesine neden olmuştur. Her ne kadar YOLO algoritması durmak bilmeyen bir şekilde geliştirilmeye ve yeni sürümleriyle ilerlemeye devam etse de algoritmanın problemlerde uygulanması hala uzmanlık gerektirmektedir. Bu durum algoritmanın kısıtlı bir kesim tarafından etkili bir şekilde kullanılmasına yol açmaktadır. YOLO algoritmasında nesne tespit modeli eğitilirken alınan karalarda etkili olan hiperparametreler bahsedilen uzmanlığa ihtiyaç duyan alanlardan biridir. Bu tez çalışmasında YOLO algoritmasının hiperparametrelerinin ABC ve PSO optimizasyon algoritmalarıyla bulunması ve bulunan sonuçların kıyaslanması amaçlanmıştır. Çalışmada bulunan sonuçların el yordamıyla hiperparametre ayarlanması sorununa çare üretecek bulgular üretmesi ve yapay zekâ çözümlerinin gelecekte geniş kitlelerce kullanılabilecek araçlar haline getirilmesinde faydası olmasını umut etmekteyim.

Yüksek lisans eğitim hayatım boyunca benimle tecrübelerini ve bilgisini benden esirgemeyen sayın danışmanım Prof. Dr. Mustafa Servet KIRAN'a sonsuz teşekkürlerimi ve minneti bir borç bilirim. Bu tez çalışmasının amacına ulaştırılması adına eleştirilerini, yorumlarını, deneyimlerini paylaşan sayın jüri üyelerine şükranlarımı sunarım. Son olarak ise tez çalışmalarım sırasında beni daima destekleyen, motive eden sevgili eşime ve sevgili aileme minnetlerimi sunarım.

Yahya GÜNER
KONYA-2024

İÇİNDEKİLER

ÖZET ..... **iv**

ABSTRACT ..... **v**

ÖNSÖZ ..... **vi**

İÇİNDEKİLER ..... **vii**

SİMGELER VE KISALTMALAR ..... **ix**

1. GİRİŞ **1**

2. KAYNAK ARAŞTIRMASI **6**

2.1. Hiperparametre Optimizasyonu Hakkında Kaynak Araştırması 6
2.2. YOLO Algoritmasının Resmi Sürümleri Hakkında Kaynak Araştırması 11

3. MATERYAL VE YÖNTEM **13**

3.1. YOLO Algoritması 13
3.1.1. YOLO algoritmasının temel çalışma prensibi 13
3.1.2. YOLOv5 algoritması 16
3.1.2.1. YOLOv5 mimarisi 17
3.1.2.2. YOLOv5 algoritmasının hiperparametreleri 20
3.1.2.3. YOLOv5 algoritmasının metrikleri 25
3.2. Parçacık Sürü Optimizasyonu Algoritması 25
3.2.1. Parçacık sürü optimizasyonu algoritmasının temel çalışma prensibi 26
3.2.2. Parçacık sürü optimizasyonu algoritmasının parametreleri 27
3.3. Yapay Arı Kolonisi Optimizasyonu Algoritması 28
3.3.1. Yapay arı kolonisi algoritmasının temel çalışma prensibi 28
3.3.2. Yapay arı kolonisi algoritmasının parametreleri 29
3.4. Oxford-IIIT Evcil Hayvan Veri Kümesi 30
3.5. Sunulan Metot 30

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA **33**

4.1. Hiperparametre Optimizasyon Aşaması 33
4.1.1. ABC ile hiperparametre optimizasyon sonuçları 33
4.1.2. PSO ile hiperparametre optimizasyon sonuçları 34
4.1.3. ABC ve PSO hiperparametre sonuçlarının kıyaslaması 34
4.2. Model Eğitimi Aşaması 36
4.2.1 ABC ile optimize edilen hiperparametreler ile eğitim sonuçları 36
4.2.2 PSO ile optimize edilen hiperparametreler ile eğitim sonuçları 39

4.2.3 Ön tanımlı hiperparametreler ile eğitim sonuçları 42
4.2.4 Model eğitimlerinin karşılaştırması 43
4.3. Eğitilen Modeller ile Test Aşaması 44
4.3.1 ABC ile optimize edilen modelin test sonuçları 44
4.3.2 PSO ile optimize edilen modelin test sonuçları 47
4.3.3 Ön tanımlı hiperparametrelerle eğitilen modelin test sonuçları 50
4.3.4 ABC ve PSO ile optimize edilen modellerin karşılaştırılması 51

5. SONUÇLAR VE ÖNERİLER 56

5.1 Sonuçlar 56
5.2 Öneriler 58

KAYNAKLAR 59

SİMGELER VE KISALTMALAR

Simgeler

S	: YOLO algoritmasında ızgara boyutu
c	: YOLO algoritmasında tespit sonucunun kanal sayısı
n_{cls}	: YOLO algoritmasında sınıf sayısı
c_1	: PSO algoritmasında kişisel güncelleme katsayısı
c_2	: PSO algoritmasında global güncelleme katsayısı
w	: PSO algoritmasında sabitlik katsayısı

Kısaltmalar

ABC	: Yapay Arı Kolonisi (Artificial Bee Colony)
AP	: Ortalama Kesinlik (Average Precision)
C3	: Concentrated-Comprehensive Convolution (Konsantre Kapsamlı Evrişim)
CIFAR-10	: Kanada İleri Araştırmalar Enstitüsü, 10 Sınıf (Canadian Institute of Advanced Research, 10 classes)
CMA-ES	: Kovaryans Matrisi Uyarlama Evrim Stratejisi (Covarience Matrix Adaptation Evolution Strategy)
ConvBNSiLU	: Evrişim, Yığın Normalleştirme, SiLU (Convolution, Batch Normalization, SiLU)
CPU	: Central Processing Unit (Merkezi İşlem Birimi)
CSPNet	: Aşamalar Arası Kısmi Ağ (Cross Stage Partial Network)
CUDA	: Birleşik Hesaplama Aygıtı Mimarisi (Compute Unified Device Architecture)
Faster R-CNN	: Daha Hızlı Bölge Tabanlı Evrişimsel Sinir Ağı (Faster Region Based Convolutional Neural Network)
GFLOPs	: Saniyede Milyar Kayan Nokta İşlem Sayısı (Billion Floating-Point Operations Per Second)
GPU	: Grafik İşlem Birimi (Graphical Processing Unit)
IoU	: Kesişimin Birleşime Oranı (Intersection over Union)
LTSM	: Uzun Kısa Süreli Bellek (Long Short-Term Memory)
mAP	: Genel Ortalama Kesinlik (Mean Average Precision)
MB	: Megabayt
MNIST	: Değiştirilmiş Ulusal Standartlar ve Teknoloji Enstitüsü (Modified National Institue of Standards and Technology)
MS-COCO	: Microsoft Bağlamı Olan Yaygın Nesneleri (Microsoft Common Objects in Context)
OneCycleLR	: Tek Döngü Öğrenme Oranı (One Cycle Learning Rate)
PCA	: Temel Bileşen Analizi (Principal Component Analysis)
PSO	: Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)
RAM	: Rastgele Erişimli Bellek (Random Access Memory)

SAES	: Vekil Destekli Evrim Stratejisi (Surrogate-Assisted Evolutionary Strategy)
SiLU	: Sigmoid Lineer Birim (Sigmoid Linear Unit)
SPPF	: Hızlı Uzamsal Piramit Havuzlama (Spatial Pyramid Pooling Fast)
SSD	: Tek Atışta Tespit (Single Shot Detection)
YOLO	: Yalnızca Bir Kez Bak (You Only Look Once)
YOLO9000	: Yalnızca Bir Kez Bak 9000 (You Only Look Once 9000)
YOLOv3	: Yalnızca Bir Kez Bak Sürüm 3 (You Only Look Once Version 3)
YOLOv4	: Yalnızca Bir Kez Bak Sürüm 4 (You Only Look Once Version 4)
YOLOv5	: Yalnızca Bir Kez Bak Sürüm 5 (You Only Look Once Version 5)
YOLOv5l	: Yalnızca Bir Kez Bak Sürüm 5 Geniş Model (You Only Look Once Version 5 Large Model)
YOLOv5m	: Yalnızca Bir Kez Bak Sürüm 5 Orta Model (You Only Look Once Version 5 Medium Model)
YOLOv5n	: Yalnızca Bir Kez Bak Sürüm 5 Nano Model (You Only Look Once Version 5 Nano Model)
YOLOv5s	: Yalnızca Bir Kez Bak Sürüm 5 Küçük Model (You Only Look Once Version 5 Small Model)
YOLOv5x	: Yalnızca Bir Kez Bak Sürüm 5 Ekstra Geniş Model (You Only Look Once Version 5 Extra Large Model)

1. GİRİŞ

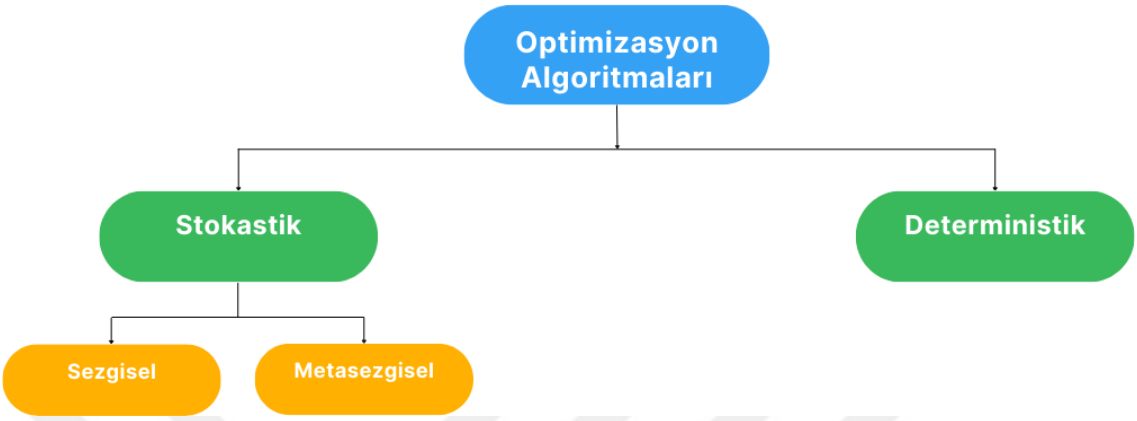
Günümüzde makine öğrenmesi ve dolayısıyla yapay zekâ tekniklerinin ilerlemesiyle birçok gerçek yaşam problemleri bilgisayarlar tarafından çözülebilir hale getirildi. Bilgisayarların insan hayatına ilk girdiği dönemlerde problemlerin çözümü için insanlar tarafından probleme özgün çözümlerin uygulandığı yazılımlar geliştirilmekteydi. Ancak günümüzde birçok problemin çözümü için makine öğrenmesi modelleri uygulanabilir hale gelmiştir. Yapay zekâ sayesinde insanın bilişsel becerilerini suni olarak gerçekleştirip insana özgü çıkarımlar yapabilen modeller geliştirilebilmektedir. (Lake ve Baroni, 2023) Bu durum günlük sorunlara çözümler getirmekle beraber kendi sorunlarını da beraberinde getirmiştir. Yapay zekâ uygulamalarında kullanılan makine öğrenmesi tekniklerinin çözülen probleme yönelik daha hızlı ve daha isabetli sonuçlar üretebilmesi için optimizasyon ihtiyacı doğmuştur.

Optimizasyon sözlük anlamı itibariyle en uygun duruma getirme işidir. Optimizasyon teknikleri bir problemi matematiksel yöntemler aracılığıyla verilen kısıtlar dahilindeki en iyi sonuca ulaştırır. Optimizasyonun uygulandığı alan bir üretimin maliyetini minimize etmek olabileceği gibi bir sosyal ağ üzerinde etkileşimleri maksimize ederek bilginin yayılımını sağlamak da olabilir. Optimizasyon problemlerinde optimize edilmesi istenen problemin matematiksel ifadesi için hedef fonksiyonu, amaç fonksiyonu gibi isimler kullanılır.

Optimizasyon algoritmaları çalışma prensiplerini göre deterministik ve stokastik olarak iki gruba ayrılmaktadır (Liberti ve Kucherenko, 2005). Deterministik algoritmalar aynı parametreler ile çalıştırıldıkları sürece hep aynı sonucu üretirken stokastik algoritmalar rassal yapıları itibariyle aynı parametrelerle defaatle çalıştırılsa bile farklı denemelerde farklı sonuçlar üretebilecek şekilde çalışmaktadır. Stokastik algoritmalar sezgisel ve metasezgisel olarak iki grupta incelenir. Optimizasyon algoritmalarının bu sınıflandırılması Şekil 1.1'de gösterilmiştir.

Sezgisel algoritmalar genel olarak belirli bir problem çözümü için o problemin alan bilgisine uygun prosedürlerin ve kuralların takip edilmesiyle hızlı bir şekilde kabul edilebilir sonuçlar almak amacı taşırlar. Metasezgisel algoritmalar ise herhangi bir problem özelinde geliştirilmeyen, çözüm arayışı sırasında rassal seçimlerden, keşfetme ve sömürme döngülerinden faydalanan optimuma yakın sonuçlar vermeyi amaçlayan yapıdadırlar. Aynı zamanda sezgisel algoritmalar belirli bir sezginin (en yakın, en maliyetsiz, en kolay vb.) uygulanmasıyla optimizasyonu amaçlarken metasezgisel

algoritmalar sezginin ötesinde tecrübeyi de (ajanların izledikleri yolların diğer ajanlara aktarılması, bilgi paylaşımı vb.) optimizasyon sürecinde kullanmaktadır (Desale ve diğ., 2015).



Şekil 1.1 Optimizasyon algoritmalarının gruplandırılması

Birbirinden farklı birçok problemi aynı yaklaşımı kullanarak çözebilen soyut yöntemler sunan metasezgisel optimizasyon algoritmalarının geliştiricileri ilhamlarını çevremizdeki olguları gözlemleyerek edinmiştir. Metasezgisel optimizasyon algoritmaları kendi içinde ilham verici kaynaklarına göre evrimsel algoritmalar, sürü zekâsı algoritmaları, fizik tabanlı algoritmalar, doğa etkileşimli algoritmalar gibi gruplara ayrılabilmektedir (Tang ve diğ., 2021).

Metasezgisel optimizasyon algoritmalarının bir alt dalı olan sürü zekâsı algoritmaları doğadaki canlı sürülerinin davranışlarından etkilenerek geliştirilen tekniklerdir. Sürü zekâsı algoritmaları çok boyutlu ve karmaşıklık derecesi yüksek olan problemlerin çözümü için çözüm üretebilmeleri ile bilinirler. Birçok farklı türü olmasıyla birlikte bu algoritmaların hepsinin çalışma prensibi bir sürü içindeki iş bölümü ile arzulanan kaynakları arayıp bulma ve bulunan kaynakların yakınındaki diğer potansiyel kaynakları sömürme şeklindedir. Parçacık Sürü Optimizasyonu (Kennedy ve Eberhart, 1995), Yapay Arı Kolonisi (Karaboğa, 2005), Karınca Koloni Optimizasyonu (Dorigo ve diğ., 2006) popüler sürü zekâsı algoritmalarındandır.

Sürü zekâsı algoritmaları farklı problemlerde sorunsuzca uygulanabilmeleri, lokal minimumları takılmadan kolayca aşabilmeleri ve istenildiği takdirde paralel programlamaya uygun yapıları ile önemli avantajlar sağlamaktadır. Bu avantajlarına karşın bu algoritmaların görece yüksek hesaplama maliyetleri ve algoritma

parametrelerindeki değişiklikler ile optimizasyonun başarısının ciddi anlamda etkilenmesi gibi faktörler de dezavantajlar olarak görülmektedir.

Makine öğrenmesi algoritmalarında optimizasyon çözümlerine konu olabilecek uygulamalar mevcuttur. Bir makine öğrenme algoritmasına yapılacak olan optimizasyon, algoritmanın çalışma prensiplerine bağlı olarak değişkenlik gösterebilir. Bundan dolayı makine öğrenmesi algoritmalarının çalışma prensiplerine göre alt gruplarda incelenmesi yapılacak optimizasyonda nasıl bir strateji izlenmesi gerektiğine dair fikir verici olacaktır. Bu bağlamda makine öğrenmesi algoritmalarını modelin eğitimi sürecinde dış bir kaynaktan gelen yönlendirme olup olmaması durumuna göre iki başlıkta inceleyebiliriz:

1. Denetimli Eğitim Algoritmaları: Bu algoritmalar eğitim için oluşturulan girdilere karşılık gelen doğru çıktılarla etiketlenmiş veriler ile beslenerek modelin eğitilmesini sağlar. Sonuçları bilinen verilerle yapılan öğrenme aşamasından sonra karşılıkları bulunmayan veriler üstünde tahminler üretilebilir. Lineer ve lojistik regresyon, karar ağaçları, derin nöral ağlar bu tipteki algoritmalara örnektir.
2. Denetimsiz Eğitim Algoritmaları: Bu algoritmalar etiketlenmiş veriler olmaksızın bir eğitim süreci amaçlar. Algoritmanın verilerdeki örüntüleri ve özellikleri keşfetmesi amaçlanır. K-means kümeleme algoritması, hiyerarşik kümeleme algoritması, PCA (Temel Bileşenler Analizi) algoritması bu tipteki algoritmalara örnektir.

Evrişimsel sinir ağları görüntü tanıma amacıyla kullanılan bir denetimli eğitim algoritmasıdır. Evrişimsel sinir ağları görüntünün tespiti ve sınıflandırılmasında başarılı sonuçlar vermektedir. Görüntü tespiti ve sınıflandırması tıbbi görüntüleme, güvenlik, endüstriyel üretim, tarım ve hayvancılık, eğitim, çevre ve doğa gibi alanlarda etkin bir şekilde kullanılmaktadır. Evrişimsel sinir ağlarının çözüm verebildiği alanların geniş bir yelpazede olması sebebiyle günümüzdeki önemi de giderek artmaktadır. Ancak her ne kadar popülerliği artsa da yüksek miktarda hesaplama gücüne olan ihtiyaçlarıyla bilinen evrişimsel sinir ağlarının eğitim aşamasının maliyeti ve uzmanlık gerektiren parametre ayarlaması alanda tecrübesi ve uzmanlığı olmayan kişiler tarafından başarılı modeller üretmesinin önünde engel teşkil etmektedir. Evrişimsel sinir ağlarının eğitimi sırasında algoritmanın gidişatını yönlendiren parametrelerin uygun ayarlanmamasından ötürü tekrar tekrar yinelenen model eğitimleri zaman ve kaynak kaybı olarak dikkat çekmektedir. Ayrıca deneme yanılma yöntemlerinin sezgisel olarak yapılmasından ötürü elde edilen sonuçların yeterliliğinin tespiti, denemeleri gerçekleştiren kişilerin

inisiyatifine ve tecrübesine kalmaktadır. Bu durum kabul edilebilir başarı yüzdesiyle çözüldüğü varsayılan problemlerin aslında daha iyi çözümlerine erişilmemesine neden olabilmektedir. Bu kayıpların önüne geçmek adına evrişimsel sinir ağlarının parametrelerinde yapılacak olan bir optimizasyon çalışması yaşanacak kayıpları azaltıp daha doğru sonuçlar üreten modellerin yaratılmasına olanak sağlayacaktır.

Redmon ve diğ. (2016) tarafından 2016 yılında geliştirilen evrişimsel siniri ağları algoritmalarından YOLO (You Only Look Once) algoritması nesne tespiti ve nesne sınıflandırmasında günümüzün en popüler algoritmalarından biridir (Jiang ve diğ., 2022). YOLO algoritmasının benzeri algoritmaların aksine girdi olarak verilen görüntüyü tek geçişte işlemesi bu algoritmanın benzerlerinden daha hızlı olmasının başlıca sebebi olmuştur (Lee ve Kim, 2020). YOLO algoritmasının kazandırdığı hız, birim zamanda yüksek yoğunlukta verinin işlendiği alanlarda akla ilk gelen algoritma olmasının önünü açmıştır. Tan ve diğ. (2021) yaptıkları çalışmada YOLO, Faster R-CNN ve SSD algoritmalarını tespit hızı ve model başarısı açısından kıyaslamışlardır. Elde ettikleri sonuçlarda YOLO ve Faster R-CNN algoritmaları benzer tespit başarıları yakalamıştır. Tespit hızı olarak ise YOLO algoritmasının Faster R-CNN algoritmasından birim sürede 8 kat daha fazla sonuç ürettiği görülmüştür. SSD algoritması ise başarı ve hız noktasında diğer iki algoritmanın yakaladığı başarıya erişememiştir (Tan ve diğ., 2021).

YOLO algoritması diğer evrişimsel sinir ağlarında olduğu gibi model eğitimi esnasında ciddi miktarda işlem gücü ve zaman gerektirmektedir. Bu algoritmanın daha başarılı sonuçlar üretmesinde etkisi olan en önemli faktörlerden birisi de hiperparametre olarak bilinen ve model eğitimi sırasında yapılan işlemlerin ve karar anlarındaki seçimlerin gidişatını doğrudan etkileyen değişkenlerdir. YOLO algoritmasında eğitilen her bir veri kümesinin optimal çözümü için hiperparametreler çok farklı çeşitlilikteki değerleriyle uygulanabilir. Eğitilen modelden optimum sonucu alabilmek için hiperparametreler üstünde ustalık gerektiren ince ayarlamalar yapılması gerekmektedir.

Hiperparametrelerin ayarlanması görevi günümüzde hala yaygın olarak deneme yanılmalar yapılarak yerine getirilmektedir. Ancak bu şekilde yapılan ayarlamalar sezgisel bir ilerleyişe sahip olup herhangi bir sistematik barındırmadığı için optimum noktaların es geçilmesi oldukça olasıdır. Bunun yerine optimizasyon algoritmalarının kullanılması optimum sonuçlar elde etmek adına daha sağlıklı sonuçlar üretecektir.

Son yıllarda YOLO algoritmasının popülerlik kazanmasına paralel olarak algoritmanın hiperparametrelerinin optimizasyonunu konu alan yeni yöntemler sunulmuştur. Bu çalışmada ise YOLO algoritmasının metasezgisel optimizasyon

algoritmaları olan ABC ve PSO algoritmaları ile optimizasyonu yapılmıştır ve optimizasyon sonuçları karşılaştırmalı olarak incelenmiştir.

Bu tez çalışmasının daha sonraki bölümlerinde öncelikle makine öğrenmesi tekniklerinde optimizasyonu ve özellikle YOLO algoritmasının hiperparametrelerini optimize etmeyi amaçlayan kaynak araştırmaları derlenerek sunulmuştur. Ardından materyal ve metot bölümünde bu çalışmada kullanılacak olan YOLO, PSO ve ABC algoritmalarından ve Oxford-IIIT veri kümesinin tanıtımı yapılmış olup bu materyal ve metotlar kullanılarak yapılan deneyin nasıl hazırlandığından bahsedilmiştir. Devamında ise sonuçlar ve öneriler bölümünde elde edilen sonuçların analizi yapılıp, PSO ve ABC algoritmalarının YOLO hiperparametrelerini optimize etmedeki başarıları kıyaslanmıştır. Son olarak da elde edilen sonuçlar yorumlanmış ve YOLO hiperparametrelerini optimize etmek amacıyla yapılabilecek gelecek çalışmalar için öneriler sunulmuştur.

2. KAYNAK ARAŞTIRMASI

2.1. Hiperparametre Optimizasyonu Hakkında Kaynak Araştırması

Bu tez çalışmasında önerilen yönteme yardımcı olması amacıyla makine öğrenmesi tekniklerinde yapılan optimizasyon çalışmaları araştırılmıştır. Bu amaçla hiperparametreleri manuel ayarlamaya yönelik çalışmalar, hiperparametrelerin optimizasyon teknikleriyle ayarlandığı çalışmalar ve hiperparametrelerin birden çok optimizasyon tekniği ile ayarlanıp karşılaştırıldığı çalışmalar incelenmiştir. Çalışmaların özetleri literatür gelişiminin görülebilmesi amacıyla yıllara göre kronolojik bir sıra ile verilmiştir.

