T.C.
NECMETTİN ERBAKAN
ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME YÖNTEMİ İLE İDRAR SEDİMENT GÖRÜNTÜLERİNDEKİ HÜCRELERİN TESPİTİ VE SINIFLANDIRILMASI

Yusuf AKBAŞ

YÜKSEK LİSANS TEZİ

Mekatronik Mühendisliği Anabilim Dalı

Eylül - 2023
KONYA

TEZ KABUL VE ONAYI

Yusuf AKBAŞ tarafından hazırlanan "Derin Öğrenme Yöntemi İle İdrar Sediment Görüntülerindeki Hücrelerin Tespiti ve Sınıflandırılması" adlı tez çalışması 21/09/2023 tarihinde aşağıdaki jüri tarafından oy birliği ile Necmettin Erbakan Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan **Prof. Dr. Sabri KOÇER**	**……..**
Danışman **Doç. Dr. İlhan İLHAN**	**……..**
Üye **Dr. Öğr. Üyesi Emrehan YAVŞAN**	**……..**
Üye **Unvanı Adı SOYADI**	**……..**
Üye **Unvanı Adı SOYADI**	**……..**

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …./…/20.. gün ve …….. sayılı kararıyla onaylanmıştır.

Prof. Dr. Şerife Yurdagül KUMCU
FBE 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.

Yusuf AKBAŞ

Tarih: …./09/2023

ÖZET

YÜKSEK LİSANS TEZİ

DERİN ÖĞRENME YÖNTEMİ İLE İDRAR SEDİMENT GÖRÜNTÜLERİNDEKİ HÜCRELERİN TESPİTİ VE SINIFLANDIRILMASI

Yusuf AKBAŞ

NECMETTİN ERBAKAN ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Danışman: Doç. Dr. İlhan İLHAN

2023, 66 Sayfa

Jüri
Doç. Dr. İlhan İLHAN
Prof. Dr. Sabri KOÇER
Dr. Öğr. Üyesi Emrehan YAVŞAN

Günümüzde, doktorlar tarafından en sık istenilen testlerden biri de idrar testidir. Bunun en büyük nedenlerinden biri idrarın insan metabolizması hakkında birçok bilgiyi içermesidir. Diğeri ise örnek alımının oldukça kolay olmasıdır. Alınan örneklerin analizi için tam otomatik idrar analizörleri kullanılmaktadır. Fakat bu analizörlerin birçoğu sonradan öğrenme yeteneğine sahip değildir. Yani, daha önceden tanıtılmayan bir maddenin cihaza öğretilmesi zordur. Bu cihazlar analiz esnasında, genellikle, kenar ve renk tespiti gibi ilkel görüntü işleme algoritmalarını kullanırlar. Bu algoritmalar, hava kabarcığı gibi fiziksel nesnelerin ölçüme dahil edilmesine neden olurlar. Bu da testin doğruluğu açısından, laboratuvarda çalışan laborantlar için oldukça problemli bir durumun ortaya çıkmasına neden olur.

Bu tez çalışmasında, yukarıda bahsedilen idrar analizörlerindeki problemlere derin öğrenme yöntemi ile çözüm bulunmaya çalışılmıştır. İkisi hazır biri ise idrar analiz cihazından olmak üzere üç farklı veri kümesi elde edilmiş ve bu veri kümeleri üzerinde birtakım deneyler uygulanmıştır. Bu deneyler sonucunda, eritrosit tespitinde %94, lökosit tespitinde ise %87 seviyelerine ulaşan bir doğruluk oranı elde edilmiştir. Böylece, insan idrarındaki partiküller YOLOv7-tiny derin öğrenme modeli kullanılarak başarıyla tespit edilmiştir.

Anahtar Kelimeler: Derin Öğrenme, İdrar Analizi, Sınıflandırma, YOLOv7

ABSTRACT

MS THESIS

DETECTION AND CLASSIFICATION OF CELLS IN URINE SEDIMENT IMAGES WITH DEEP LEARNING METHOD

Yusuf AKBAŞ

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF NECMETTİN ERBAKAN UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN MECHATRONIC ENGINEERING

Advisor: Assoc. Prof. Dr. İlhan İLHAN

2023, 66 Pages

Jury
Assoc. Prof. Dr. İlhan İLHAN
Prof. Dr. Sabri KOÇER
Asst. Prof. Dr. Emrehan YAVŞAN

In today's world, one of the most requested tests by doctors is the urine test. One of the main reasons for this is that urine contains a wealth of information about human metabolism. Another reason is that urine sample collection is relatively easy. Fully automated urine analyzers are used for the analysis of the collected samples. However, many of these analyzers do not have the ability to learn retrospectively. In other words, it is difficult to teach the device about a substance that has not been previously introduced. During the analysis, these devices typically utilize primitive image processing algorithms such as edge and color detection. These algorithms can lead to the inclusion of physical objects such as air bubbles in the measurement. This poses a significant problem for laboratory technicians in terms of the accuracy of the test.