Bengio (2012) yaptığı çalışmada bir sinir ağının eğitilmesine yönelik rehber niteliğinde olabilecek önerilerde bulunurken ağın hiperparametrelerinin ayarlanması konusuna da değinmiştir. Çalışmasında derin öğrenme modellerinin eğitiminden önce ön eğitim (pretraining) işlemlerinin elde edilen modeldeki başarıyı arttırdığından, hiperparametre ayarının başarılı bir model oluşturmadaki anahtar rolünden bahsetmiştir. Öğrenme oranı, momentum katsayısı, ağırlık düzenleme gibi parametrelerin önemine vurgu yapılmış olup 10'un üzerinde hiperparametrenin özelliklerinden ve nasıl ayarlanması gerektiğinden bahsedilmiştir. Hiperparametreler tanıtıldıktan sonra el ile arama (manual search) ve her bir parametre için önceden belirlenen uygun değerlerin kombinasyonlarını sırayla deneyen ızgara araması (grid search) ile bu parametrelerin ayarlanmasına yönelik tavsiyeler sunulmuştur.

Bergstra ve Bengio (2012) yaptıkları çalışmada rastgele arama adı verdiklerini bir hiperparametre optimizasyon yöntemini tanıtmışlardır. Öne sürülen yöntem hiperparametrelerin arama uzayından rastgele seçilen değerlerin önceden tanımlanmış bir durma koşulu sağlanana kadar denenmesine dayanır. Sunulan yöntem koşut çalışabilmesi ve işlem gücünü görece az kullanması sebebiyle düşük maliyetli olarak nitelendirilmiştir. Çalışmada sunulan yöntemin dezavantajı olarak rassallıktan dolayı daha iyi performans verebilecek parametrelerin atlanabilmesi ihtimali gösterilmiştir. Ancak buna rağmen kıyaslandığı diğer yöntemlere göre başarılı sonuçlar verdiği öne sürülmüştür.

Snoek ve diğ. (2012) hiperparametre optimizasyonunun önemine değindikleri çalışmalarında Bayes Optimizasyonu (Bayesian Optimization) ile hiperparametre optimizasyonu denemişlerdir. Bayes optimizasyonu amaç fonksiyonunu bir olasılık modeli vasıtasıyla modelleyerek sonuçlar üretmeye çalışır. Optimizasyonun iterasyonları

ilerledikçe örneklem uzayındaki belirsizliklerin azalmasıyla daha optimize sonuçlar elde edilir. Bayes optimizasyonu ile arama ve sömürme arasındaki kazanım farkı ilk iterasyonlarda daha çok arama lehine iken sonuca yaklaştıkça sömürme lehine ilerler. Çalışmada lojistik regresyon, karar destek makineleri ve evrişimsel sinir ağı üzerinde bu yöntem kullanılarak hiperparametre optimizasyonu denenmiştir. Çıkan sonuçlar el ile ayarlanan hiperparametrelerden elde edilen sonuçlarla kıyaslandığında Bayes Optimizasyonunun daha iyi sonuçlar verdiği görülmüştür.

Domhan ve diğ. (2015) yaptıkları çalışmada hiperparametre optimizasyonunun zaman açısından maliyetli bir iş olduğunu belirtmişlerdir ve hiperparametre optimizasyonu esnasında geçen zamanı kısaltmak için öğrenme eğrisinde ekstrapolasyon yapılmasını önermişlerdir. Ekstrapolasyon, eldeki verilerden yararlanarak gelecek verilerin ne olabileceğine dair matematiksel çıkarım yapma tekniğidir. Eğitilen modelin öğrenme eğrisi üstünde yapılan ekstrapolasyon sayesinde optimizasyon sırasında kötü sonuç verecek eğitimlerin atlanması sağlanmıştır. Yapılan deneylerde eğitim süresinin kısaldığı ve üretilen modelin başarımının el ile ayarlanan modellerden yüksek olduğu görülmüştür.

Loshchilov ve Hutter (2016) hiperparametre optimizasyonu için Bayes optimizasyonu, ızgara araması ve rastgele arama algoritmalarına alternatif olarak "kovaryans matrisi uyarlama evrim stratejisi algoritmasını" (CMA-ES) önermişlerdir. CMA-ES doğrusal olmayan ve dışbükey olmayan problemlerde sıkça kullanılan bir evrimsel algoritmadır. CMA-ES algoritmasında popülasyondaki her bir birey bir çözüm örneğini teşkil ederken iterasyonların sonunda tüm popülasyonun kovaryans matrisi oluşturularak çözüm adaylarının dağılımı incelenir. Dağılım sayesinde popülasyonun gelişimi ve ilerleyişi yönlendirilir. Popülasyon içindeki en iyi bireyler ileri jenerasyonlara aktarılarak algoritmanın optimuma yakınsaması sağlanır. MNIST veri kümesinde yapılan deney sonucunda CMA-ES algoritmasının Bayes Optimizasyonuna rakip olabilecek sonuçlar ürettiği görülmüştür.

Lorenzo ve diğ. (2017) yaptıkları çalışmada evrişimsel sinir ağlarının hiperparametrelerini PSO algoritması ile optimize etmeyi önerdiler. MNIST ve CIFAR-10 veri kümelerinde yapılan deneylerde çalışmanın yapıldığı dönemde optimizasyon standardı sayılan ızgara araması ve rastgele arama algoritmalarıyla yapılan optimizasyonlara kıyasla daha başarılı sonuçlar elde ettiler. Aynı yıl içinde Lorenzo ve diğ. (2017) PSO algoritmasını koşutlaştırarak optimizasyon sürecini daha hızlı hale getirmek için çalışmışlardır. Önceki çalışma ile aynı deney düzeneği kullanılmış ve

sonuçlar karşılaştırılmıştır. Koşut olarak yapılan deneyde eğitim ve optimizasyon süreleri önemli miktarda kısalmışken önceki deneyin sonuçlarına benzer başarımlar elde edilmiştir.

Ozaki ve diğ. (2017) hiperparametre optimizasyonu için Nelder-Mead optimizasyon algoritmasının kullanımını sunmuşlardır. Nelder-Mead yöntemi bir başlangıç noktasından iteratif adımlarla ilerleyen optimizasyon algoritmasıdır. Algoritma bir başlangıç noktasından başlayarak fonksiyonun maksimum ve minimum noktalarını bulmaya çalışır. Başlangıç noktaları arama uzayındaki yerlerine yerleştirilerek "simplex" adı verilen bir şekil oluşturulur. Her adımda o iterasyonda bulunan noktalar üstünde yakınsama/genişleme ve içe alma/daraltma işlemleri yapılarak optimum noktaların korunması, kötü sonuç üreten noktaların atılması sağlanır. Bu sayede bir sonraki adımdaki simplex şekli hesaplanır. İteratif ilerleyen bir algoritma olduğu için lokal minimum noktalardan kolay sıyrılabilen bir algoritmadır. Ancak bütün noktalarda iteratif olarak gezinmesinden dolayı çok boyutlu problemlerde performans sorunları oluşabilmektedir. Algoritmanın bitiş kriterleri belirli bir iterasyon sayısına ulaşmak, optimize edilen fonksiyonun sonucunun belirlenmiş eşik değeri aşması olarak ayarlanabilmektedir. Yazarlar Nelder-Mead yöntemini çeşitli derin öğrenme modeline uygulayıp kesinlik, duyarlılık, F1 skoru gibi metrikler üzerinden değerlendirme yapmışlardır. Yapılan değerlendirmeler sonucunda Nelder-Mead yönteminin hiperparametre optimizasyonunda rastgele arama, Bayes optimizasyonu ve CMA-ES algoritmasından daha iyi sonuçlar ürettiği görülmüştür.

Aszemi ve Dominic (2019) evrişimsel sinir ağlarında eğitilen her veri kümesi için o probleme özgü hiperparametreler seçilmesi gerektiğine dikkat çekmişler ve bunun için genetik algoritmalar ile lokal arama tekniklerinin birleşimini kullanan bir optimizasyon tekniği sunmuşlardır. CIFAR-10 veri kümesi üzerinde yapılan farklı düzenlerde oluşturulmuş 3 deney sonunda kayda değer bir optimizasyon elde edememişlerdir ve genetik algoritma kullanımını işlem gücü açısından maliyetli olarak belirtmişlerdir.

Cabada ve diğ. (2019) akıllı ders sistemlerinde kullanılmak üzere öğrencilerin yüz ifadelerinden duygu durumlarına dair çıkarım yapabilecek bir yapay sinir ağı modeli önermişlerdir. Yapılan çalışma evrişimsel sinir ağı kullanılarak eğitilmiş olup ağın hiperparametrelerinin optimizasyonu için genetik algoritma kullanılmıştır. Yapılan deneyin sonucunda %82 oranında başarı elde edildiği belirtilmiş ve daha önceki çalışmada "deneme yanılma" tekniği ile oluşturulan hiperparametreler kullanılarak eğitilen modele göre %8 daha fazla başarı elde ettikleri belirtilmiştir.

Guo ve diğ. (2020) evrişimsel sinir ağlarının hiperparametrelerinin optimizasyonu için dağıtık PSO algoritması kullanımını önermişlerdir. Optimizasyon sırasında her bir parçacığın işletilmesi ve dolayısıyla sinir ağının eğitimi dağıtık bir şekilde gerçekleştirilmiştir. MNIST veri kümesi üstünde yapılan deney sonucunda hiperparametrelerin başarıyla optimize edildiği görülmüştür. Aynı zamanda dağıtık optimizasyon sayesinde optimizasyon ve eğitim süreci daha hızlı sonuçlanmıştır.

Andonie ve Florea (2020) yaptıkları çalışmada evrişimsel sinir ağlarının hiperparametre optimizasyonu için yeni bir yöntem sunmuşlardır. Ağırlıklı Rastgele Arama (Weighted Random Search) adı verilen yöntemde klasik rastgele aramada olduğu gibi rassal noktalar seçilmiştir. Noktaların seçiminden sonra model eğitimi yapılıp elde edilen sonuca göre seçilen noktaların ağırlık puanı alması önerilmiştir. Ağırlıklandırma sayesinde sonraki iterasyonlarda başarısız olan hiperparametrelerin benzerlerinin önerilmesinin önüne geçilmiştir. Önerilen yöntemin evrişimsel sinir ağının hiperparametrelerinin optimizasyonunda etkili olduğu görülmüş olup hesaplama gücü açısından verimli olduğu belirtilmiştir.

Zhang ve diğ. (2021) akciğer kanserinin tespiti amacıyla akciğer nodüllerinin iyi ve kötü huylu olma durumları üzerinden sınıflandırmasını yapan bir evrişimsel sinir ağı modeli ve optimizasyonu sunmuşlardır. Yapılan çalışmada vekil destekli evrim stratejisi (surrogate-assisted evolutionary strategy, SAES) kullanan evrimsel algoritma ile bir hiperparametre optimizasyon yöntemi tanıtılmıştır. SAES yöntemi ile evrimsel algoritmadaki bireylerin gerçek fonksiyonu çalıştırması yerine gerçek fonksiyonun vekili olarak tanımlanan bir istatistiksel modeli (Gauss süreci, rastgele orman vb.) çalıştırması amaçlanır. Bu sayede çok maliyetli olduğu varsayılan gerçek fonksiyon yerine görece daha az maliyetli olan bir işlem gerçekleştirilir. Sunulan yöntemin ızgara araması ve rastgele aramaya göre daha başarılı olduğu görülmüştür ve akciğer nodüllerinin sınıflandırmasında tahmin performansında artış sağladığı belirtilmiştir.

Alibrahim ve Ludwig (2021) yaptıkları çalışmada Bayes optimizasyonu (Bayesian Optimization), ızgara araması (Grid Search) ve genetik algoritmayı yapay sinir ağlarında hiperparametre optimizasyonu açısından karşılaştırmışlardır. Bu üç yöntem ortalama hata karesi (mean squared error), ortalama logaritmik hata karesi (mean squared logaritmic error), kesinlik (precision), duyarlılık (recall) ve F1 skoru metrikleri üzerinden kıyaslanmıştır. Elde edilen sonuçlarda tüm yöntemlerin benzer hiperparametreler ürettiği ve yakın sonuçlar verdiği görülmüştür. Ancak genetik algoritmanın diğer iki algoritmadan biraz daha iyi sonuç verdiği görülmüştür.

Erkan ve diğ. (2022) yılında bitki görsellerinden bitki türünün tespiti için hiperparametrelerinin yapay arı kolonisi algoritması ile optimize edilen bir yapay sinir ağı modeli sunmuşlardır. Yapılan deney sonucunda kesinlik (accuracy), duyarlılık (sensitivity), F1 skoru metrikleri üzerinden yapılan kıyaslamalarda sunulan metodun döneminin kabul gören metotlarının sonuçlarına göre daha başarılı olduğu görülmüştür.

Wang ve diğ. (2022) meyve sanayisinde elmaların sapının ve kaliks adı verilen dip kısmının görünümünün elma kalitesinin belirlenmesinde önemli ayırıcı unsurlar olduğunu belirtmiş ve bu sınıflandırmanın hala işçilerin gözlemiyle altını çizmişlerdir. Wang ve diğ. çalışmalarında sundukları yöntemde YOLOv5 algoritmasını elmaların sap ve kaliks görüntüleriyle eğiterek kaliteli ve kalitesiz elmaların sınıflandırılması amaçlanmıştır. Yapılan deneyde YOLO algoritmasının COCO veri kümesi için hazırladığı ön tanımlı hiperparametreler ve 320 jenerasyon boyunca çalışan bir evrimsel algoritma aracılığıyla optimize edilmiş hiperparametreler ile 2 farklı model eğitilmiştir. Evrimsel algoritmalar kullanılan hiperparametrelerle eğitilen modelin daha başarılı sonuçlar verdiği ancak saniyedeki kare tahmininde daha yavaş olduğu görülmüştür.

Kouhalvandi ve Matekovits (2022) yaptıkları çalışmada antenlerin uzun vadeli çalışmalardaki frekans tepkilerini öngörebilmek için uzun kısa süreli bellek (Long short-term memory, LSTM) derin sinir ağı modeli için hiperparametre optimizasyonu sunmuşlardır. LTSM yapısı zaman serileri gibi sıralı verilerin bulunduğu problemlerdeki uzun vadeli bağımlılıkları bulmakta kullanılan sinir ağı algoritmasıdır. Çalışmada sunulan LTSM sinir ağının hiperparametreleri genetik algoritma, yapay arı kolonisi algoritması, parçacık sürü optimizasyonu algoritması, Thompson örnekleme algoritması gibi optimizasyon teknikleri ile optimize edilmiştir. Yapılan deneyler sonucunda en başarılı optimizasyon Thompson örnekleme algoritması ile yapılmışken en az başarılı sonucu ise genetik algoritma vermiştir. Yazarlar sunulan yöntemin başarı elde ettiğini belirtip 6G teknolojisinde anten ağı tasarımı için kazanımlar sağlayacağını belirtmişlerdir.

Karaman ve diğ. (2023) kolorektal kanser tanısında destek olması amacıyla YOLO algoritması ile polip dokularının tespiti için bir yapay sinir ağı modeli geliştirmişlerdir. Kanser tanısında en ufak başarı artışının bile hayati olduğuna vurgu yaparak YOLOv5 algoritmasının hiperparametrelerini ve aktivasyon fonksiyonlarını yapay arı kolonisi algoritması aracılığı ile optimize etmişlerdir. Sunulan yöntemde veri kümesi öncelikle YOLO algoritması tarafından sağlanan ön eğitimli model kullanılarak eğitilmiştir. Bu işlemin ardından elde edilen yeni model hiperparametre optimizasyonu amacıyla ABC algoritmasında kullanılmıştır. Optimizasyon sırasında amaç fonksiyonu

olarak eğitim sonucunda elde edilen genel ortalama kesinlik (mean average precision, mAP) metriği seçilmiştir. ABC ile yapılan optimizasyondan elde edilen hiperparametreler ile yeni bir model eğitimi daha yapılmıştır ve bu yeni model test için kullanılmıştır. Geliştirilen yapının YOLO algoritmasının parametrelerini başarıyla optimize ettiği görülmüştür.

2.2. YOLO Algoritmasının Resmi Sürümleri Hakkında Kaynak Araştırması

Algoritmanın ilk sürümü 2016 yılında Redmon ve diğ. (2016) tanıtılmıştır. Tek geçişte görseldeki tüm nesneleri bulmayı amaçlayan yaklaşımıyla dikkat çeken algoritma dönemi için kabul edilebilir sonuçlar vermesine karşın birbirine yakın küçük nesneleri tespit edememesi, her bir ızgarada tek bir obje tespit edebilmesi ve eğitilen nesnelerin test sırasında farklı en boy oranlarında görünce tespit edememesi gibi sorunlara sahipti.

YOLO algoritmasının ilk sürümündeki bu problemlerden ötürü Redmon ve diğ. (2017) bir sene sonra YOLO9000 adını verdikleri YOLO algoritmasının ikinci sürümünü tanıttılar. Tanıtılan yeni algoritmada evrişimsel sinir ağı mimarisindeki yapılan değişiklikler ve çapa kutuları tanıtıldı. Evrişimsel sinir ağının omurgası DarkNet adı verilen mimari ile güçlendirildi. Bu sayede özellik çıkarımlarının daha başarılı yapılması amaçlandı. Çapa kutuları vasıtasıyla ise bir ızgara içinde farklı en boy oranına sahip birçok nesnenin tespitinin yapılabilmesi sağlandı. Önceki sürümdeki veri kümesi üzerinden yapılan deneylerde hem tespit başarısı hem de tespit hızı konusunda başarı elde edildiği belirtildi. Bu sürümde de ilk sürümde olduğu küçük nesnelerin tespitindeki sorunlar devam etmekteydi.

Redmon ve Farhadi (2018) algoritmada yaptıkları güncellemeleri yayınlayarak YOLO algoritmasının üçüncü sürümünü duyurdular. Bu sürümde çoklu ölçekli tahminler (Multi-Scale Predictions) adını verdikleri yöntemle küçük nesneleri de tespit edebilmeye başladılar. Çok ölçekli tahminler yöntemi küçük nesneleri tespit edebilmek için girdileri üç farklı boyuttayken (küçük, orta ve büyük) tespit mekanizmasına teker teker sokarak farklı ebatlardaki nesnelerin tespitini ayrıştırmışlardır. Aynı zamanda bu çalışmada algoritmanın sinir ağı mimarisinde geliştirmeler yapılmıştır. Bu çalışmada yapılan deneyler dönemin popüler test veri kümesi olan MS-COCO üzerinde yapılmıştır. Elde edilen sonuçlara göre YOLOv3 algoritması çağdaşı metotlarla benzer başarım gösterirken bunu diğer algoritmalardan iki kat daha hızlı başarmıştır.

Bochkovskiy ve diğ. (2020) YOLO algoritmasını özgün geliştiricilerinin bıraktığı noktadan devam ettirerek algoritmanın dördüncü sürümünü duyurdular. Bu sürümde eğitim süresini hızlandıran ve küçük nesnelerin tespitini kolaylaştıran model mimarisi geliştirmeleri tanıtıldı. Ayrıca model mimarisine uzamsal piramit havuzlama katmanı eklenerek farklı boyutlardaki cisimlerden öğrenilen aynı özelliklerin sabit boyutta bir özellik vektörüne dönüştürülmesi sağlandı. Mimari değişikliklerin yanı sıra eklenen yeni stratejilerle algoritmanın hızı ve başarımı arttırıldı. Yayımlanan çalışmada YOLOv4 algoritmasının YOLOv3'e göre hız ve tespit başarısı açısından daha üstün olduğu belirtildi.

Jocher ve diğ. (2020) Ultralytics LLC şirketinin çatısı altında YOLO algoritmasının beşinci sürümünü duyurdular. YOLO algoritmasının önceki sürümlerden farklı olarak tamamen Python dili ve Pytorch kütüphanesi kullanılarak geliştirildi. Bu sayede algoritmanın daha geniş kitleler tarafından kullanılabilmesi amaçlandı. Bu sürümde de geliştiriciler sinir ağı modelinin omurgasını CSPNet ağı ile değiştirerek daha az hesaplama gücü kullanılmasını sağladılar. Ayrıca bu sürüm kapsamında farklı boyutlardaki beş farklı sinir ağı mimarisi sunuldu ve algoritmanın çeşitli güçlerdeki sistem kaynaklarıyla o sisteme uygun mimari kullanılarak çalıştırılabilmesi sağlandı. Yapılan deneyler sonucunda algoritmanın bu sürümünün önceki sürümlere kıyasla daha performanslı çalıştığı ve güçlü sonuçlar verdiği görüldü.

3. MATERYAL VE YÖNTEM

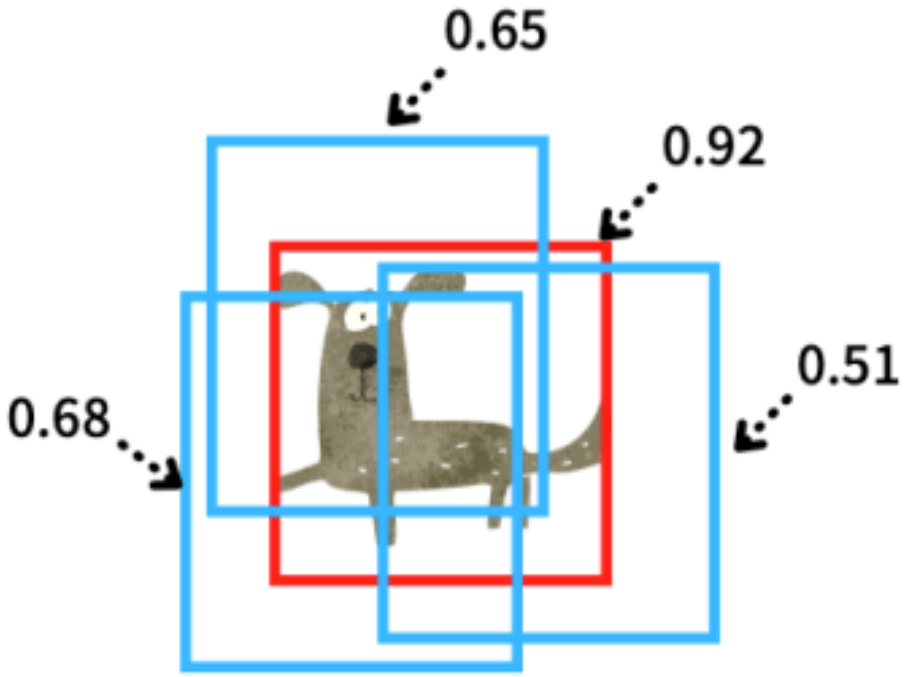
Çalışmanın bu bölümünde deney için kullanılan materyaller ve yöntemler tanıtılmıştır. Öncelikle, tez çalışması için yapılan deneyde optimize etme amacıyla kullanılmış olan YOLO algoritmasının genel işleyişi, algoritmanın resmi sürümlerinin tanıtımı ve algoritmanın deneyde kullanılan sürümü olan YOLOv5 algoritmasının tanıtımı ve çalışma prensipleri anlatılmıştır. Daha sonra optimizasyonda kullanılan yapay arı kolonisi algoritmasının ve parçacık sürü optimizasyonu algoritmasının tanıtımı yapılıp işleyişleri açıklanmıştır. Deneyde kullanılan yöntemler tanıtıldıktan sonra deneyin gerçekleştirildiği Oxford-IIIT Evcil Hayvan veri kümesi tanıtılmıştır. Son olarak ise deneyde bu materyal ve metotların çalıştırıldığı ortam ve nasıl kullanıldıkları anlatılmıştır.

3.1. YOLO Algoritması

YOLO, İngilizce "Yalnızca Bir Kez Bak" anlamına gelen You Only Look Once söz grubunun kısaltması olan bir evrişimsel sinir ağı algoritmasıdır. Redmon ve diğ. (2016) tarafından geliştirilen bu algoritma görsellerdeki nesneleri tek taramada tespit edebilmesiyle saniyede çok fazla karede tespit yapabilmesi ve yüksek başarım vermesi sebebiyle obje tanıma problemlerinde yaygın halde kullanılmaya başladı.

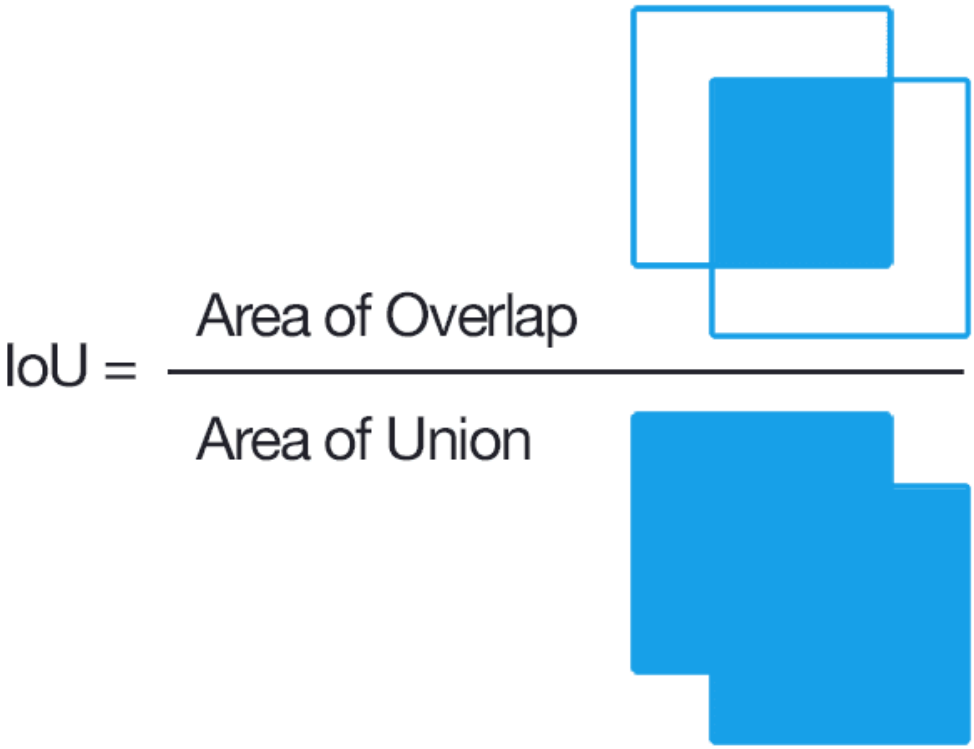
3.1.1. YOLO algoritmasının temel çalışma prensibi

YOLO algoritması bir görüntüyü SxS boyutunda ızgaralara bölerek her bir ızgara bölmesinde obje tespit etmeye çalışır. Izgaralarda obje varlığı tespit edilirken objenin orta noktasının bulunduğu ızgara bulunur. Orta noktanın bulunduğu ızgara obje için yükseklik, genişlik, sınıf ve güven skorunu hesaplar ve bu değerleri kullanarak objeye ait sınırlayıcı kutu çizmekten sorumlu olur. Bir nesne için çizilen sınırlayıcı kutular Şekil 3.1'de sunulmuştur.

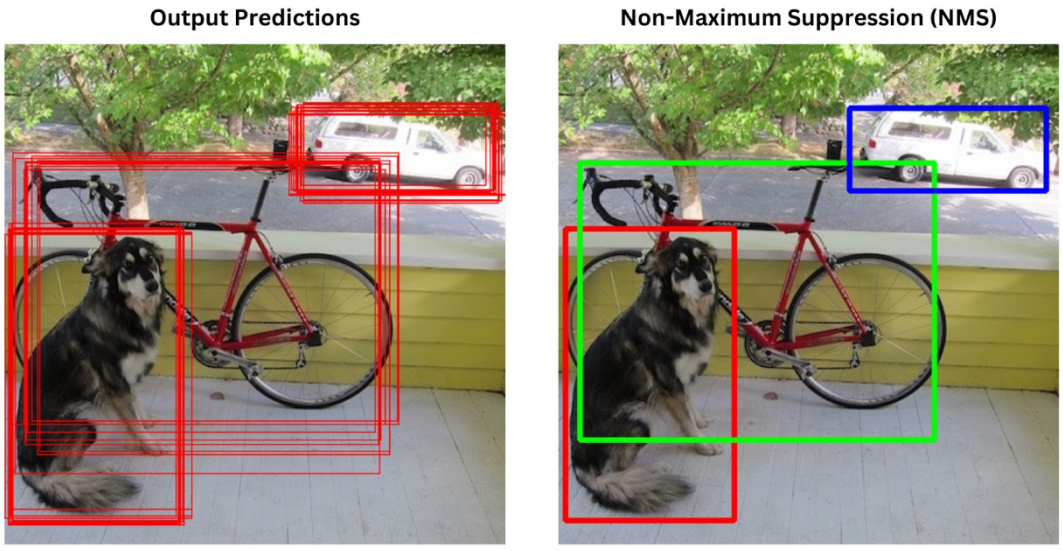


Şekil 3.1 Bir nesne için çizilen sınırlayıcı kutular

Algoritma bir obje için birden fazla sınırlayıcı kutu çizebildiğinden dolayı içlerinden en uygun olan sınırlayıcı kutunun seçilmesi gerekmektedir. Bunun için maksimum olmayanı baskılama (Non-Maximum Suppression) yöntemi uygulanır. Bu yöntem her bir sınıf için çizilen sınırlayıcı kutular arasından en yüksek ihtimale sahip sınırlayıcı kutuyu referans alıp diğer sınırlayıcı kutularla kesişimin birleşime oranı (Intersection over Union, IoU) üzerinden kıyaslama yapar. Eğer IoU değeri belirlenen eşik değerinden büyükse o sınıf için tahmin olasılığı düşük olan sınırlayıcı kutu kendisinden daha iyi kapsama sahip bir kutu var olduğu için silinir. IoU değerinin görsel ifadesi Şekil 3.2'de, maksimum olmayanı baskılama işleminin uygulanmasının öncesi ve sonrasında oluşan durumlar Şekil 3.3'te sunulmuştur.



Şekil 3.2 Kesişimin birleşime oranı



Şekil 3.3 Maksimum olmayanı baskılama işlemi

YOLO algoritmasının ilk sürümünde her bir ızgara yalnızca bir obje için sonuç üretebilmekteydi. Bu sorunun çözümü için ise YOLO algoritmasının ikinci sürümü itibariyle çapa kutuları tekniği geliştirildi ve her bir ızgarada ön tanımlı olan çapalar için sınıf sonuçları üretmeye başlandı. Bu sayede farklı en boy oranına sahip objelerin aynı

ızgarada yer almaları durumunda her objenin ayrı ayrı tespit edilebilmesi sağlandı. YOLO algoritmasının sözde kodu Şekil 3.4'te verilmiştir.

YOLO Nesne Tespiti Algoritması

```
Girdi: girdi_goruntu
Çıktı: nesne_tespitleri
function YOLO(girdi_goruntu)
    on_islenmis_goruntu ← OnIsleme(girdi_goruntu)
    ozellik_haritalari ← SinirAgiGecisi(on_islenmis_goruntu)
    tahminler ← boş liste
    for ozellik_haritasi in ozellik_haritalari do
        for izgara_hucre in ozellik_haritasi do
            for anchor_kutu in izgara_hucre do
                kutu_koord, sinif_skorlar, guven_skoru ← TahminleriHesapla(anchor_kutu)
                if guven_skoru > belirli_bir_esik then
                    tahmin ← {'Kutu': kutu_koord, 'Sinif': sinif_skorlar, 'Guven': guven_skoru}
                    tahminler.append(tahmin)
                end if
            end for
        end for
    end for
    son_tahminler ← NonMaximumSuppression(tahminler)
    return son_tahminler
end function
```

Şekil 3.4 YOLO algoritmasının sözde kodu

3.1.2. YOLOv5 algoritması

YOLOv5, Jocher ve diğ. (2020) tarafından tanıtılmış olan YOLO algoritmasının beşinci sürümüdür. Bu sürümde önceki versiyonlarda kullanılan C dili ve CUDA kütüphanesinin bırakılıp Python dili ve Pytorch kütüphanesine geçilmiştir. Ayrıca sinir ağının mimarisi farklı sayıda parametrelere ve boyutlara sahip Çizelge 3.1'de özellikleri verilen beş ön tanımlı modelde uygulanmıştır.

Çizelge 3.1. YOLOv5 algoritmasının sinir ağı modelleri

Model Adı	Katman Sayısı	Toplam Parametre (Milyon)	İşlem Gücü Gereksinimi (GFLOPs)	Diskteki Yaklaşık Boyutu (MB)
YOLOv5n (Çok Küçük)	150	1,9	4,5	7
YOLOv5s (Küçük)	212	7,2	17,0	14
YOLOv5m (Orta)	308	21,2	51,0	42
YOLOv5l (Büyük)	408	47,0	115,0	93

YOLOv5x (Çok Büyük)	508	87,0	205,0	173

Algoritma ile sunulan bu modellerin omurga, boyun ve kafa yapılarında kullanılan mimariler birebir aynı olmakla beraber bu modellerin arasındaki farklılık kullanılan parametrelerin sayısı, tekrarlı olarak kullanılan evrişimsel katmanların model içindeki kullanım sıklığının değişmesinden dolayı oluşmaktadır. Algoritmada sunulan farklı modeller sayesinde çeşitli niteliklere ve işlem güçlerine sahip ortamlara (mobil, masaüstü, sunucu vb.) uygun model eğitimlerinin yapılabilmesi sağlanmaktadır.

3.1.2.1. YOLOv5 mimarisi

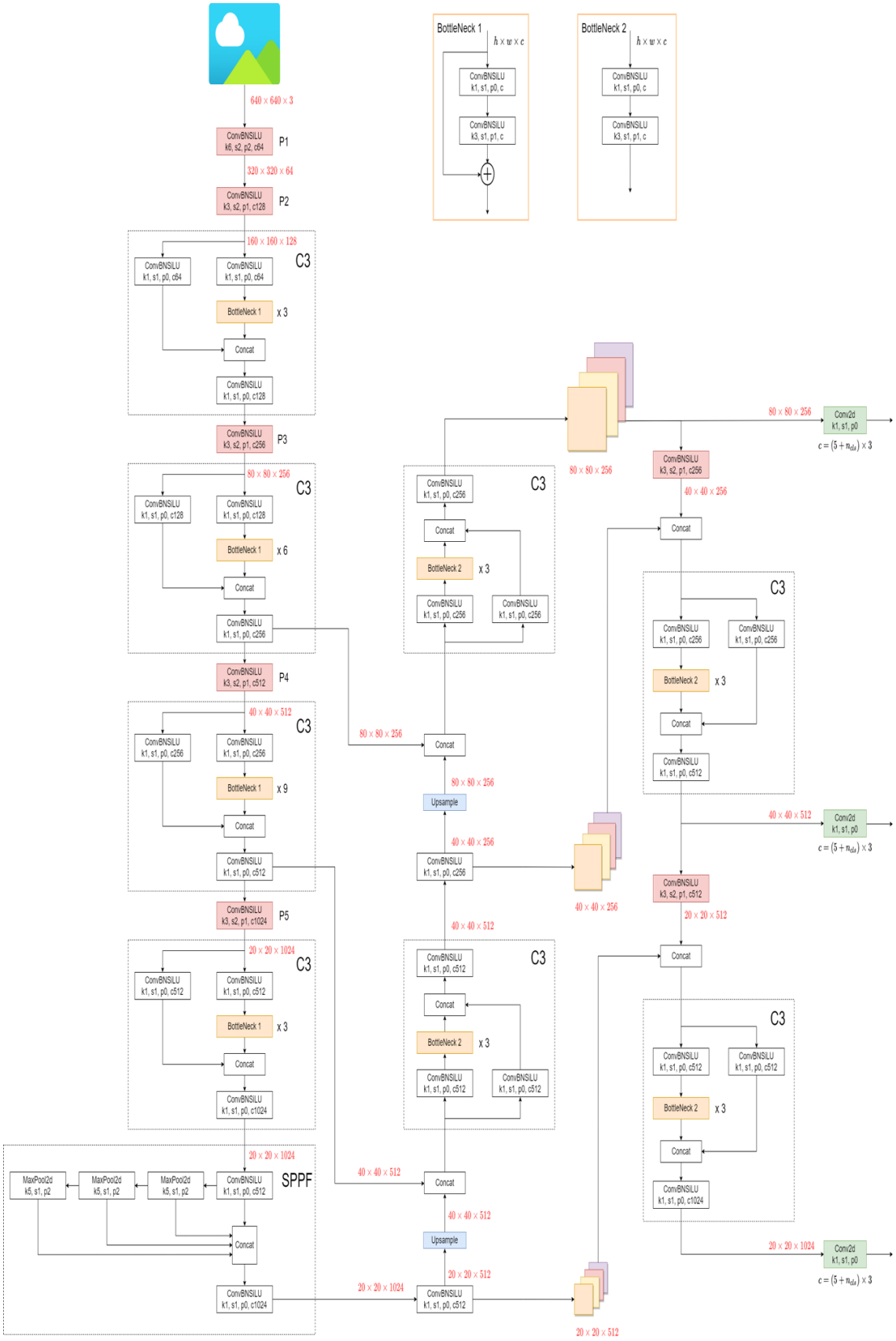
YOLOv5 algoritmasının mimarisi diğer evrişimsel sinir ağlarında olduğu gibi omurga, boyun ve kafa kısımlarından oluşacak şekilde 3 bölümden oluşmaktadır.

Modele omurga aşamasında girdi olarak verilen görüntü (Algoritmada girdilerin varsayılan en boy çözünürlüğü 640x640 olarak kabul edilmektedir.) ConvBNSiLU adı verilen bir evrişim katmanı, bir yığın normalleştirme (batch normalization) işlemi ve bir SiLU aktivasyonu işleminden oluşan yapıdan birkaç kez art arda geçirilerek görüntüden özellik öğrenimi sağlanır. Ardından elde edilen özellik haritası C3 modülüne girdi olarak verilir ve özellik haritası C3 modülünde iki kol üzerinden ilerler. Kollardan biri ConvBNSiLU ve darboğaz (bottleneck) işlemlerinden geçerken diğeri doğrudan sonuç aşamasına iletilir. C3 modülü en sonunda bu iki kolu birleştirip yeni bir özellik haritası olarak sonuç verir. C3 modülünde yapılan bu işlemler sayesinde hem gradyan kaybı önlenir hem de yeni özelliklerin öğrenmesi sağlanır. C3 modülü birkaç kez tekrarlanarak daha derin katmanlarda yeni özellik haritalarının oluşturulur. Özellik haritaları yüzeysel katmanlarda daha detaylı özellikleri ve küçük nesneleri öğrenebilecek şekilde oluşurken derin katmanlardaki özellik haritaları daha genel özellikleri ve büyük nesneleri öğrenecek şekilde oluşmaktadır. C3 modüllerinden oluşan özellik haritaları bir koldan sıradaki C3 modülüne işlenmesi için iletilirken başka bir koldan da boyun modüllerine özellik zenginleştirme için iletilmektedir.

Modelin boyun aşamasında omurga aşamasının en son C3 modülünden gelen özellikler ile beslenen Hızlı Uzamsal Piramit Havuzlama (Spatial Pyramid Pooling Fast, SPPF) modülü ile özellik haritalarından gelen bilgiler özellik boyutu sabitlenen (Mekânsal boyutlar girdi görüntüsünün boyutlarına göre değişkenlik gösterecektir.) bir

özellik haritasında birleştirilir ve farklı boyutlardaki girdiler için çözüm üretebilecek bir özellik haritası sağlanır. SPPF ve omurga yapısından gelen özellik haritaları her bir katman için birleştirilerek tespitlerin yapılacağı kafa yapısı için hazırlanır. Özellik haritalarının birleştirilmesi sırasında derin katmanlardan gelen genelleştirilmiş özellikler küçük boyutlu görsel tespiti için birleştirilirken sığ katmanlardan gelen detaylı özellikler büyük görsellerin tespiti için birleştirilir. Bu sayede içinden daha az detay yakalanabilen küçük nesnelerin genel özelliklerden; daha çok detayın yakalanabildiği büyük nesnelerin ise detaylı özelliklerden tespit edilebilmesi sağlamaktır.

Modelin baş yapısında ise boyundan gelen özellik haritaları 3 farklı boyutta sınıf olasılıkları yapılması amacıyla en son 2 boyutlu evrişim işleminden geçirilir. Evrişim işleminin sonucunda elde edilen matrislerde çevreleyici kutunun koordinatları, güven skoru ve her bir sınıf için olasılık skorları yer alır. Tahmin matrisinin kanal sayısı $c = (5 + n_{cls})\ x\ 3$ formülü kullanılarak hesaplanabilir. Bu formülde c sonuçtaki kanal sayısını, *5* sayısı çevreleyici kutu koordinatları ve güven skorunu, n_{cls} ifadesi ise problemdeki sınıf sayısını ve *3* sabiti ise her bir boyut ölçeğinde kullanılan çapa kutularının sayısını belirtir. Elde edilen sonuçlar daha sonra maksimum olmayanı baskılama (non-maximum supression) işleminden geçirilerek görsel üstünde yapılan tahminler tamamlanır. YOLOv5 algoritmasının sinir ağı mimarisi Şekil 3.5'te gösterilmiştir.



Şekil 3.5 YOLOv5 algoritmasının sinir ağı mimarisi

3.1.2.2. YOLOv5 algoritmasının hiperparametreleri

YOLOv5 algoritmasına dahil edilen 29 adet hiperparametre ile algoritmanın davranışının ve performansının optimize edilmesi sağlanır. Bu hiperparametreler model eğitiminden önce seçilir ve model eğitiminde alınan kararların, yapılan işlemlerin işleyişini yönetmek amacıyla kullanılır.

1. İlk Öğrenme Oranı (Initial Learning Rate): Modelin eğitimin başlangıcında kullandığı öğrenme oranıdır. Yüksek bir değerde olması hızlı bir öğrenmeye sebep olup aşırı uymaya (overfitting) sebep olurken düşük değerler de öğrenmenin yavaşlamasına ve lokal minimumlarda takılı kalmasına sebep olabilmektedir.
2. Final OneCycleLR Öğrenme Oranı (Final OnecycleLR Learning Rate): OneCycleLR öğrenme politikasında en son kullanılan öğrenme oranıdır. OneCycleLR eğitim sırasında öğrenme oranını düzenleyen bir öğrenme politikasıdır. Bu politika sayesinde eğitimin sonuna yaklaştıkça öğrenme oranı azaltılarak optimal noktaya daha küçük adımlara yakınsama yapılması sağlanır.
3. Momentum: Önceki ağırlık güncellemelerinin mevcut güncelleme üstündeki etkisini belirler. Yüksek bir momentum değeri modelin hızlı bir şekilde yakınsamasını sağlar ancak böyle bir senaryoda lokal minimum noktalarından kaçmak da zorlaşır.
4. Ağırlık Sönümü (Weight Decay): Sinir ağındaki ağırlıkların büyüklüğünü sınırlayarak aşırı uymaya (overfitting) engel olur. Yüksek ağırlık sönümü değeri genel yeteneklerin öğrenimi arttırırken aşırı uymaya engel olur. Fakat bu durum önemli özelliklerin kaçırılmasına sebep olabilir. Aşırı düşük ağır sönümü ise aşırı uymaya yol açabilmektedir.
5. Isınma Dönemi Sayısı (Warmup Epochs): Eğitimin başında momentumun ve öğrenme oranının kademi olarak arttırıldığı dönem sayısını ifade eder. Uzun ısınma dönemleri büyük ölçekli veri kümelerinde modelin daha kararlı bir şekilde öğrenmesini sağlar.
6. Isınma Momentumu (Warmup Momentum): Isınma dönemlerinde kullanılan momentum değeridir. Bu değer ısınma dönemleri boyunca arttırılarak optimum değerine ulaştırılır. Bu değerin yüksek olması

başlangıçta yapılan ağırlık güncellemelerini büyük ilerlemelerle yaparak modelin hızlıca lokal minimumlardan kaçırır. Ancak bu esnada eğitim verisindeki gürültülerden dolayı yapılan güncellemeler de büyük olacağı için modelin kararsız davranmasına sebep olabilir.

7. Isınma Yanlılık Öğrenme Oranı (Warmup Bias Learning Rate): Eğitimin başlangıcında yanlılık (bias) parametrelerinin öğrenme oranlarını belirleyen değerdir. Bu parametrenin yüksek olması eğitimin erken dönemlerinde yanlılık parametrelerinin hızlıca belirlenmesine neden olur. Bu durum modelin hızlıca öğrenilmesine yardımcı olabilir ancak aşırı uymaya da sebebiyet verebilir.
8. Kutu Kaybı Kazancı (Box Loss Gain): Sınırlayıcı kutu regresyon kaybının eğitim içindeki ağırlığını belirler. Bu parametreyi arttırmak eğitim sırasında sınırlayıcı kutu çizimlerinin duyarlılığının arttırılmasını sağlar. Ancak bu değer arttıkça diğer kayıp fonksiyonlarının ağırlığı azalır.
9. Sınıf Kaybı Kazancı (Class Loss Gain): Sınıf tahminlerinin eğitim içindeki ağırlığını belirler. Bu parametre arttıkça sınıf tahminlerindeki duyarlılık artar. Ancak diğer kayıp fonksiyonlarının ağırlığı azalır. Bu parametre çok sınıflı modellerin eğitiminde önemli bir yere sahiptir.
10. Pozitif Sınıf Ağırlığı (Class Positive Weights): Sınıf kaybı hesaplaması yapılırken pozitif örneklerin ağırlığını belirleyen parametredir. Değeri arttıkça modelin doğru sınıflandırma başarısı artar. Bu durum sınıflardaki etiketlenmiş örnek sayıları dengesiz olan veri kümelerinde tercih edilmektedir. Ancak düşük bir değer kullanıldığında modelin yanlış sınıflandırmalara karşı toleransı daha fazla olur.
11. Nesne Kaybı Kazancı (Object Loss Gain): Nesne varlığının kaybının eğitim içindeki ağırlığını belirler. Bu parametre arttıkça nesne tespitlerinin duyarlılığı artar. Ancak diğer kayıp fonksiyonlarının ağırlığı azalır.
12. Pozitif Nesne Ağırlığı (Object Positive Weights): Nesne varlığı kaybı hesaplaması yapılırken pozitif nesnelerin ağırlığını belirleyen parametredir. Değeri arttıkça modelin nesne tespit başarısı artar. Bu durum az nesnenin bulunduğu tespitlerde başarıyı arttırır. Değer düşük seçildiği durumlarda modelin arka plan görüntüsünü hatalı nesne olarak tespit ettiği durumlara karşı daha toleranslı davranılmasını sağlar.

13. IoU Eşik Değeri (IoU Threshold): Yapılan sınıf tahminlerinin pozitif olarak kabul görmesi için gerekli olan eşik değerini belirtir. Bu değeri yüksek tutmak eksik tahmin yapılmasına neden olabilecekken düşük olması durumunda ise hatalı tahminlerin sayısı artabilir.
14. Çapa Eşiği (Anchor Threshold): Bir çapa kutusunun hangi gerçek kutuyla eşleşeceğini belirleyen IoU değeridir. Bu eşik çapa kutusunun ne kadar iyi bir tahmin olduğuna karar veren eşik değeridir. Bu değer ne kadar yüksek seçilirse kabul edilen sınırlayıcı kutuların yerleşimleri daha isabetli hale gelir. Değerin düşük tutulması halinde daha büyük çapa kutularının kabul edilme miktarı artar. Bu durumda daha çok tahmin üretilebilir.
15. Çapa Kutusu Sayısı (Anchors): Modelin çıkış katmanında yapılacak her bir tahmin için kaç tane çapa kutusunun kullanılacağını belirten parametredir. Büyük seçilmesi halinde farklı en boy oranları ve büyüklükler için tahminler üretilebilmektedir. Bu durum farklı boyutlarda çok fazla sınıfın bulunduğu modellerde tahmin başarısını arttırmaktadır. Küçük seçilmesi halinde belirli en boy oranına ve büyüklüğe sahip sınıfların bulunduğu modellerin başarısını ve performansını arttırmaktadır. Bu değerin yanlışlıkla büyük seçilmesi modelin performansını düşürmektedir.
16. Odak Kaybı Gama Değeri (Focal Loss Gamma): Bu parametre odak kaybı fonksiyonuna ait bir değişkendir. Odak kaybı fonksiyonu sınıflandırma yapılırken eğitimin zor ve nadir örnekler üzerinde yoğunlaşmasını sağlayan bir metottur. Odak kaybı fonksiyonu eğitim sırasında zaten iyi tahminler üretilen sınıfların ağırlıklarını azaltıp zorlu örneklerin ağırlıklarını arttıracak şekilde çalışır. Bu fonksiyonun gama değişkeni arttıkça nadir ve zorlu örneklerde daha iyi bir eğitim gerçekleştirilir. Dengeli olarak dağıtılmış veri kümelerinde bu değer düşük tutularak tüm sınıfların benzer şekilde ağırlıklandırılması amaçlanır.
17. Renk Tonu (Hue): Bu parametre görüntünün renk tonunu ayarlar. Bu parametre arttıkça görüntüdeki renk tonları arasındaki çeşitlilik artar. Bu sayede farklı aydınlatma koşullarındaki nesneleri tespit edebilme yeteneği artar.
18. Renk Doygunluğu (Saturation): Görüntünün renk doygunluğunu ayarlar. Bu değerin arttırılması özellikle renkli nesnelerin tespitini kolaylaştırır.

19. Renk Değeri (Value): Bu parametre görüntünün parlaklığını ayarlar. Bu parametrenin arttırılması modelin farklı ışık seviyelerinde ve gölgelendirmelerde daha güçlü sonuç vermesini sağlar.
20. Derece (Degrees): Bu parametre görüntünün saat yönünde ya da tersinde rastgele seçilen bir açıda döndürülerek kullanılmasını sağlar. Bu sayede nesnelerin farklı yönelimlerinin öğrenilmesi amaçlanır. Ancak bu değerin çok fazla girilmesi halinde nesneler bağlamlarından kopabileceği için eğitimi olumsuz etkileyebilir.
21. Çevirme (Translate): Görüntünün dikeyde veya yatayda rastgele bir şekilde kaydırılmasını sağlar. Bu parametre sayesinde nesnelerin kısmen görünür parçaları da eğitime dahil edilir. Böylece görüntüden tamamen yer almayan nesnelerin tespiti kolaylaşır.
22. Ölçekleme (Scale): Görüntünün büyütülmesine veya küçültülmesine sebep olur. Bu sayede görüntünün farklı ölçeklerde tanınabilmesi sağlanır. Ancak aşırı ölçekleme halinde nesnelerin tanınabilirliği azalabilir.
23. Eğme (Shear): Görüntünün yatay veya dikey eksen eğilmesini sağlar. Eğimli olarak görünen nesnelerin tanınabilmesini kolaylaştırır. Ancak bu hiperparametre görüntünün yapısını bozabilir.
24. Perspektif (Perspective): Görüntüye perspektif dönüşümü uygular. Bu sayede modelin farklı perspektiflerdeki görünümlere uyum sağlaması sağlanır. Ancak aşırı uygulanması durumunda nesne şekli bozulabilir.
25. Dikey Çevirme Olasılığı (Flip Up Down): Görüntüyü dikey olarak çevirir. Bu sayede modelin dikey çevrilmiş görüntülerdeki nesneleri tanıma başarısı artar.
26. Yatay Çevirme Olasılığı (Flip Left Right): Görüntüyü yatay olarak çevirir. Bu sayede modelin yatay çevrilmiş görüntülerdeki nesneleri tanıma başarısı artar.
27. Mozaik Veri Arttırma Olasılığı (Mosaic): Dört farklı görüntüyü birleştirerek tek bir mozaik görüntü oluşturur. Bu sayede bitişik nesnelerin model tarafından tanınma başarısı artar ve zorlu arka planlarda daha başarılı sonuçlar alınır.
28. Görüntü Karıştırma Olasılığı (Mix up): İki görüntüyü birbirine karıştırarak yeni görüntü oluşturur. Bu sayede modelin birbirinin üstüne geçmiş

cisimleri tanıma başarısı artar. Model etiketlenmiş görüntülerdeki gürültüleri elemede daha başarılı hale gelir.

29. Segment Kopyalama Yapıştırma Olasılığı (Copy Paste): Bir görüntüden alınan segmentlerin görüntülerini başka görüntülere yapıştırır. Bu sayede görüntünün kısmi göründüğü durumlarda tanınması ve nesnenin başka bir nesne tarafından kısmen engellendiği durumlarda tanınması kolaylaşır.

Çizelge 3.1. YOLOv5 algoritmasının sinir ağı modelleri

Hiperparametre	Minimum Değer	Maksimum Değer
İlk Öğrenme Oranı	1,00E-05	1,00E-01
Final OneCycleLR Öğrenme Oranı	1,00E-02	1,00E+00
Momentum	6,00E-01	9,80E-01
Ağırlık Sönümü	0,00E+00	1,00E-03
Isınma Dönemi Sayısı	0,00E+00	5,00E+00
Isınma Momentumu	0,00E+00	9,50E-01
Isınma Yanlılık Öğrenme Oranı	0,00E+00	2,00E-01
Kutu Kaybı Kazancı	2,00E-02	2,00E-01
Sınıf Kaybı Kazancı	2,00E-01	4,00E+00
Pozitif Sınıf Ağırlığı	5,00E-01	2,00E+00
Nesne Kaybı Kazancı	2,00E-01	4,00E+00
Pozitif Nesne Ağırlığı	5,00E-01	2,00E+00
IoU Eşik Değeri	1,00E-01	7,00E-01
Çapa Eşiği	2,00E+00	8,00E+00
Çapa Kutusu Sayısı	2,00E+00	1,00E+01
Odak Kaybı Gama Değeri	0,00E+00	2,00E+00
Renk Tonu	0,00E+00	1,00E-01
Renk Doygunluğu	0,00E+00	9,00E-01
Renk Değeri	0,00E+00	9,00E-01
Derece	0,00E+00	4,50E+01
Çevirme	0,00E+00	9,00E-01
Ölçekleme	0,00E+00	9,00E-01
Eğme	0,00E+00	1,00E+01
Perspektif	0,00E+00	1,00E-03
Dikey Çevirme Olasılığı	0,00E+00	1,00E+00
Yatay Çevirme Olasılığı	0,00E+00	1,00E+00
Mozaik Veri Arttırma Olasılığı	0,00E+00	1,00E+00

Görüntü Karıştırma Olasılığı	0,00E+00	1,00E+00
Segment Kopyalama Yapıştırma Olasılığı	0,00E+00	1,00E+00

3.1.2.3. YOLOv5 algoritmasının metrikleri

YOLOv5 algoritması üretilen modelin performansını hesaplamak adına birtakım metrikler kullanmaktadır. Bunlardan yaygın olarak kullanılanlar aşağıda tanıtılmıştır.

1. Kesinlik (Precision): Pozitif olarak sınıflandırılan tespitlerin gerçekten pozitif olma oranıdır. Modelin yanlış pozitif bulma etkisini ölçer. Kesinlik metriğinin denklemi Denklem 3.1'de verilmiştir.
2. Duyarlılık (Recall): Pozitif olarak sınıflandırılan tespitlerin gerçekte pozitif olan tüm örneklere oranı. Kaçırılan pozitifleri ölçmeye yarar. Duyarlılık metriğinin denklemi Denklem 3.2'de verilmiştir.
3. F1 Skoru: Kesinlik ve duyarlılık arasındaki dengeyi gösteren metriktir. İki değerin harmonik ortalaması alınarak hesaplanır. F1 skorunun metriğinin denklemi Denklem 3.3'te verilmiştir.
4. Ortalama Kesinlik (Average Precision, AP): Kesinlik ve duyarlılık metriklerinin performansının bir arada değerlendirildiği metriktir. Sınıflar için performans belirlemesinde kullanılır. Kesinlik-duyarlılık grafiğinin altında kalan alan hesaplanarak bulunur.
5. Genel Ortalama Kesinlik (Mean Average Precision, mAP): Tüm sınıflar için hesaplanan AP değerlerinin ortalamasıdır. Tüm modelin performansını belirlemede kullanılabilir.

$$Kesinlik = \frac{Gerçek\ Pozitif}{Gerçek\ Pozitif + Yanlış\ Pozitif} \quad (3.1)$$

$$Duyarlılık = \frac{Gerçek\ Pozitif}{Gerçek\ Pozitif + Yanlış\ Negatif} \quad (3.2)$$

$$F1\ Skoru = \frac{2\ x\ Kesinlik\ x\ Duyarlılık}{Kesinlik + Duyarlılık} \quad (3.3)$$

3.2. Parçacık Sürü Optimizasyonu Algoritması

Parçacık sürü optimizasyonu (PSO) algoritması Eberhart ve Kennedy (1995) tarafından geliştirilmiş bir metasezgisel optimizasyon algoritmasıdır. PSO kuş sürülerinin

ve balık sürülerinin toplu hareketlerini taklit ederek karmaşık optimizasyon problemlerini çözmeye çalışır. Algoritma birçok parçacığın probleme ait arama uzayında hareket ederek en iyi çözüme ulaşmaya çalıştığı bir süreç işletir. PSO algoritması devre tasarımı, robotik (Abido, 2002), algoritma parametre seçimi (Shi ve Eberhart., 1998) vb. alanlarda etkili bir şekilde kullanılmaktadır.

3.2.1. Parçacık sürü optimizasyonu algoritmasının temel çalışma prensibi

PSO algoritması sürüdeki parçacıkların arama uzayında rastgele konumlarda ve hızlarda dağıtılmasıyla ilklendirilir. Her parçacık kendisinin bulduğu en iyi konumu bilirken aynı zamanda tüm sürü sürüye ait en iyi konumu bilir. Her iterasyonda parçacıklar, önceki iterasyondan gelen hız bilgisi, parçacığın bulduğu en iyi konum bilgisi ve sürüye ait en iyi konum bilgisini kullanarak hızlarını günceller. Güncellenen hız bilgisi kullanılarak da parçacığın yeni konumu hesaplanır. Eğer yeni konum parçacık için o ana kadarki en iyi konumundan daha iyi ise parçacık kendi en iyi konumunun bilgisini günceller. Ve şayet güncellenen bu yeni konum bütün sürü için de en iyi konum anlamına geliyorsa sürü için en iyi konum bilgisi de güncellenir. Algoritma bu adımları önceden belirlenmiş bir iterasyon sayısı ya da en iyi çözümün belirli bir süre boyunca güncellenememesi gibi bir kritere bağlı olarak tekrar eder. İterasyonlar sona erdiğinde sürüye ait en iyi konum bilgisi problemin çözümü olarak sunulur. PSO algoritmasının sözde kodu Şekil 3.6'da verilmiştir.

Parçacık Sürü Optimizasyonu (PSO)

Parçacık sayısını N, iterasyon sayısını T olarak belirle
Hedef fonksiyonu $f(x)$ olarak belirle
for her parçacık $i = 1$ to N **do**
    Parçacık i'nin pozisyonunu x_i ve hızını v_i rastgele başlat
    $pBest_i$'yi x_i olarak belirle
end for
$gBest$'i en iyi $pBest$ olarak belirle
for her iterasyon $t = 1$ to T **do**
    for her parçacık $i = 1$ to N **do**
        Parçacık i'nin hızını güncelle: $v_i \leftarrow w \cdot v_i + c_1 \cdot r_1 \cdot (pBest_i - x_i) + c_2 \cdot r_2 \cdot (gBest - x_i)$
        Parçacık i'nin pozisyonunu güncelle: $x_i \leftarrow x_i + v_i$
        if $f(x_i) < f(pBest_i)$ **then**
            $pBest_i \leftarrow x_i$
            if $f(pBest_i) < f(gBest)$ **then**
                $gBest \leftarrow pBest_i$
            end if
        end if
    end for
end for
return $gBest$

Şekil 3.6 PSO Algoritmasının sözde kodu

3.2.2. Parçacık sürü optimizasyonu algoritmasının parametreleri

PSO algoritmasında algoritmanın akışı sırasında alınan kararları etkileyen yedi adet parametre vardır.

1. Parçacık Sayısı: Algoritmanın kullanacağı parçacık sayısını belirtir. Eberhart ve Kennedy (1995) tarafından önerilen parçacık sayısı 20 ile 40 arasındadır. Bu seçimin işlem yükünü makul düzeyde tutacağı belirtilmiştir. Shi ve Eberhart (2001) yapılan çalışmada ise daha büyük parçacık sayısının bazı durumlarda daha iyi sonuç verebileceği gösterilmiştir.
2. Hız Kısıtlamaları: Hız kısıtlaması parametresi algoritmanın her iterasyonunda parçacık hızının güncellenmesi sırasında hızın aşırı miktarlarda güncellenmesini engellemektedir. Bu parametre algoritmanın hızlı yakınsamasını ve stabil çalışmasını sağlamaktadır.
3. Kişisel ve Global Güncelleme Katsayıları (c_1 ve c_2): Bu katsayılar hız güncellemesi sırasında kişisel ve global en iyi değerlerin güncellemedeki etkisini belirler. Eberhart ve Kennedy (1995) bu değerin 2 seçilmesini önermişlerdir.

4. Sabitlik Katsayısı (Intertia Weight): Hız güncellemesi sırasında mevcut hızın ne kadarının korunacağını belirten parametredir. Yüksek tutulması halinde sonuca yakınsama daha yavaş olacaktır. Shi ve Eberhart (1998) çalışmalarında bu sayının iterasyonlar ilerledikçe dinamik olarak azaltılması ile yakınsamanın hızlandırılabileceğini önermişlerdir.
5. Maksimum İterasyon Sayısı: Algoritmanın kaç iterasyon boyunca çalışacağını belirten parametredir. Eberhart ve Shi (2001) bu parametrenin çözülen probleme özgü olduğunu belirtmişlerdir ve deneme yanılma yoluyla bulunması gerektiğini söylemişlerdir.

3.3. Yapay Arı Kolonisi Optimizasyonu Algoritması

Yapay arı kolonisi (Artificial Bee Colony, ABC) algoritması Karaboğa tarafından (2005) geliştirilmiş bir metasezgisel optimizasyon algoritmasıdır. Algoritma bal arısı kolonilerindeki işçi arıların ve gözcü arıların davranışlarını taklit ederek karmaşık optimizasyon problemlerini çözmeye çalışır. ABC algoritmasının yapı tasarımı (Sönmez, 2011), görüntü işleme (Horng ve diğ., 2010), enerji sistemi optimizasyonunda (Oliva ve diğ., 2017) vb. alanlarda etkin olarak kullanılmaktadır.

3.3.1. Yapay arı kolonisi algoritmasının temel çalışma prensibi

ABC algoritması kolonideki işçi arıların arama uzayında rastgele konumlarda dağıtılmasıyla ilklendirilir. Her iterasyonda işçi arılar aşaması, gözcü arılar aşaması ve kaynaktan ayrılma aşaması adı verilen üç aşama işletilir. İşçi arılar aşamasında her bir işçi arı mevcut çözümün etrafında yeni çözüm adayı arar. Bu arama işlemini yaparken arı kendi kaynağı ile diğer bilinen kaynaklar arasından rastgele seçilen başka bir kaynak arasında bulunan rastgele seçilmiş bir noktayı belirler. Eğer yeni ürettiği çözüm mevcut çözümünden iyiyse arı yeni çözümü benimser. İzleyici arılar aşamasında işçi arıların yaptığı çözümler arasından daha iyi kaynakların daha yüksek skor aldığı olasılıkçı bir dağılım içinden seçilen kaynaklara izleyici arılar dağıtılır. Her izleyici arı gönderildiği kaynağın etrafında işçi arıların önceki aşamada kullandığı arama yöntemiyle bir kaynak belirler. Eğer yeni seçilen kaynak izleyici arının gönderildiği kaynaktan daha iyiyse izleyici arı yeni bulduğu kaynağı benimser. Kaynaktan ayrılma aşaması ise mevcut çözümün belirlenmiş bir süre boyunca iyileştirilememesi durumunda gerçekleşir. İşçi ve

izleyici arıların kaynak aramaları belirli bir iterasyon süresi boyunca gelişme göstermez ise arı kâşif arı olarak kaynağını terk eder ve tamamen rastgele seçilen yeni bir kaynağa yerleşir. Bu üç adım önceden belirlenmiş iterasyon sayısı kadar veya önceden belirlenmiş CPU çalışma süresi kadar tekrar edilir. Algoritma tamamlandığında arılar tarafından bulunan en iyi konum problemin çözümü olarak sunulur. ABC algoritmasının sözde kodu Şekil 3.7'de verilmiştir.

Yapay Arı Kolonisi (ABC) Algoritması

Problemin uygunluk fonksiyonunu tanımla $f(x)$
Arı sayısını N olarak belirle
İterasyon sayısını $MaxIterasyon$ olarak belirle
Her besin kaynağını x_i rastgele başlat ve değerlendir
for $iter = 1$ to $MaxIterasyon$ **do**
    for her işçi arı $i = 1$ to N **do**
        Yeni çözüm v_i üret: $v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$
        Uygunluk değerlerini karşılaştır: Eğer $f(v_i) > f(x_i)$ ise, $x_i = v_i$
    end for
    for her izleyici arı $i = 1$ to N **do**
        Çözüm seçim olasılığına göre x_i seç
        Yeni çözüm v_i üret: $v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{mj})$
        Uygunluk değerlerini karşılaştır: Eğer $f(v_i) > f(x_i)$ ise, $x_i = v_i$
    end for
    for her besin kaynağı $i = 1$ to N **do**
        if besin kaynağı belirli bir süre iyileştirilemiyorsa **then**
            Kaynaktan ayrıl ve yeni rastgele x_i üret: $x_i = RastgeleSecim(sınır_{min}, sınır_{max})$
        end if
    end for
    En iyi global çözümü güncelle
end for
return En iyi bulunan çözüm

Şekil 3.7 ABC Algoritmasının sözde kodu

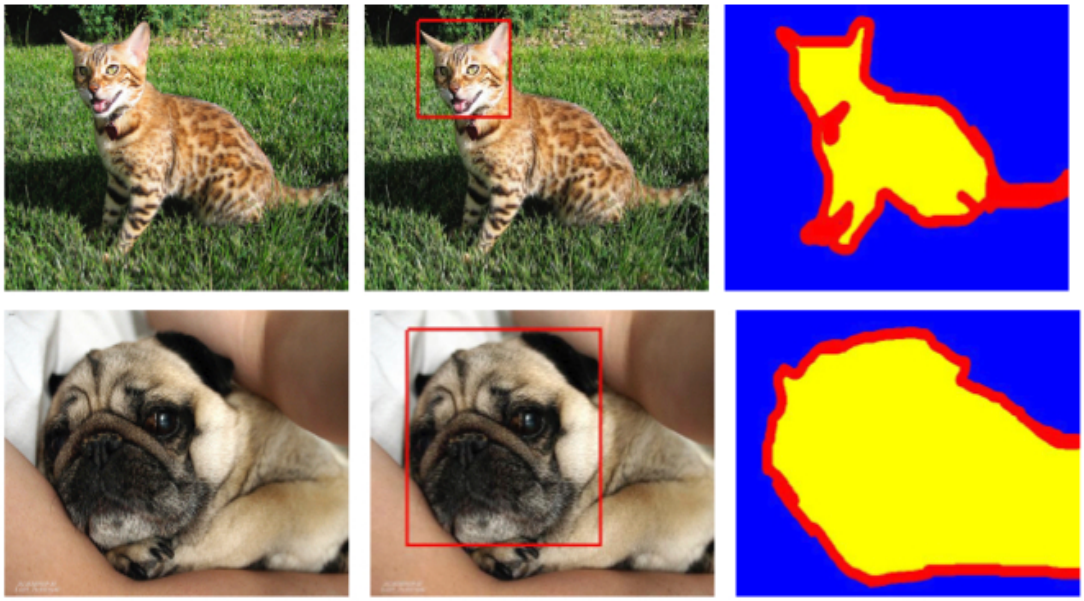
3.3.2. Yapay arı kolonisi algoritmasının parametreleri

ABC algoritmasında algoritmanın akışı sırasında alınan kararları etkileyen 3 adet parametre vardır.

1. Toplam Arı Sayısı: Kolonideki arıların sayısıdır. İşçi ve izleyici arıların sayısı toplam arı sayısının yarısı kadar olmaktadır.
2. Kaynaktan Ayrılma Limiti: Bir arının kaynaktan ayrılıp yeni bir kaynağa geçmesi için geçmesi gereken verimsiz geçen iterasyon sayısını belirtir.
3. İterasyon Sayısı: Algoritmanın kaç iterasyon boyunca koşulacağını ifade eden sayıdır.

3.4. Oxford-IIIT Evcil Hayvan Veri Kümesi

Oxford-IIIT evcil hayvan veri kümesi Parkhi ve diğ. (2012) tarafından oluşturulmuş, nesne tespiti, sınıflandırması ve görüntü işlemede kullanılan veri kümesidir. Bu veri kümesinde evcil kedi ve köpek görselleri bulunmaktadır. Veri kümesinde 25 köpek ırkı, 12 kedi ırkını temsil eden çeşitli ölçeklendirmelerde ve farklı ışık koşullarında çekilmiş 7349 görsel bulunmaktadır. Veri kümesinde her bir ırk için 200 civarında görsel bulunmaktadır. Veri kümesinde toplamda 2.371 kedi ve 4.978 köpek görseli bulunmaktadır. Veriler yapay zekâ çalışmalarında kullanılması amacıyla etiketlenmiş olarak sunulmaktadır. Şekil 3.8'de veri kümesine ait örnekler sunulmuştur.



Şekil 3.8 Oxford-IIIT Evcil Hayvan Veri Kümesinden örnek görseller

3.5. Sunulan Metot

Bu çalışmada YOLOv5 algoritmasının hiperparametrelerinin ABC ve PSO algoritmaları kullanılarak optimize edilmesi amaçlanmıştır. Bu amaçla Oxford-IIIT evcil hayvan veri kümesi kullanılmıştır. Hem evrişimsel sinir ağlarının eğitim maliyeti hem de sürü zekâsı algoritmalarının görece yüksek işlem gücü ihtiyacı göz önüne alınarak 37 evcil hayvan ırkının sınıflandırılması yerine kedi ve köpek sınıfları olmak üzere 2 sınıf

kullanılmıştır ve veri kümesinden veri eksiltilerek rastgele olarak 3.680 görsel seçilmiştir. Veri kümesindeki görseller 2.576 tanesi (%70) eğitim verisi, 736 tanesi (%20) doğrulama verisi, 368 tanesi de (%10) test verisi olacak şekilde gruplara ayrıldı. Veri kümesinde eksiltme yapılırken veri kümesinin özgün halindeki kedi ve köpek dağılımının korunması amacıyla 1.188 kedi ve 2.492 köpek görseli seçilmiştir.

ABC ve PSO algoritmalarının arama uzayı olarak YOLOv5 algoritmasının hiperparametreleri arasından seçilen 9 adet hiperparametre belirlenmiştir. Bu hiperparametreler ilk öğrenme oranı, final OneCycleLR öğrenme oranı, momentum, ağırlık sönümü, pozitif sınıf ağırlığı, pozitif nesne ağırlığı, IoU eşik değeri, çapa eşiği ve odak kaybı gama değeri olarak seçilmiştir. Hiperparametrelerin azaltılıp bu hiperparametrelerin seçilmesindeki amaç çözüm uzayının karmaşıklığını azaltmak ve model eğitimi sürecine harcanacak maliyeti azaltmaktır. Hiperparametre uzayının boyutları azaltılırken öğrenme odaklı, ağırlık ve momentum odaklı, kayıp fonksiyonları odaklı ve veri arttırma odaklı hiperparametrelerin seçilmesi göz önünde bulundurulmuştur. ABC ve PSO algoritmalarında arıların ve parçacıkların üreteceği çözüm değerleri bahsedilen hiperparametreler olarak belirlenmiştir ve amaç fonksiyonu olarak YOLOv5 algoritmasının eğitimi koşulmuştur. ABC algoritması 18 arılık bir koloniyle 50 iterasyon boyunca kâşif arıya dönüşme kriteri 80 deneme olacak şekilde koşulmuştur. PSO algoritması ise 18 parçacık ile 50 iterasyon boyunca kişisel güncelleme katsayısı $c_1 = 2{,}0$, global güncelleme katsayısı $c_2 = 2{,}0$, sabitlik katsayısı $w = 0{,}9$ olacak şekilde koşulmuştur. Optimizasyon algoritmaları için fonksiyon olarak belirlenen her bir model eğitimi 10 devir (epoch) sürecek, görsel boyutlarını 416x416 olarak kullanacak, görselleri RAM üzerinde önbellekleyecek ve YOLOv5s (küçük) modelini kullanacak şekilde tasarlanmıştır. Eğitimlerin yığın boyutları (batch size) bellek kullanımının maksimize edilmesi amacıyla YOLOv5 algoritması tarafından otomatik olarak seçilmiştir. Eğitimlerin hızlı ilerlemesi adına 8 adet GPU üzerinde dağıtık olarak çalışması sağlanmıştır. Tamamlanan her bir eğitimin sonunda öğrenilen mAP50 metriği optimizasyon algoritmaları için fonksiyonun ürettiği sonuç olarak kullanılmıştır. Bu şekilde optimizasyon algoritmalarının mAP50 metriğini optimize etmesi amaçlanmıştır.

ABC ve PSO optimizasyonları tamamlandıktan sonra testlerde kullanmak üzere iki farklı model eğitimi yapılmıştır. Bu model eğitimleri 500 devir (epoch) sürecek, görselleri boyutlarını 416x416 kullanacak, görselleri RAM üzerinde önbellekleyecek ve YOLOv5s (küçük) modelini kullanacak şekilde tasarlanmıştır. Eğitimlerin yığın boyutları (batch size) bellek kullanımının maksimize edilmesi amacıyla YOLOv5 algoritması

tarafından otomatik olarak seçilmiştir. Eğitimlerin hızlı ilerlemesi adına 8 adet GPU üzerinde dağıtık olarak çalışması sağlanmıştır. ABC ve PSO tarafından optimize edilmiş hiperparametrelerle eğitilen modeller daha sonra test verileri üstünde kullanılmıştır. Bunlara ek olarak optimize edilen hiperparametrelerle üretilen modellerin ön tanımlı hiperparametre kıyasla başarısının anlaşılması amacıyla YOLOv5 algoritmasında hazır sunulan Objects365 hiperparametre kümesi kullanılarak optimizasyonda elde edilen hiperparametrelerle aynı koşullar altında bir model eğitimi daha yapılmıştır. Optimize edilmeyen bu model de test verileri üstünde denenmiştir ve test sonuçları optimize edilen modellerin sonuçları ile test optimize edilen modeller ile kıyaslanmıştır. Deney düzenleri yukarıda bahsedilen bu işlemler Çizelge 3.2'de özellikleri verilen sistem aracılığı ile koşulmuştur.

Çizelge 3.2. Deneyler için kullanılan sistem özellikleri

Bileşen Adı	Kullanılan
İşlemci	AMD Epyc 7402 24 Çekirdek 3.35 GHz
Ekran Kartı	8 Adet NVIDIA RTX 4090
VRAM	192 GB
RAM	256 GB
İşletim Sistemi	Ubuntu 22.04 LTS
CUDA Versiyonu	12.2

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu tez çalışmasında yapılan deneyin sonuçları ABC ve PSO algoritmaları ile yapılan hiperparametre optimizasyonları, optimizasyon aracılığı ile elde edilen hiperparametreler ve ön tanımlı hiperparametreler ile yapılan model eğitimleri ve eğitilen modeller kullanılarak yapılan testler için ayrı alt başlıklarda incelenmiştir.

4.1. Hiperparametre Optimizasyon Aşaması

Bu bölümde ABC ve PSO algoritmaları ile yapılan hiperparametre optimizasyonunun sonuçları her bir optimizasyon yöntemi için kendi alt başlığında sunulmuştur.

4.1.1. ABC ile hiperparametre optimizasyon sonuçları

ABC algoritması ile yapılan hiperparametre optimizasyonu bir kez çalıştırılmış olup optimizasyon sırasında arıların faaliyeti sonucunda 941 kez model eğitimi yapılmıştır. Optimizasyon işlemi materyal ve metot bölümünde tanıtılan sistem üstünde 35 saat 20 dakika boyunca çalışmıştır. Optimizasyon sonucunda aşağıdaki hiperparametreler elde edilmiştir. 10 devirden oluşan 941 eğitim içinde en fazla 0,401 mAP50 skoru elde edildiği görüldü. İşlem sonucunda elde edilen hiperparametreler Çizelge 4.1'de verilmiştir.

Çizelge 4.1. ABC ile optimize edilmiş hiperparametreler

Hiperparametre	Değer
İlk Öğrenme Oranı	7,44E-02
Final OneCycleLR Öğrenme Oranı	7,25E-01
Momentum	7,96E-01
Ağırlık Sönümü	9,58E-04
Pozitif Sınıf Ağırlığı	6,97E-01
Pozitif Nesne Ağırlığı	1,68E+00
IoU Eşik Değeri	6,24E-01
Çapa Eşiği	5,72E+00
Odak Kaybı Gama Değeri	0,00E+00

4.1.2. PSO ile hiperparametre optimizasyon sonuçları

PSO algoritması ile yapılan hiperparametre optimizasyonu bir kez çalıştırılmış olup optimizasyon sırasında parçacıkların faaliyeti sonucunda 900 kez model eğitimi yapılmıştır. Optimizasyon işlemi deneyde kullanılan sistem üstünde 25 saat 30 dakika boyunca çalışmıştır. Optimizasyon sonucunda aşağıdaki hiperparametreler elde edilmiştir. 10 devirlik 900 eğitim içinde en fazla 0,206 mAP50 skoru elde edildiği görüldü. İşlem sonucunda elde edilen hiperparametreler Çizelge 4.2'de verilmiştir.

Çizelge 4.2. PSO ile optimize edilmiş hiperparametreler

Hiperparametre	Değer
İlk Öğrenme Oranı	9,50E-02
Final OneCycleLR Öğrenme Oranı	7,98E-01
Momentum	6,62E-01
Ağırlık Sönümü	9,12E-04
Pozitif Sınıf Ağırlığı	1,08E+00
Pozitif Nesne Ağırlığı	1,99E+00
IoU Eşik Değeri	3,22E-01
Çapa Eşiği	6,38E+00
Odak Kaybı Gama Değeri	1,69E+00

4.1.3. ABC ve PSO hiperparametre sonuçlarının kıyaslaması

ABC ve PSO algoritmalarının hiperparametreler için buldukları sonuçlar aşağıda her hiperparametre için ayrı ayrı ele alınmıştır.

1. İlk Öğrenme Oranı: PSO algoritmasında (0,09) ABC'ye göre (0,07) daha yüksek bulunan ilk öğrenme oranı ile daha hızlı ve daha agresif bir öğrenme amaçlandığı görülmüştür. Bu durum PSO algoritmasının hiperparametrelerinin aşırı uymaya sebep olmasına veya yakınsamakta zorlanmasına neden olabilmektedir. Ayrıca ABC algoritmasının tamamlanma süresinin PSO'ya göre %38 daha uzun olmasında bu parametredeki farklılıktan kaynaklanabileceği görülmüştür.
2. Final OneCycleLR Öğrenme Oranı: ABC ve PSO algoritmaları sırasıyla 0,72 ve 0,79 değerlerini yakalaması her iki algoritmanın da son öğrenme oranlarını benzer oranlarda düşürmesine sebep olmuştur.

3. Momentum: ABC algoritmasının (0,79) PSO'ya göre (0,66) daha yüksek momentuma sahip olduğu görülmüştür. Bu sayede ABC algoritmasından elde edilen hiperparametrelerle eğitilen modelin PSO'daki hiperparametrelerle eğitilen modele göre geçmiş devirlerde öğrenilen özellikleri daha çok aktarabilmesi sağlanmıştır.
4. Ağırlık Sönümü: Her iki algoritmanın da benzer değerler yakaladığı görülmüştür. İki algoritma da benzer oranlarda genelleme yapılması gerektiğine yönelik sonuç vermiştir.
5. Pozitif Sınıf Ağırlığı: PSO algoritması (1,07), ABC'ye göre (0,69) daha fazla pozitif sınıf ağırlığı bulmuştur. PSO algoritmasının hiperparametreleri ile eğitilen modelin hatalı sınıflandırmalara karşı ABC'ye göre daha az tolerans tanımıştır.
6. Pozitif Nesne Ağırlığı: PSO algoritması (1,99), ABC'ye göre (1,68) daha fazla pozitif nesne ağırlığı belirlemiştir. Bu durum PSO'nun nesne tespiti hatalarına karşı ABC'ye göre daha az toleranslı olmasına neden olmuştur.
7. IoU Eşik Değeri: ABC algoritmasının (0,62) PSO algoritmasına göre (0,32) daha yüksek IoU eşiği belirlediği görülmüştür. Bu durum ABC algoritmasının parametreleriyle yapılan model eğitiminde daha kesin skorlara sahip sonuçların doğru olarak kabul görmesine sebep olmuştur.
8. Çapa Eşiği: PSO algoritması (6,37), ABC'ye göre (5,72) daha yüksek bir çapa eşiği belirlemiştir. Bu durum PSO'nun çok keşişim sağlayan görece daha küçük çapa kutularındaki sonuçları kabul etmesine sebep olmuştur. ABC algoritmasındaki parametre değeri ise daha büyük çapa kutularındaki değerleri daha sık doğru sonuç olarak kabul edebilmektedir.
9. Odak Kaybı Gama Değeri: ABC algoritması (0,0) odak kaybı gaması kullanmazken PSO algoritmasının sonuçlarına göre (1,69) önemli miktarlarda odak kaybı gaması kullanılmıştır. Bu durum PSO algoritmasındaki hiperparametrelerin zor örnekler üstünde daha fazla odaklanmasını sağlamıştır.

Bu değerler kapsamında PSO ile elde edilen hiperparametrelerin daha zor sınıflandırma ve tespit problemlerinin altından kalkabileceğini ve hızlı ama aşırı öğrenmeye açık bir eğitim yapacağını göstermektedir. ABC ile elde edilen hiperparametrelerin ise daha dengeli ve kararlı bir eğitim süreciyle kolayca genelleştirilebilen, yanlış pozitif miktarı düşük isabetli modeller üretmesi beklenmektedir. Eğitim sırasında elde edilen en iyi mAP50 değerleri kıyaslandığında

ABC algoritması ile üretilen hiperparametrelerin PSO ile üretilen hiperparametrelere göre kullanılan eğitim verileri üstünde daha isabetli sonuçlar ürettiği görülmüştür.

4.2. Model Eğitimi Aşaması

ABC ve PSO algoritmaları ile optimize edilen hiperparametreler kullanılarak iki farklı model eğitimi yapılmıştır. Model eğitimleri her bir hiperparametre kümesi için 500 devir uzunluğunda sürmüştür. Model eğitimlerinin sonuçları optimize edildikleri algoritmaya göre alt başlıklarında açıklanmıştır.

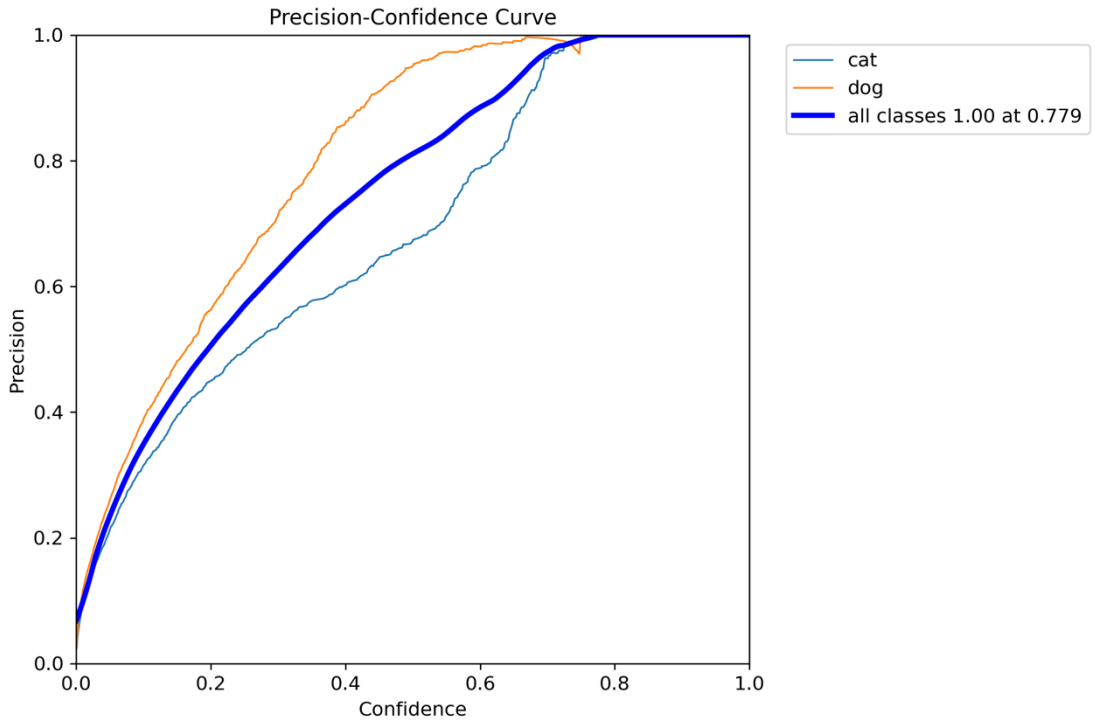
4.2.1 ABC ile optimize edilen hiperparametreler ile eğitim sonuçları

ABC ile yapılan eğitim sürecinde Çizelge 4.3 ile verilen sonuçlar elde edilmiştir.

Çizelge 4.3. ABC hiperparametreleri ile yapılan eğitimin sonuçları

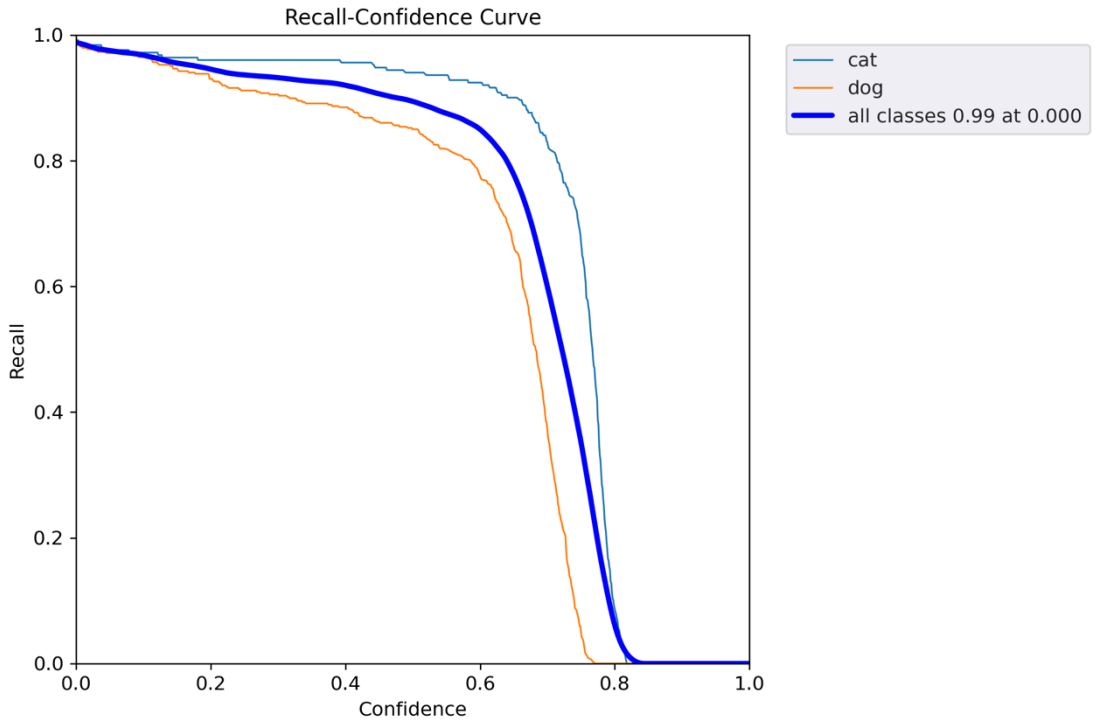
Sınıf	Örnek Sayısı	Kesinlik	Duyarlılık	F1 Skoru	mAP50
Tüm Sınıflar	738	0,881	0,86	0,86	0,936
Kedi	251	0,782	0,924	0,84	0,944
Köpek	487	0,98	0,795	0,87	0,928

Modelin eğitim sırasında doğrulama verileri üstünde tüm sınıflar için %88,1 oranında kesinliğe sahip olduğu görülmüştür. Kesinlik oranları kedi sınıfı için %78,2 iken köpek sınıfı için %98 olarak görülmüştür. Modelin doğrulama verilerindeki köpek sınıfı için ürettiği tahminlerde nadiren hatalı tahmin yaptığı görülmüştür. Sınıflar arasındaki kesinlik farkının kedi ve köpek sınıfları arasındaki örnek dengesinin daha çok köpek örneği bulunacak şekilde bozuk olmasından dolayı olduğu düşünülmektedir. Bu duruma ABC ile elde edilen hiperparametrelerin odak kaybı uygulamamasının yol açtığı düşünülmektedir. Şekil 4.1'de verilen kesinlik-güven aralığı grafiğinde modelin güven aralığı arttıkça kesinliğinin de arttığı görülmektedir. Grafikte de sunulduğu üzere modelin köpek sınıfı için kedi sınıfına göre daha yüksek kesinlikli sonuçlar verdiği görülmektedir.



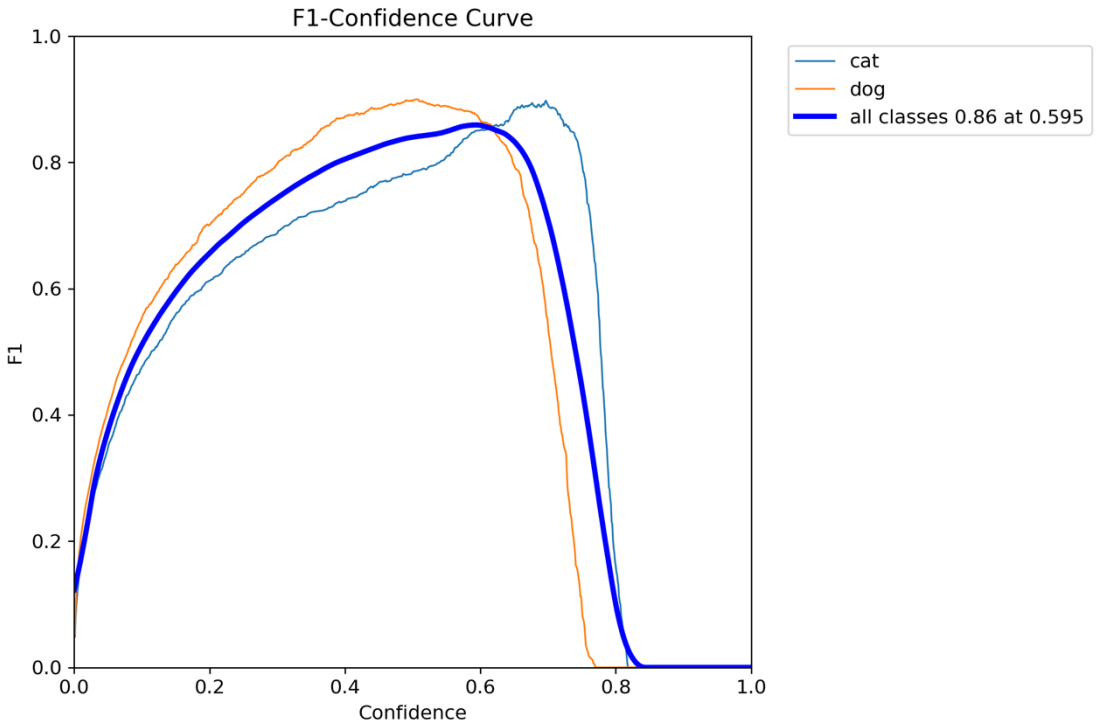
Şekil 4.1 ABC hiperparametreleriyle yapılan model eğitiminin Kesinlik-Güven Aralığı grafiği

Eğitilen model duyarlılık değeri tüm sınıflar için %86 iken bu değer kedi sınıfı için %92,4 ve köpek sınıfı için ise %79,5 olarak bulunmuştur. Bu durum kediler için yapılan tahminlerde gerçek kedi örneklerinin nadiren gözden kaçırıldığını ve köpekler için ise gözden kaçan örneklerin kedilere göre daha sık olduğunu göstermektedir. Şekil 4.2'de verilen duyarlılık-güven aralığı grafiğinde modelin %70'lik güven aralığına kadar tatmin edici miktarda duyarlılık sağlayabildiği görülmektedir. Aynı zamanda kedi sınıfının köpek sınıfına göre hem daha yüksek hem de daha geniş güven aralığında duyarlılık sağladığı görülebilmektedir.



Şekil 4.2 ABC hiperparametreleriyle yapılan model eğitiminin Duyarlılık-Güven Aralığı grafiği

Kedi ve köpek sonuçlarındaki kesinlik ve duyarlılık sonuçları bir arada değerlendirmeye alınacak olursa modelin kedi örneklerinde köpek grubuna göre daha sık yanlış pozitifler ürettiğini ancak gerçek kedi görsellerini ise nadiren gözden kaçırdığı görülmüştür. Köpek örnekleri ise model tarafından gözden kaçırılmaya daha müsait iken yapılan köpek tahminlerinin nadiren yanlış olduğu görülmüştür. Şekil 4.3'te verilen F1 skoru-güven aralığı grafiği incelendiğinde modelin %59,5'lik güven aralığında 0,86 F1 skoru verdiği görülmektedir. Modelin verdiği F1 skoru sınıfların temsili açısından dengesizlik olan bu veri kümesinde başarılı sayılabilecek bir konumdadır.



Şekil 4.3 ABC hiperparametreleriyle yapılan model eğitiminin F1 Skoru-Güven Aralığı grafiği

Sınıflar arasındaki kesinlik ve duyarlılık değerlerinin birbirlerinin zıttı trendler içinde olmasının nedeni ise optimizasyon ile elde edilen hiperparametrelerin farklı nitelikte, farklı nicelikte, farklı görsel koşul çeşitliliğinde olan sınıfların bir araya geldiklerinde olabilecek en uyumlu noktaya yaklaştırılırken yapılan bir kazanım takası ile açıklanabilmektedir. Bu durum YOLO algoritmasında hiperparametre optimizasyonu yapılırken kullanılan verinin de optimizasyon sürecinde belirlenen kazanımların ve kayıpların bir sebebi olduğuna dair izlenim vermiştir.

Son olarak ise eğitilen modelin tüm sınıflar için %93,6, kedi sınıfı için %94,4 ve köpek sınıfı için %92,8 oranında mAP50 skoru verdiği görülmüştür. Bütün sınıflar için %90 üstünde alınan mAP50 skorları modelin güvenilir sonuçlar ürettiğini göstermektedir.

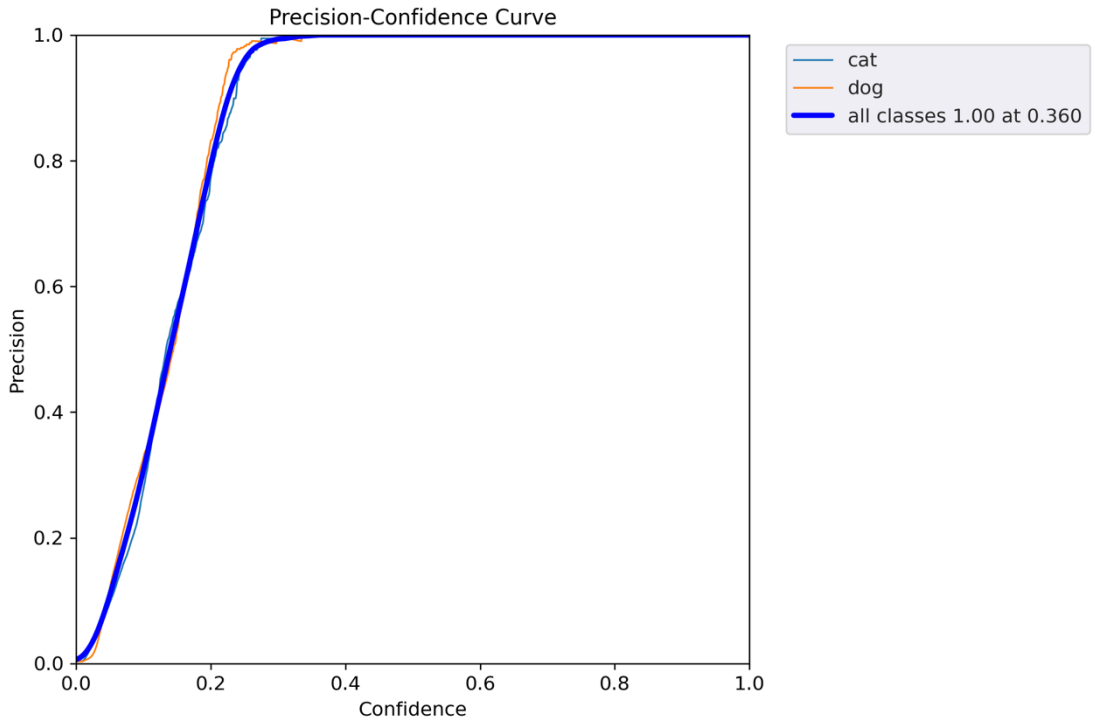
4.2.2 PSO ile optimize edilen hiperparametreler ile eğitim sonuçları

PSO ile yapılan eğitim sürecinde Çizelge 4.4 ile verilen sonuçlar elde edilmiştir.

Çizelge 4.4. PSO hiperparametreleri ile yapılan eğitimin sonuçları

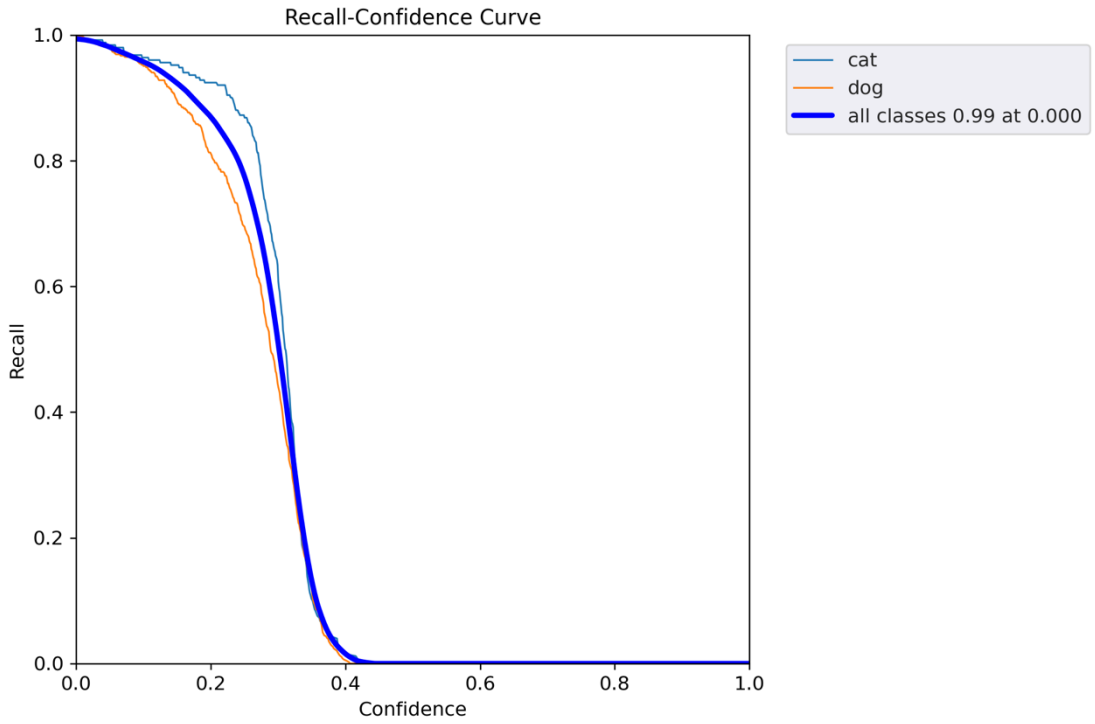
Sınıf	Örnek Sayısı	Kesinlik	Duyarlılık	F1 Skoru	mAP50
Tüm Sınıflar	738	0,92	0,826	0,86	0,917
Kedi	251	0,88	0,9	0,88	0,943
Köpek	487	0,961	0,751	0,84	0,892

Modelin eğitim sırasında doğrulama verileri üstünde tüm sınıflar için %92,1 oranında kesinliğe sahip olduğu görülmüştür. Kesinlik oranları kedi sınıfı için %88 iken köpek sınıfı için %96,1 olarak görülmüştür. Modelin doğrulama verilerindeki köpek tahminlerinin ABC ile optimize edilen modelde olduğu gibi çok hatalı çıkarım üretilmeden yapıldığı görülmüştür. Sınıflar arasındaki kesinlik farkının ABC hiperparametreleri ile eğitilen modelde olduğu gibi köpek sınıfının lehinde olmasına rağmen PSO algoritmasının sınıflar arasındaki kesinlik farkını daha küçük tuttuğu görülmüştür. Bu durumun ise PSO ile elde edilen hiperparametrelerde odak kaybı uygulamasının seçilmesi sayesinde olduğu düşünülmektedir. Şekil 4.4'te verilen kesinlik-güven aralığı grafiğinde modelin güven aralığı arttıkça kesinliğinin de arttığı görülmektedir. Grafikte de sunulduğu üzere modelin köpek sınıfı için kedi sınıfına göre nispeten daha yüksek kesinlikli sonuçlar verdiği görülmektedir.



Şekil 4.4 PSO hiperparametreleriyle yapılan model eğitiminin Kesinlik-Güven Aralığı grafiği

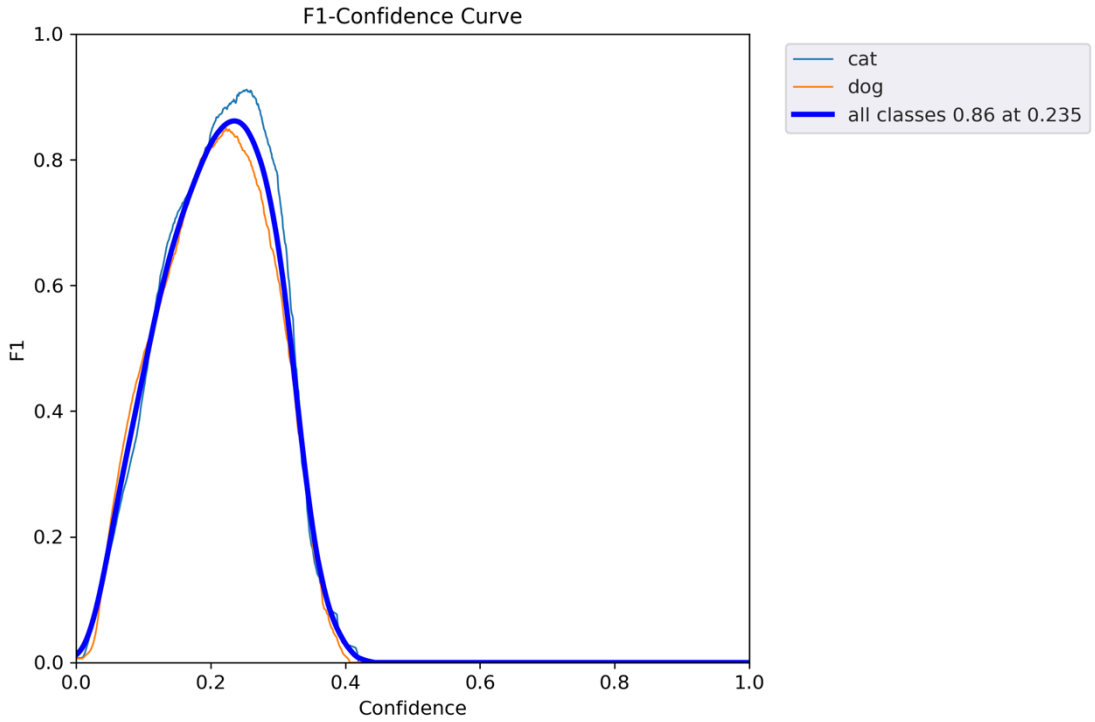
Eğitilen model duyarlılık değeri tüm sınıflar için %82,6 iken bu değer kedi sınıfı için %90 ve köpek sınıfı için ise %75,1 olarak bulunmuştur. Bu durum ABC hiperparametreleri ile eğitilen modele benzer bir şekilde kediler için yapılan tahminlerde gerçekten pozitif olan durumların nadiren gözden kaçırıldığını ve köpekler için ise gözden kaçan örneklerin kedilere göre daha sık olduğunu göstermektedir. Şekil 4.5'te verilen duyarlılık-güven aralığı grafiğinde modelin %30'luk güven aralığına kadar tatmin edici miktarda duyarlılık sağlayabildiği görülmektedir. Aynı zamanda kedi sınıfının köpek sınıfına göre hem daha yüksek hem de daha geniş güven aralığında duyarlılık sağladığı görülebilmektedir.



Şekil 4.5 PSO hiperparametreleriyle yapılan model eğitiminin Duyarlılık-Güven Aralığı grafiği

Kedi ve köpek sonuçlarındaki kesinlik ve duyarlılık sonuçları bir arada değerlendirmeye alınacak olursa modelin kedi tahminlerinde köpek tahminlerine göre daha sık yanlış pozitifler ürettiğini ancak gerçek kedi görsellerini ise nadiren gözden kaçırdığı görülmüştür. Köpek örnekleri ise model tarafından gözden kaçırılmaya daha müsait iken yapılan köpek tahminlerinin nadiren yanlış olduğu görülmüştür. Şekil 4.6'da verilen F1 skoru-güven aralığı grafiği incelendiğinde modelin %23,5'lik güven aralığında 0,86 F1 skoru verdiği görülmektedir. Modelin verdiği F1 skoru ABC ile optimize edilen

hiperparametreler ile eğitilen modele kıyasla daha kısıtlı bir güven aralığı içinde benzer başarıda sonuç vermektedir.



Şekil 4.6 PSO hiperparametreleriyle yapılan model eğitiminin F1 Skoru-Güven Aralığı grafiği

Son olarak ise eğitilen modelin tüm sınıflar için %91.7, kedi sınıfı için %94,3 ve köpek sınıfı için %89,2 oranında mAP50 skoru verdiği görülmüştür. Bütün sınıflar için %90'a yakın olan mAP50 skorları modelin güvenilir sonuçlar ürettiğini göstermektedir.

4.2.3 Ön tanımlı hiperparametreler ile eğitim sonuçları

Ön tanımlı hiperparametreler ile yapılan eğitim sürecinde Çizelge 4.5 ile verilen sonuçlar elde edilmiştir.

Çizelge 4.5. Ön tanımlı hiperparametreler ile yapılan eğitimin sonuçları

Sınıf	Örnek Sayısı	Kesinlik	Duyarlılık	F1 Skoru	mAP50
Tüm Sınıflar	738	0,447	0,768	0,56	0,566
Kedi	251	0,341	0,633	0,44	0,398

Köpek	487	0,553	0,903	0,68	0,734

Modelin eğitiminin tüm sınıflar için %44,7 oranında kesinliğe sahip olduğu görülmüştür. Kesinlik oranları kedi sınıfı için %34,1 iken köpek sınıfı için %55,3 olarak görülmüştür. Modelin duyarlılık başarımının tüm sınıflar için %76,8 olduğu görülmüştür. Duyarlılığın kedi ve köpek sınıfları için %63,3 ve %90,3 oranında olduğu görülmüştür. Modelin F1 skorunun tüm sınıflar, kediler ve köpekler için sırasıyla %56, %44 ve %68 olduğu görülmüştür. Son olarak ise modelin tüm sınıflar için %56,6 oranında mAP skoru verdiği görülmüştür. Kedi ve köpek sınıfları için ise %39,8 ve %73,4 oranında mAP50 skorları elde edilmiştir. F1 skorunun düşük olması modelin kesinliği ve duyarlılığı arasındaki uyumun zayıf olduğunu gösterirken mAP50 skorunun optimizasyon ile üretilen modellere göre zayıf olduğu görülmüştür.

4.2.4 Model eğitimlerinin karşılaştırması

Yapılan model eğitimlerinde optimizasyon aracılığıyla elde edilen modeller ile ön tanımlı hiperparametreler ile eğitilen modelin sonuçları karşılaştırıldığında optimizasyon uygulanan modellerin özellikle kesinlik açısından diğer modelden başarılı sonuçlar verdikleri görülmüştür. Bu durum bu veri kümesi özelinde yapılan optimizasyonun eğitilen modelin kesinliğinde iyileştirmeler yaptığını göstermiştir. Ön tanımlı hiperparametreler ile eğitilen modelin zayıf kesinlik sonucu modelin F1 skoru ve mAP50 skoru açısından diğer modellere kıyasla zayıf sonuçlar üretmesine neden olmuştur.

Optimizasyon ile elde edilen modeller kıyaslandığında ise ABC ile optimize edilen modelin PSO ile optimize edilen modele göre eğitim verileri üstünde görece daha iyi sonuçlar verdiği görülmüştür. ABC ile optimize edilen modelin kedi ve köpek sınıflarında başarıları arasındaki değişkenlik PSO ile optimize edilen modele göre daha fazla bulunmuştur. Bu durum PSO modelinin eğitim sırasında az temsil edilen ve zor olarak kabul edilen verileri ağırlıklandırmasına ve genele yönelik bir eğitimden çok nadir özellikleri öğrenmesine bağlanabilir. Bundan ötürü PSO modeli eğitim verileri içinde daha az bulunan kedi örneklerinde köpek sınıfı ile daha uyumlu kesinlik ve duyarlılık skorları vermiştir. ABC modeli ise kedi örneklerinde PSO modeline göre daha yüksek bir duyarlılık vermesine karşın kesinlik açısından PSO modeline görece zayıf kalmıştır. ABC modelinin PSO modeline göre daha genele odaklanan eğitimi sonucunda az temsil edilen kedi verilerinin detaylarının eğitimindeki eksiklikler ABC modelinin kedileri tespit

ederken doğru olmayan kedi tespitleri de yapmasına yol açmıştır. Bir diğer yandan veri kümesinde kedilere göre daha çok temsil edilen köpek sınıfında ise ABC modeli hem kesinlik hem de duyarlılık açısından PSO modeline göre görece daha başarılı olmuştur. ABC modelinin eğitim verilerine eşitlikçi yaklaşımı yeterli miktarda temsil edilen sınıflar için olumlu etki etmiştir. Ayrıca PSO hiperparametrelerinde ABC hiperparametrelerine kıyasla daha yüksek seçilen pozitif sınıf ağırlığı ve pozitif nesne ağırlığı hiperparametreleri modelin odak kaybı gaması uygulamasından dolayı nadir ve zor örneklere hem çok odaklanmasına hem de zor örnekler üstünde uğraşırken yapılan hataları ağır cezalandırmasına sebep olmuştur. PSO hiperparametrelerinin bunlara ek olarak yüksek öğrenme oranı seçimi de modelin daha hızlı ve agresif bir şekilde eğitilmesine sebep olduğu için genel özelliklerin daha az öğrenilmesine, nadir örneklerin öğrenimi sırasındaki hatalardan dolayı yaygın örneklerdeki kazanımların ağır cezalar kullanılarak kaybedilmesine ve öğrenmesi eğrisi üstünde büyük sıçramalar yapıp optimum noktaların ıskalanmasına sebep olmuştur.

4.3. Eğitilen Modeller ile Test Aşaması

ABC ve PSO algoritmaları ile optimize edilen hiperparametreler ile eğitilen modeller ile test verilerinde kedi ve köpek sınıfları için tespitler yapılmıştır. Testlerin sonuçları eğitim hiperparametrelerinin optimize edildikleri algoritmaya göre alt başlıklarında açıklanmıştır.

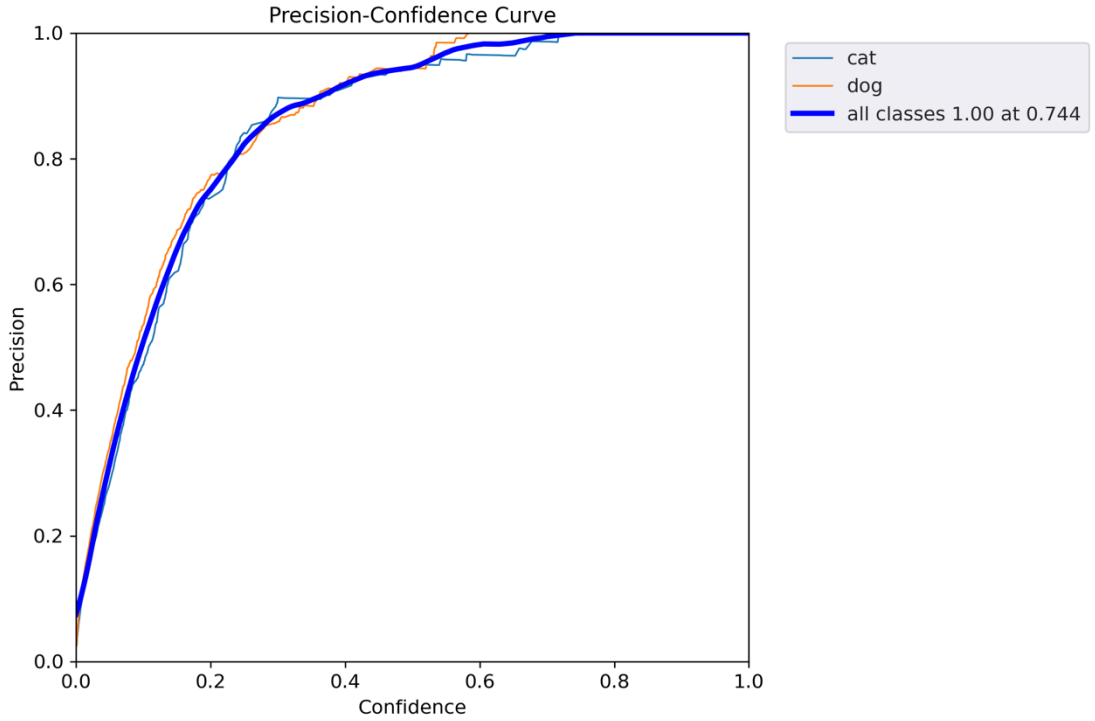
4.3.1 ABC ile optimize edilen modelin test sonuçları

ABC modeli ile yapılan test sürecinde Çizelge 4.6 ile verilen sonuçlar elde edilmiştir.

Çizelge 4.6. ABC modeli ile yapılan testin sonuçları

Sınıf	Örnek Sayısı	Kesinlik	Duyarlılık	F1 Skoru	mAP50
Tüm Sınıflar	368	0,883	0,771	0,82	0,817
Kedi	125	0,896	0,829	0,86	0,851
Köpek	243	0,87	0,713	0,78	0,784
Tahmin Hızı	4,6 milisaniye				

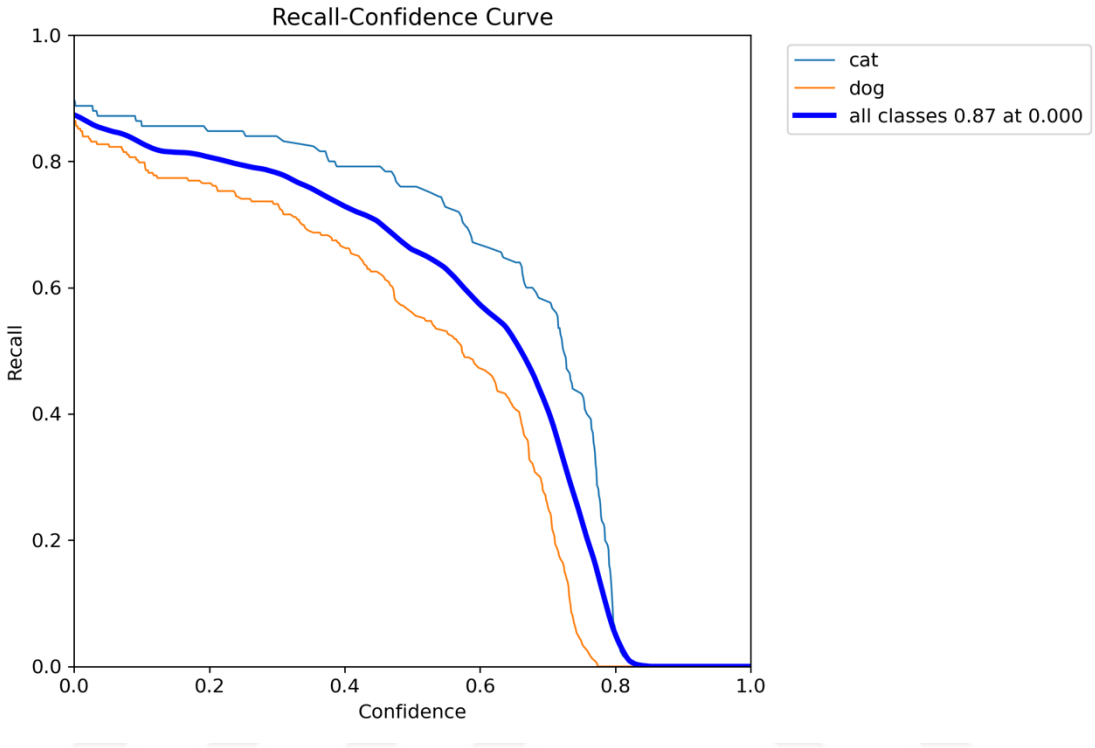
Modelin test verileri üstünde tüm sınıflar için %88,3 oranında kesinliğe sahip olduğu görülmüştür. Kesinlik oranları kedi sınıfı için %89,6 iken köpek sınıfı için %87 olarak görülmüştür. Modelin test verilerindeki köpek örneklerini eğitim aşamasındaki doğrulama verilerine göre daha az kesinlikle tahmin ettiği görülmüştür. Sınıflar arasındaki kesinlik farkının model eğitimindeki sonuca göre oldukça kapandığı görülmüştür. Bu durum bize model eğitiminde ABC hiperparametrelerinin daha iyi bir genelleştirme sağladığını ve gerçek hayat verileri üstünde modelin çalıştırıldığında çoğu durumda benzer kararlılıkta tahmin üretebildiğini göstermektedir. Şekil 4.7'de verilen kesinlik-güven aralığı grafiğinde modelin güven aralığı arttıkça kesinliğinin de arttığı görülmektedir. Grafikte de sunulduğu üzere modelin köpek sınıfı için kedi sınıfına göre ihmal edilebilir seviyede daha yüksek kesinlikli sonuçlar verdiği görülmektedir. Grafikte de modelin eğitim sonuçlarına göre daha iyi bir genelleştirme yaptığı ve sınıflar arasındaki farklılıkları kapatabildiği görülmektedir.



Şekil 4.7 ABC modeliyle yapılan testin Kesinlik-Güven Aralığı grafiği

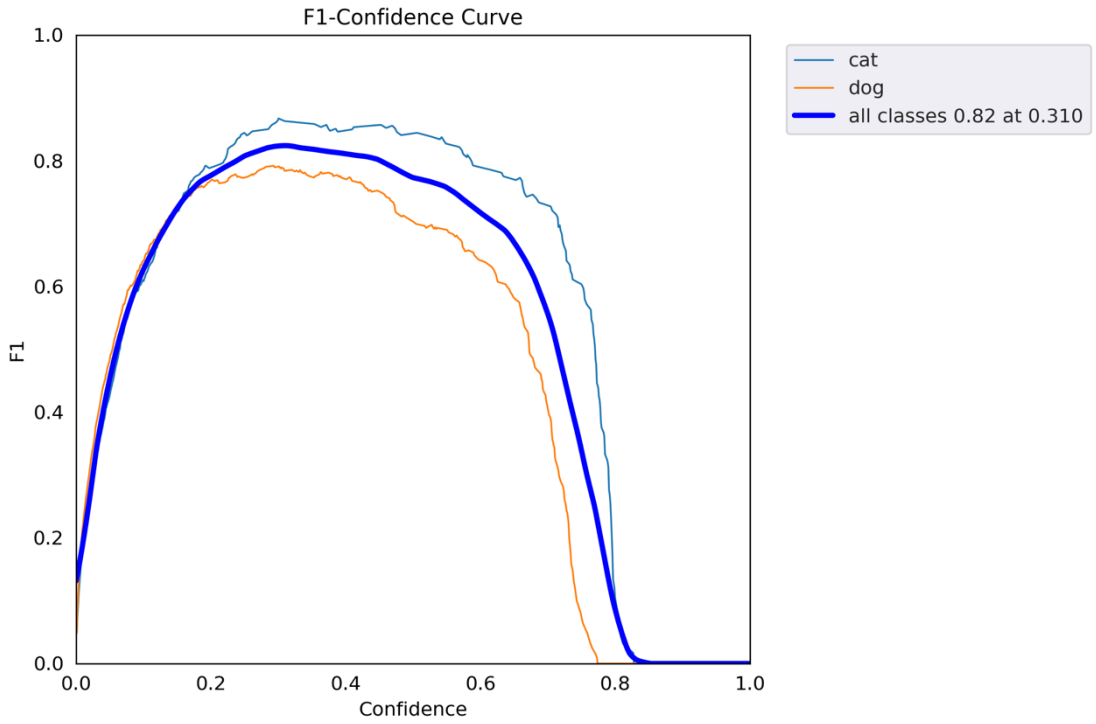
Testte elde edilen duyarlılık değeri tüm sınıflar için %77,1 iken bu değer kedi sınıfı için %82,9 ve köpek sınıfı için ise %71,3 olarak bulunmuştur. Bu durum kediler için yapılan tahminlerde gerçek kedi örneklerinin nadiren gözden kaçırıldığını ve köpekler için ise gözden kaçan örneklerin kedilere göre daha sık olduğunu göstermektedir. Şekil 4.8'de verilen duyarlılık-güven aralığı grafiğinde modelin %45'lik

güven aralığına kadar tatmin edici miktarda duyarlılık sağlayabildiği görülmektedir. Aynı zamanda kedi sınıfının köpek sınıfına göre hem daha yüksek hem de daha geniş güven aralığında duyarlılık sağladığı görülebilmektedir.



Şekil 4.8 ABC modeliyle yapılan testin Kesinlik-Güven Aralığı grafiği

Kedi ve köpek sonuçlarındaki kesinlik ve duyarlılık sonuçları bir arada değerlendirmeye alınacak olursa modelin köpek örneklerinde kedi grubuna göre ihmal edilebilecek seviyede daha sık yanlış pozitifler ürettiğini ve gerçek kedi örneklerini ise belirgin bir şekilde daha az gözden kaçırdığı görülmüştür. Köpek örnekleri ise model tarafından gözden kaçırılmaya daha müsait iken yapılan köpek tahminlerinin kedilerde olduğu gibi genellikle doğru olduğu görülmüştür. Şekil 4.9'da verilen F1 skoru-güven aralığı grafiği incelendiğinde modelin %31'lik güven aralığında 0,82 F1 skoru verdiği görülmektedir. Modelin verdiği F1 skorunun test verileri üstünde başarı elde ettiği görülmüştür.



Şekil 4.9 ABC modeliyle yapılan testin F1 Skoru-Güven Aralığı grafiği

Son olarak ise eğitilen modelin tüm sınıflar için %81,7, kedi sınıfı için %85,1 ve köpek sınıfı için %78,4 oranında mAP50 skoru verdiği görülmüştür. Elde edilen mAP50 skoru modelin test verileri için kabul edilebilir seviyede başarılı olduğunu ancak özellikle köpek verisi için kaçırılan örnekler üzerinde geliştirmelere ihtiyaç duyduğunu göstermektedir.

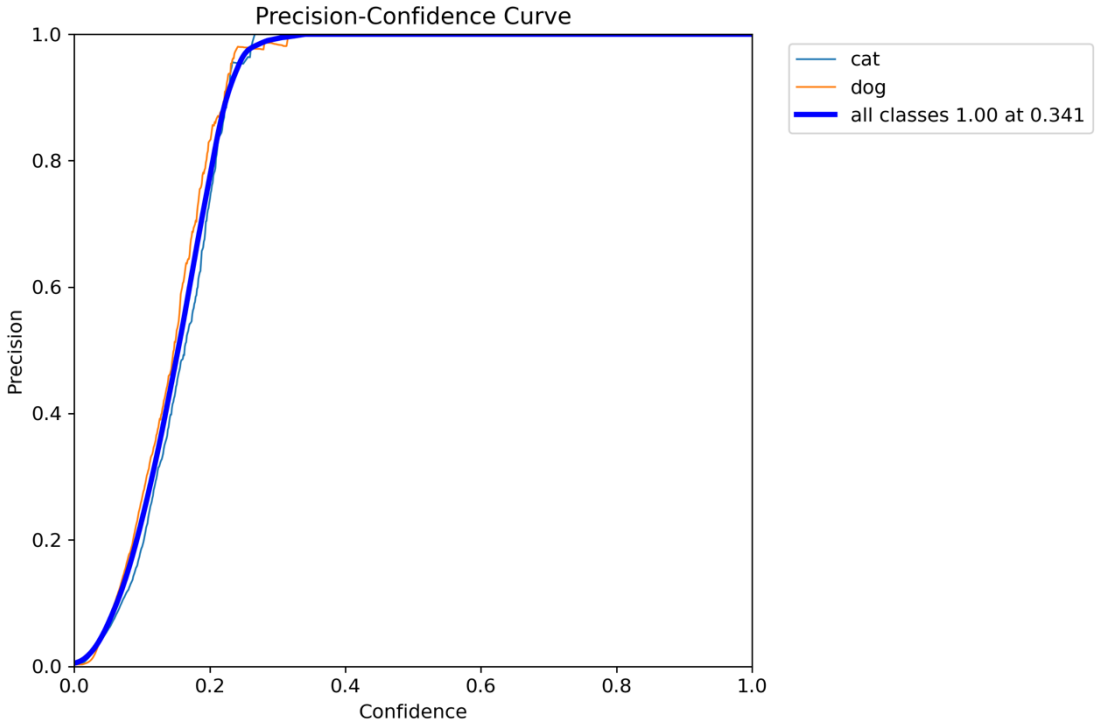
4.3.2 PSO ile optimize edilen modelin test sonuçları

PSO modeli ile yapılan test sürecinde Çizelge 4.7 ile verilen sonuçlar elde edilmiştir.

Çizelge 4.7. PSO modeli ile yapılan testin sonuçları

Sınıf	Örnek Sayısı	Kesinlik	Duyarlılık	F1 Skoru	mAP50
Tüm Sınıflar	368	0,873	0,602	0,69	0,768
Kedi	125	0,859	0,744	0,79	0,829
Köpek	243	0,888	0,461	0,6	0,707
Tahmin Hızı	41,7 milisaniye				

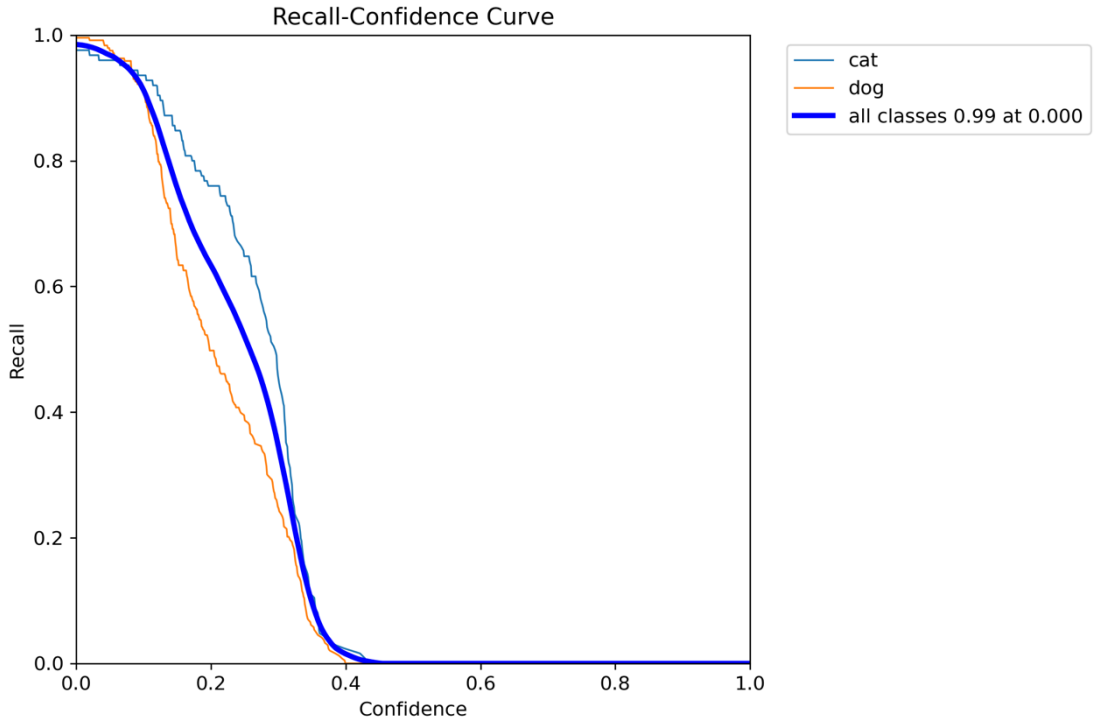
Modelin test verileri üstünde tüm sınıflar için %87,3 oranında kesinliğe sahip olduğu görülmüştür. Kesinlik oranları kedi sınıfı için %85,9 iken köpek sınıfı için %88,8 olarak görülmüştür. Modelin test verilerindeki kesinliğindeki düşüşün ABC modelinin test sonuçlarındaki düşüş kadar büyük olmadığı görülmüştür. Sınıflar arasındaki kesinlik farkı eğitim sonuçlarında olduğu gibi test sonuçlarında da göz ardı edilebilecek şekilde uyumludur. Bu durum modelin tahmin yapabildiği takdirde tüm sınıflar için tutarlı bir şekilde doğru sonuç verebildiğini göstermektedir. Şekil 4.10'da verilen kesinlik-güven aralığı grafiğinde modelin güven aralığı arttıkça kesinliğinin de arttığı görülmektedir. Grafikte de sunulduğu üzere modelin köpek sınıfı ve kedi sınıfı için benzer kesinlikli sonuçlar verdiği görülmektedir.



Şekil 4.10 PSO modeliyle yapılan testin Kesinlik-Güven Aralığı grafiği

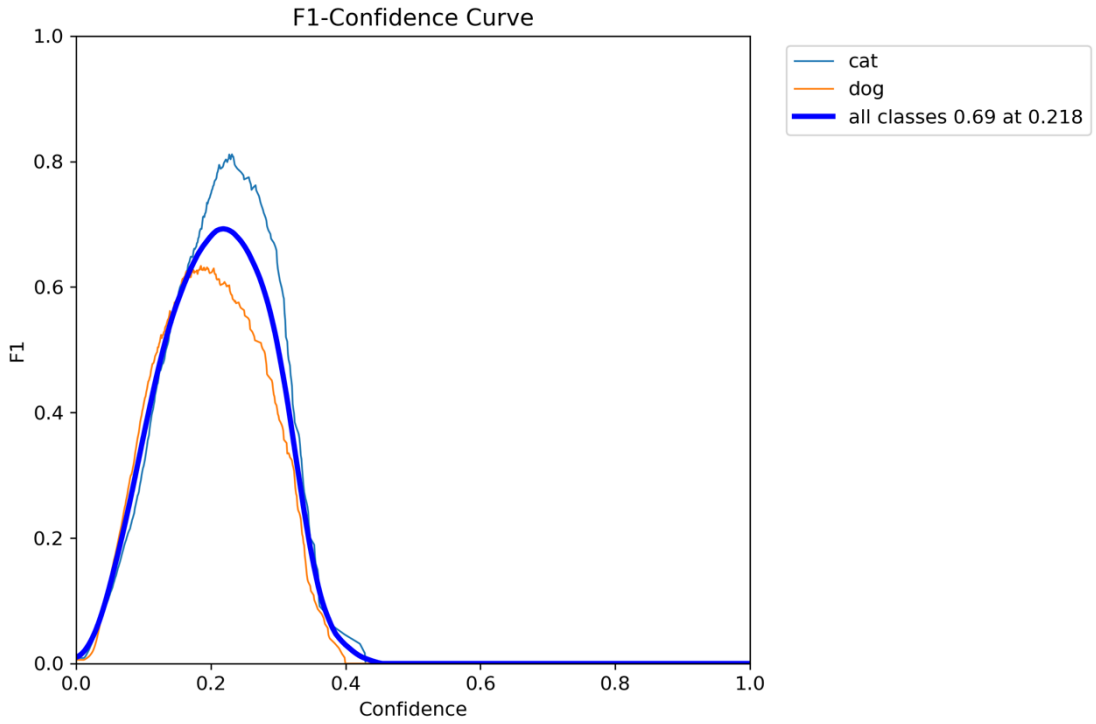
Testte elde edilen duyarlılık değeri tüm sınıflar için %60,2 iken bu değer kedi sınıfı için %74,4 ve köpek sınıfı için ise %46,1 olarak bulunmuştur. Bu durum kediler için yapılan tahminlerde gerçek kedi örneklerinin yarısından çoğunun tespit edilebildiğini ve köpekler için ise gözden kaçan örneklerin toplam köpek örneklerinin yarısından fazla olduğunu göstermektedir. Şekil 4.11'de verilen duyarlılık-güven aralığı grafiğinde modelin %15'lik güven aralığına kadar tatmin edici miktarda duyarlılık sağlayabildiğini

ancak ardından duyarlılığın hızlı kaybedildiğini göstermektedir. Aynı zamanda kedi sınıfının köpek sınıfına göre hem daha yüksek hem de daha geniş güven aralığında duyarlılık sağladığı görülmüştür.



Şekil 4.11 PSO modeliyle yapılan testin Duyarlılık-Güven Aralığı grafiği

Kedi ve köpek sonuçlarındaki kesinlik ve duyarlılık sonuçları bir arada değerlendirmeye alınacak olursa modelin kedi ve köpek örneklerinde benzer şekilde yanlış pozitifler ürettiğini ve her ne kadar tatmin edici duyarlılık sonuçları vermese de gerçek kedi örneklerinde köpek verilerine göre belirgin bir şekilde daha az gözden kaçırma olduğu görülmüştür. Köpek verilerinin ise model tarafından gözden kaçırılmaya çok müsait olduğu görülmüştür. Şekil 4.12'de verilen F1 skoru-güven aralığı grafiği incelendiğinde modelin %21,8'lik güven aralığında 0,69 F1 skoru verdiği görülmektedir. Modelin verdiği F1 skorunun test verileri üstünde yetersiz kaldığı görülmüştür.



Şekil 4.12 PSO modeliyle yapılan testin F1 Skoru-Güven Aralığı grafiği

Son olarak ise eğitilen modelin tüm sınıflar için %76,8, kedi sınıfı için %82,9 ve köpek sınıfı için %70,7 oranında mAP50 skoru verdiği görülmüştür. Elde edilen mAP50 skoru ve güven aralığı dağılımları modelin test verileri için her koşulda güvenilir olmayan sonuçlar ürettiğini göstermektedir.

4.3.3 Ön tanımlı hiperparametrelerle eğitilen modelin test sonuçları

Ön tanımlı hiperparametreler ile eğitilen modelin test sürecinde Çizelge 4.8 ile verilen sonuçlar elde edilmiştir.

Çizelge 4.8. PSO modeli ile yapılan testin sonuçları

Sınıf	Örnek Sayısı	Kesinlik	Duyarlılık	F1 Skoru	mAP50
Tüm Sınıflar	368	0,45	0,808	0,57	0,58
Kedi	125	0,343	0,711	0,46	0,267
Köpek	243	0,556	0,905	0,68	0,758
Tahmin Hızı	38,2 milisaniye				

Modelin test verileri üstünde tüm sınıflar için %45 oranında kesinliğe sahip olduğu görülmüştür. Kesinlik oranları kedi sınıfı için %34,3 iken köpek sınıfı için %55,6 olarak görülmüştür. Modelin duyarlılık başarımının tüm sınıflar için %80,8 olduğu görülmüştür. Duyarlılığın kedi ve köpek sınıfları için %71,1 ve %90,5 oranında olduğu görülmüştür. Modelin F1 skorunun tüm sınıflar, kediler ve köpekler için sırasıyla %57, %46 ve %68 olduğu görülmüştür. Son olarak ise modelin tüm sınıflar için %58 oranında mAP skoru verdiği görülmüştür. Kedi ve köpek sınıfları için ise %26,7 ve %75,8 oranında mAP50 skorları elde edilmiştir. Elde edilen verilere göre ön tanımlı hiperparametreler ile yapılan model eğitiminin tüm sınıflar için dengeli olamadığı, başarımın eğitimde daha fazla temsil edilen köpek sınıfı lehinde olduğu ve kedi sınıfı aleyhinde sonuçlar ürettiği gözlenmiştir.

4.3.4 ABC ve PSO ile optimize edilen modellerin karşılaştırılması

ABC ve PSO algoritmalarının optimize ettiği modellerin test sonuçlarını optimize edilmek için seçilen hiperparametreler bazında aşağıdaki gibi kıyaslayabiliriz.

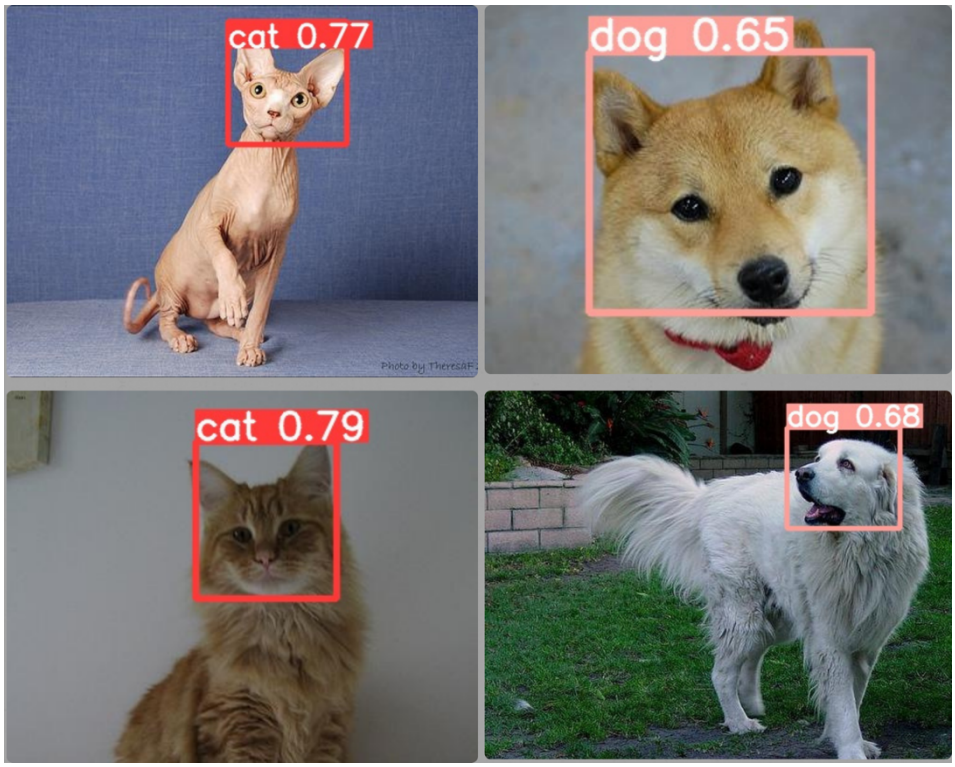
1. İlk Öğrenme Oran: ABC algoritmasının daha küçük bir öğrenme oranı seçmesi modelin yavaş ancak aşırı uymaya daha dirençli bir eğitim süreci geçirmesine sebep olmuştur. PSO modelinin eğitim ve test sürecinin birbiriyle tutarsız başarım göstermesi PSO'nun yüksek ilk öğrenme oranı seçiminin modeli aşırı uymaya ittiğini göstermektedir.
2. Final OneCycleLR Öğrenme Oranı: İki optimizasyon algoritması da benzer değerler seçtiği için bu hiperparametrenin model eğitimler ve test sonuçlarında önemli bir farklılık yaratmadığı düşünülmektedir.
3. Momentum: PSO hiperparametrelerindeki daha düşük momentum seçiminin geçmiş dönemlerden gelen öğrenilmiş bilginin daha az aktarılmasına sebep olmaktadır. Bu durum öğrenimi daha yavaş ilerlettiği için PSO modelinin daha fazla döneme ihtiyaç duyacak bir eğitime göre optimize edilmiş olmasına neden olmuştur.
4. Ağırlık Sönümü: İki optimizasyon algoritması da benzer değerler seçtiği için ağırlık sönümünün eğitim ve test üzerindeki etkilerinin fark yaratmadığı düşünülmüştür.
5. Pozitif Sınıf Ağırlığı: ABC'nin daha düşük pozitif nesne ağırlığı seçimi üretilen modelin belirli bir sınıfa ağırlık vermeden sınıflar arasında

eşitlikçi bir eğitim yapmasına neden olmuştur. Bu seçim özellikle eğitim sonuçlarında PSO modelinin daha az temsil edilen kedi sınıfını daha verimli öğrenmesini açıklamaktadır.

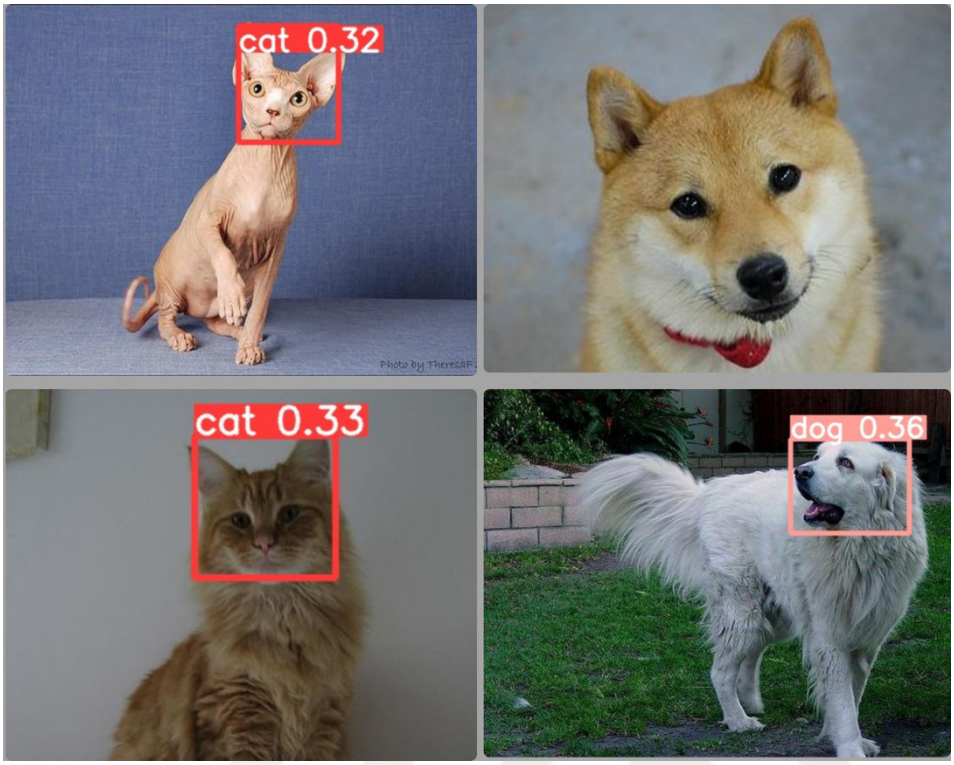
6. Pozitif Nesne Ağırlığı: PSO'nun daha yüksek pozitif nesne ağırlığı seçiminin eğitim ve test aşamalarında PSO modelinde daha yüksek duyarlılık ile sonuçlanması beklenmiştir. Ancak diğer hiperparametrelerin seçimleri bu durumun önüne geçmiştir. Eğitim sonuçlarında ABC ve PSO benzer duyarlılık skorları verirken test sonuçlarında ABC modelinin daha üstün duyarlılık sonucu verdiği görülmüştür. Bu durum ABC'nin seçtiği daha yüksek IoU eşiğinin ABC modelini eğitim esnasında daha seçici hale getirip daha verimli öğrenime itmesi ve ABC modelinin daha düşük ilk öğrenme oranı kullanıp verileri daha iyi öğrenmesi gibi nedenlere bağlıdır.
7. IoU Eşik Değeri: ABC modelinde kullanılan daha yüksek IoU eşik değeri modelin pozitif tahminler için daha katı bir eğitim stratejisi izlemesine yol açmıştır. Bu sayede ABC modeli eğitim sırasında kesinliğinden daha emin olunan örneklerin pozitif olarak işaretlenmesine ve bu örnekler üzerinden özellik öğrenilmesine odaklanmıştır. Bu sayede ABC modeli daha genel özelliklerin öğreniminde PSO modelinden daha başarılı olmuştur. Bu durum ABC ve PSO modellerinin test sonuçlarında ABC modelinin başarısına katkı sağlamıştır.
8. Çapa Eşiği: ABC optimizasyonu ile seçilen yüksek çapa eşiği değeri ABC modelinin eğitim sonuçlarının daha yüksek kesinlikte ve daha düşük duyarlılıkta olmasına katkıda bulunmuştur. PSO ile seçilen daha düşük çapa eşiği PSO modelinin eğitim sonuçlarında nispeten daha iyi duyarlılık sonuçları vermiştir. Ancak ABC algoritmasının diğer hiperparametre seçimlerinin başarısı ABC modelinin test verileri üstünde de güçlü sonuçlar vermesine destek olmuştur. PSO algoritmasının duyarlılıkta olumlu gelişmeler sağlayan çapa eşiği seçiminin etkilerini destekleyebilecek diğer hiperparametrelerde yeterince iyi seçim yapmaması test sonuçlarında etkisini göstermiştir.
9. Odak Kaybı Gama Değeri: PSO optimizasyonu ile işlevsel hale getirilen odak kaybı gama değeri PSO modelinin daha zor ve nadir örnekleri öğrenmek için daha fazla çaba sarf etmesine sebep olmuştur. Bu seçimin etkisi PSO modelinin eğitim aşamasında az örnekle temsil edilmiş kedi

verisinin daha yüksek kesinlikle tespit edilebilmesini sağlamıştır. Buna karşın daha zor örnekler üstünde yoğunlaşmak PSO modelinin genelleme yeteneğini de azalttığı için test aşamasında kayıplar yaşanmasına neden olmuştur.

ABC ile optimize edilen hiperparametrelerin daha düşük öğrenme oranı ve yüksek IoU eşiği seçimi eğitilen modelin daha dikkatli ve seçici olmasını sağlamıştır. ABC ile optimize edilen modelin test sonuçlarında PSO ile optimize edilen modele kıyasla daha yüksek duyarlılık ve mAP50 sonuçları ürettiği görülmüştür. Bu durum ABC algoritmasının seçtiği hiperparametrelerin modelin öğrenme becerisini optimize ettiğini ve çeşitli IoU eşiklerinde tutarlı performans göstermesini sağladığını göstermektedir. PSO algoritmasından elde edilen eğitim ve test sonuçlarındaki tutarsızlık ise PSO ile elde edilen hiperparametrelerin modeli aşırı öğrenmeye sürüklediğini ve bu sebeple PSO modelinin eğitim verilerinde iyi sonuçlar vermesine rağmen test verilerinde zayıf sonuçlar vermesine sebep olduğunu göstermektedir. PSO modelinin aşırı uyma yapmasının sebepleri olarak aşırı uyma riskini arttıran ağırlık sönümü, pozitif sınıf ağırlığı, pozitif nesne ağırlığının yüksek seçilmesi ve genelleştirme yeteneğini etkileyen IoU eşiği ve çapa eşiği parametrelerinin düşük seçilmesi olarak gösterilebilir.



Şekil 4.13 ABC modeliyle yapılan bazı tespitler



Şekil 4.14 PSO modeliyle yapılan bazı tespitler

Şekil 4.13 ve Şekil 4.14'te verilen görsellerde ABC modelinin ve PSO modelinin aynı görseller üstünde yaptığı tahminlere yer verilmiştir. ABC modelinin kedi ve köpekler üstünde başarılı tespitler yaptığı ve pozitif kabul edilebilecek tahmin skorları ürettiği görülmektedir. PSO modelinin ise aynı görsellerde ürettiği tahminlerin ABC modeli ile benzer sınırlayıcı kutular çizmesine rağmen tespit skorlarının kabul edilemeyecek kadar düşük olduğu veya tahmin üretemediği görülmektedir. Bu örnekler ABC modelinin daha başarılı bir öğrenme geçirdiğine dair bulgular sunmaktadır.

Ayrıca ABC modelinin tahmin hızı bakımından PSO modelinden 9 kat daha hızlı olması ve saniyede 217 kare üstünde tahmin yapabilmesi ABC ile yapılan optimizasyonun gerçek zamanlı uygulamalar için daha uygun bir sonuç ürettiğini göstermektedir.

Hiperparametre seçiminin yanı sıra veri kümesinin sınıf dağılımı, sınıflar içindeki özellik değişkenlikleri gibi faktörlerin de yapılan model eğitimlerinin başarıları üstünde etkileri olmuştur.

Veri kümesinde daha az temsil edilen kedi sınıfı optimizasyon algoritmalarını kedi sınıfının eksikliklerini giderecek çözümler üretmeye yönlendirmiştir. Bu durum PSO

optimizasyonunda belirgin bir şekilde kedi sınıfını daha iyi öğrenebilecek seçimler yapılmasına neden olmuştur. Bunlara ek olarak veri kümesinde kedilerin az örnekle gösterilmesine rağmen veri kümesinde 25 köpek ırkı ve 12 kedi ırkının bulunması köpek sınıfının kendi içinde daha çok özellik çeşitliliğine neden olmuştur. Bu durum modelin 25 farklı köpek ırkının yalnızca bir köpek sınıfı altında sınıflandırılması problemini doğurmuştur. Köpeklerin kedilere göre daha fazla çeşitlilik gösteren yüz oranları, burun ve kulak şekilleri vb. özellikleri köpek sınıfının kendi içinde genelleştirilmesi daha zor bir grup olmasına yol açmıştır. Bu çeşitliliğin etkileri hem ABC hem de PSO modellerinin köpek tahminleri yaparken çok örneğin tahmin yapılamadan ıskalanmasına sebep olmuştur.

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Yapılan deneyler sonucunda hiperparametre optimizasyonu ile elde edilen modellerin optimize edilmemiş modele göre üstünlük sağladığı görülmüşken, optimizasyon yapılan modeller arasında ABC algoritması ile yapılan hiperparametre optimizasyonunun PSO ile yapılan optimizasyona göre daha başarılı sonuçlar verdiği görülmüştür.

Optimizasyon süreleri göz önüne alındığında ABC ile yapılan optimizasyon, PSO ile yapılan optimizasyona kıyasla daha uzun sürede sonuçlanmıştır. Optimizasyon süreleri arasında oluşan fark ABC algoritmasının arama uzayında çeşitlilik yaratan yapısı nedeniyle uzun süren aramalar yaparken PSO'nun da hızlı yakınsayan doğası nedeniyle oldukça hızlı ilerlemesi ile açıklanabilmektedir.

Elde edilen hiperparametreler ile eğitim verileri kullanılarak yapılan eğitimlerde optimize edilen her iki modelin de güçlü sonuçlar verdiği görülmüştür. ABC modeli tüm sınıflar için %93,6 oranında mAP50 başarımına ulaşmışken PSO ise tüm sınıflarda %91,7 oranında bir başarım yakalamıştır. Her iki mAP50 skorunun da ön tanımlı hiperparametrelerle eğitilen modelin %56,6 oranındaki mAP50 skorundan üstün olduğu göze çarpmaktadır.

Eğitim sürecindeki diğer metriklerin incelenmesiyle ABC modelinin PSO modeline göre daha genelleştirilmiş bir eğitim gerçekleştirdiği görülmüştür. PSO modeli ise eğitimde genelleştirme yerine zor ve nadir örnekler üstüne ağırlık vermiştir. Bu durum PSO algoritmasının veriler arasındaki dengesizliği gidermeye yönelik bir çözüm kümesine doğru yakınsamaya çalıştığını; ABC algoritmasının ise tüm veriler için daha genel bir çözüm üretebilecek bir çözüm kümesine doğru yakınsamaya çalıştığını göstermektedir.

Üretilen modellerin test verileri üzerinde kullanıldığı deneyde, optimize edilmiş hiperparametreler ile üretilen modellerin ön tanımlı hiperparametrelerle elde edilen modele kıyasla daha başarılı olduğu görülmüştür. Ön tanımlı hiperparametrelerin testteki duyarlılık sonuçları optimize edilen hiperparametrelerle elde edilen duyarlılık sonuçları ile rekabetçi olsa da ön tanımlı hiperparametreler ile elde edilen kesinlik skorları diğer modellerin gerisinde kalmıştır. Bu durum ön tanımlı hiperparametrelerle eğitilen modelin F1 skoru ve mAP50 skorunda da geride kalmasına yol açmıştır. Optimize edilmiş

hiperparametrelerle eğitilen modeller karşılıklı olarak değerlendirmeye alındığında ise ABC algoritması ile optimize edilen modelin eğitim sonuçlarında olduğu gibi testte de başarılı sonuçlar ürettiği görülmüştür. PSO algoritması ile optimize edilen model ise test verileri üstünde eğitimde verdiği sonuçlardan daha başarısız sonuçlar üretmiştir. Bu durumun sebebi PSO algoritmasının bulduğu hiperparametrelerin aşırı uymaya yatkın, çözüme hızlı yakınsayan, genel özellikleri eğitmekten ziyade zor ve nadir örneklere odaklanan değerlere optimize edilmiş olması olarak değerlendirilmiştir. Test sonucu olarak ABC modeli tüm sınıflar için %81,7 mAP50 skorunu elde etmişken PSO modeli %76,8 mAP50 skorunu bulmuştur. PSO modelinin özellikle veri kümesinde daha çok temsil edilen köpek sınıfında düşük duyarlılık skoru üretmiştir ve yarıdan fazla köpek verisini tespit etmede başarısız olmuştur.

ABC algoritmasının genel özellikleri öğrenebilen hiperparametreler üretirken PSO algoritmasının aşırı uyma durumuna girmesi iki optimizasyon tekniğinin yapıları arasındaki farklılıklar ile açıklanabilmektedir. ABC algoritmasının potansiyel çözümlerin etrafından yaptığı yerel aramalar sayesinde çözüm çeşitliliği yaratması yerel minimumlarda takılma riskini azaltırken işçi ve gözcü arılar arasındaki bilgi paylaşımı sayesinde daha geniş bir arama alanında çözümlerin keşfedilmesini sağlamıştır. ABC algoritmasının bu özellikleri hiperparametre optimizasyonu sırasında aşırı uymaya sebep olabilecek yerel minimumlardan kaçmasını sağlamıştır. PSO algoritması ise sürü içindeki en iyi çözüme doğru ve parçacığın kendi en iyi çözümüne doğru yapılan ilerlemeler sayesinde yerel ve genel aramalar yapmaktadır. Bu durum bir miktar yerel arama sağlamasına rağmen globalde çok hızlı bir şekilde çözüme doğru yakınsamayı sağlamaktadır. Ancak bazı durumlarda parçacıklar ABC algoritmasındaki kadar arama yapamadığı için yerel minimumlara yakınsayabilmektedir. Hiperparametre optimizasyonu deneyinde PSO parçacıklarının aşırı uymaya neden olan bir yerel minimumda takılı kaldığı görülmüştür. Sonuç olarak ABC algoritmasının çeşitlilik yaratan yaklaşımı daha genel sonuçlar üretebilirken PSO algoritmasının hızlı ve ABC algoritmasına görece daha dar kapsamlı arama yöntemi aşırı uyma riskini arttırmıştır. Bu durum YOLOv5 hiperparametrelerinin optimizasyonu gibi yerel minimumlarda takılı kalmanın başarısızlığa sebep olduğu problemlerde ABC algoritmasının tercih edilmesinin daha dengeli ve genelleştirilmiş çözümler üretmede avantaj sağlayacağını göstermiştir.

5.2 Öneriler

Bu tez çalışmasında YOLOv5 hiperparametrelerinin optimizasyonu için kullanılan ABC ve PSO optimizasyon algoritmalarının model eğitimindeki başarıları incelenmiş ve elde edilen sonuçlar kıyaslanmıştır. Yapılan deneyler sonucunda ABC algoritmasının arama uzayında çeşitlilik yaratan özellikleri sayesinde daha uygun sonuçlar verdiği görülmüştür. Optimizasyonun doğası gereği ABC algoritmasının da verdiği sonuçların bazı açılardan geliştirilmeye açık olduğu görülmüştür. Bu kapsamda bu bölümde belirtilen öneriler daha verimli modeller üretmek için dikkate alınabilir.

ABC algoritmasının çözüm çeşitliliği sayesinde elde edilen genelleştirme yeteneği sayesinde PSO'ya göre yavaş ama daha isabetli bir şekilde hiperparametreler elde edildiği görülmüştür. ABC gibi çözüm çeşitliliği yaratan diğer optimizasyon algoritmalarının kullanımı daha verimli sonuçlar alınmasını sağlayabilir.

Bu çalışmada mAP50 metriğinin sonuçları üstünden optimizasyon yapılmıştır. Ancak YOLOv5 algoritması çeşitli metrik sonuçları verebilmektedir. mAP50 dışında kesinlik, duyarlılık, F1 skoru, mAP50-95 gibi metrikler YOLOv5 algoritması tarafından sağlanmaktadır. Bahsedilen bu metriklerin kullanımı optimizasyonun daha olumlu sonuçlar üretmesi konusunda etkilerde bulunabileceği düşünülmüştür. mAP50 yerine modelin daha düşük IoU değerlerindeki performansını ifade eden mAP50-95 metriği daha iyi bir seçim olarak kullanılabilir. Ayrıca birden çok metriğin bir arada kullanılması optimizasyonun daha doğru yönde ilerlemesini sağlayabilir. Bunu başarmak için çok amaçlı optimizasyon teknikleri kullanılabilir.

Optimizasyon algoritmalarına amaç fonksiyonu olarak verilen model eğitimleri bu deneyde 10 devir olarak tanımlanmıştır. Bu parametre seçimi hesaplama maliyeti göz önünde bulundurularak daha büyük değerlerde ayarlanabilir. Ancak aşırı uyma ve verimsiz devirlerden kaçınmak için devir sayısının dikkatle seçilmesi gerekmektedir. Bu amaçla optimizasyon sırasında kullanılan model eğitimleri için erken durma kriterleri tanımlanarak daha büyük devir sayılarının tanımlanması optimizasyonun performansını arttırabilir.

KAYNAKLAR

Abido, M. A., 2002, Optimal power flow using particle swarm optimization, *International Journal of Electrical Power & Energy Systems*, 24(7), 563-571.

Alibrahim, H., and Ludwig, S. A., 2021, Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization, *2021 IEEE Congress on Evolutionary Computation (CEC)*, Krakow, 1551-1559.

Andonie, R., and Florea, A. C., 2020, Weighted random search for CNN hyperparameter optimization, *arXiv preprint arXiv:2003.13300*.

Aszemi, N. M., and Dominic, P. D. D., 2019, Hyperparameter optimization in convolutional neural network using genetic algorithms, *International Journal of Advanced Computer Science and Applications*, 10(6), 269-278.

Bengio, Y., 2012, Practical recommendations for gradient-based training of deep architectures, *In Neural Networks: Tricks of the Trade: Second Edition*, 437-478.

Bergstra, J., and Bengio, Y., 2012, Random search for hyper-parameter optimization, *Journal of machine learning research*, 13(2), 281-305.

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

Desale, S., Rasool, A., Andhale, S., and Rane, P., 2015, Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey. *Int. J. Comput. Eng. Res. Trends*, 351(5), 2349-7084.

Domhan, T., Springenberg, J. T., and Hutter, F., 2015, Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves, *Twenty-fourth international joint conference on artificial intelligence*, Buenos Aires, 3460-3468.

Dorigo, M., Birattari, M., and Stutzle, T., 2006, Ant colony optimization., *IEEE computational intelligence magazine*, 1(4), 28-39.

Erkan, U., Toktas, A., and Ustun, D., 2022, Hyperparameter optimization of deep CNN classifier for plant species identification using artificial bee colony algorithm. *Journal of Ambient Intelligence and Humanized Computing*, 14, 1-12.

Guo, Y., Li, J. Y., and Zhan, Z. H., 2020, Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approachi *Cybernetics and Systems*, 52(1), 36-57.

Horng, M. H., 2011, Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation, *Expert Systems with Applications*, *38*(11), 13785-13791.

Jiang, P., Ergu, D., Liu, F., Cai, Y., and Ma, B., 2022, A Review of Yolo algorithm developments. *Procedia Computer Science*, 199, 1066-1073.

Jocher, G., Changyu, L., Hogan, A., Yu, L., Rai, P., and Sullivan, T., 2020, ultralytics/yolov5: Initial Release [online], Cambridge, Harvard University, https://ui.adsabs.harvard.edu/abs/2020zndo...3908560J/abstract [Ziyaret Tarihi: 8 Ocak 2024].

Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization., *Erciyes university engineering faculty computer engineering department Technical report-tr06*, *Kayseri*, Vol. 200, 1-10.

Karaman, A., Karaboga, D., Pacal, I., Akay, B., Basturk, A., Nalbantoglu, U., Coskun, S. and Sahin, O., 2023, Hyper-parameter optimization of deep learning architectures using artificial bee colony (ABC) algorithm for high performance real-time automatic colorectal cancer (CRC) polyp detection, *Applied Intelligence*, 53(12), 15603-15620.

Kennedy, J., and Eberhart, R., 1995, Particle swarm optimization, *In Proceedings of ICNN'95-international conference on neural networks*, Perth, Vol. 4, 1942-1948.

Kouhalvandi, L., and Matekovits, L., 2022, Hyperparameter Optimization of Long Short-Term Memory-Based Forecasting DNN for Antenna Modeling Through Stochastic Methods, *IEEE Antennas and Wireless Propagation Letters*, 21(4), 725-729.

Lake, B. M. and Baroni, M., 2023, Human-like systematic generalization through a meta-learning neural network. *Nature*, 623, 115-121.

Lee, Y. H., and Kim, Y., 2020, Comparison of CNN and YOLO for Object Detection, *Journal of the semiconductor & display technology*, 19(1), 85-92.

Liberti, L., and Kucherenko, S., 2005, Comparison of deterministic and stochastic approaches to global optimization. *International Transactions in Operational Research*, 12(3), 263-285.

Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S., and Pastor, J. R., 2017, Particle swarm optimization for hyper-parameter selection in deep neural networks, *genetic and evolutionary computation conference*, Berlin, 481-488.

Lorenzo, P. R., Nalepa, J., Ramos, L. S., and Pastor, J. R., 2017, Hyper-parameter selection in deep neural networks using parallel particle swarm optimization, *genetic and evolutionary computation conference*, Berlin, 1864-1871.

Loshchilov, I., and Hutter, F., 2016, CMA-ES for hyperparameter optimization of deep neural networks, *arXiv preprint arXiv:1604.07269*.

Oliva, D., Ewees, A. A., Abd El Aziz, M., Hassanien, A. E., and Cisneros, M. P., 2017, A chaotic improved artificial bee colony for parameter estimation of photovoltaic cells, *Energies*, 10(7), 865-883.

Ozaki, Y., Yano, M., and Onishi, M., 2017, Effective hyperparameter optimization using Nelder-Mead method in deep learning. *IPSJ Transactions on Computer Vision and Applications*, 9, 1-12.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V., 2012, Cats and dogs, *2012 IEEE conference on computer vision and pattern recognition*, Providence, 3498-3505.

Redmon, J., and Farhadi, A., 2017, YOLO9000: better, faster, stronger. conference on computer vision and pattern recognition, Honolulu, 7263-7271.

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

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., 2016, You only look once: Unified, real-time object detection, *In Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, 779-788.

Shi, Y. and Eberhart, R. C., 2001, Fuzzy adaptive particle swarm optimization, *2001 congress on evolutionary computation*, Seoul, 101-106.

Shi, Y. and Eberhart, R., 1998, A modified particle swarm optimizer, *1998 IEEE international conference on evolutionary computation proceedings*, San Diego. 69-73.

Snoek, J., Larochelle, H., and Adams, R. P., 2012, Practical bayesian optimization of machine learning algorithms, *Advances in neural information processing systems*, 25, 2951-2959.

Sonmez, M., 2011, Artificial Bee Colony algorithm for optimization of truss structures. *Applied Soft Computing*, 11(2), 2406-2418.

Tan, L., Huangfu, T., Wu, L., and Chen, W., 2021, Comparison of YOLO v3, faster R-CNN, and SSD for real-time pill identification [online], Research Square, https://www.researchsquare.com/article/rs-668895/latest [Ziyaret Tarihi: 8 Ocak 2024].

Tang, J., Liu, G., and Pan, Q., 2021, A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA Journal of Automatica Sinica*, 8(10), 1627-1643.

Wang, Z., Jin, L., Wang, S., and Xu, H., 2022, Apple stem/calyx real-time recognition using YOLO-v5 algorithm for fruit automatic loading system, *Postharvest Biology and Technology*, 185, 111808.

Zatarain Cabada, R., Rodriguez Rangel, H., Barron Estrada, M. L., and Cardenas Lopez, H. M., 2020, Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoring systems, *Soft Computing*, 24(10), 7593-7602.

Zhang, M., Li, H., Pan, S., Lyu, J., Ling, S., and Su, S., 2021, Convolutional neural networks-based lung nodule classification: A surrogate-assisted evolutionary algorithm for hyperparameter optimization, *IEEE Transactions on Evolutionary Computation*, 25(5), 869-882.