In this thesis, the problems in urine analyzers mentioned above were tried to be solved by deep learning method. Three different datasets, two of which are ready and one from the urine analysis device, were obtained and some experiments were performed on these datasets. As a result of these experiments, an accuracy rate of 94% in erythrocyte detection and 87% in leukocyte detection was achieved. Thus, particles in human urine were successfully detected using the YOLOv7-tiny deep learning model.

Keywords: Classification, Deep Learning, Urine Analysis, YOLOv7

ÖNSÖZ

Bu çalışmamın planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın danışman hocam Doç. Dr. İlhan İLHAN'a sonsuz teşekkürlerimi sunarım.

Çalışmanın başından sonuna kadar hep yanımızda olan, desteğini bizlerden esirgemeyen, yönlendirme ve bilgilendirmeleriyle bizlere destek olan Veyis ALTAY'a sonsuz teşekkür ederim.

Her zaman yanımda olan maddi ve manevi desteklerini hiç esirgemeyen çok değerli anne, babama ve kardeşlerime sonsuz teşekkür ederim.

Akademik kaynaklara erişim için Necmettin Erbakan Üniversitesi Kütüphane ve Dokümantasyon Daire Başkanlığı'nın bizlere sağladığı kolaylık için üniversite yönetimine teşekkürü bir borç bilirim.

Yusuf AKBAŞ
KONYA-2023

İÇİNDEKİLER

ŞEKİLLER

ÇİZELGELER

SİMGELER VE KISALTMALAR

Kısaltmalar

BACT	: Bakteri Hücresi
BYST	: Maya Hücresi
CAOX	: Kalsiyum Oksalat Kristali
CCD	: Kamera Sensörü
CNN	: Evrişimsel Sinir Ağları
CT	: Döküntü (Cast)
CYT	: Kristal Hücre
ELAN	: Geliştirilmiş Verimli Uzun Menzilli Toplama Ağı
EP	: Epitel Hücresi
ESA	: Evrişimsel Sinir Ağları
GPU	: Grafik İşlem Ünitesi
HYAL	: Şeffaf Döküntüler (Hyaline Cast)
MLP	: Çok Katmanlı Algılayıcı
MUCS	: Mukus Hücresi
NC	: Hücre Dışı Materyal (Non-cell)
pH	: Asitlik Derecesi
RBC	: Kırmızı Kan Hücresi
SPRM	: Sperm Hücresi
SQEP	: Skuamöz Epitel Hücresi
WBC	: Beyaz Kan Hücresi
WBCC	: Beyaz Kan Hücresi Topluluğu
YSA	: Yapay Sinir Ağı
YST	: Maya Hücresi

1. GİRİŞ

1.1 İdrar Analizi

Böbrek hastalıkları, küresel ölçekte milyonlarca insanı etkileyen bir sağlık sorunudur. Her yıl 830.000 kişinin ölümüne sebep olan böbrek ve üriner sistem hastalıklarıyla ilgili olarak 18.467.000 kişi risk altında bulunmaktadır (Suhail ve Brindha, 2021). Özellikle diyabet, metabolik hastalıklar, idrar yolu enfeksiyonları ve böbrek hastalıklarının teşhisinde, hastalardan idrar örnekleri alınması yaygın bir uygulamadır. Bu örnekler, tam otomatik idrar analiz cihazları ve biyokimyacılar tarafından ayrıntılı bir şekilde incelenmektedir.

Normal bir idrarın yaklaşık %95'i su, geriye kalan kısmı ise gıda ve metabolizma sonucunda oluşan çözünmeyen atıklardan oluşur (Memişoğullari ve ark., 2008). İdrar tahlili, kolay verilmesinin yanında metabolizma, böbrek ve üriner sistem hakkında son derece önemli bilgiler sağlamaktadır (Huysal ve Üstündağ, 2015). Ucuz, kolay ve hızlı olması nedeniyle idrar tahlili, doktorlar tarafından sıkça istenmekte ve hastane laboratuvarlarında en çok talep edilen testler arasında yer almaktadır (Perazella, 2015). Tam idrar analizi, idrarın fiziksel, kimyasal (pH, kan, glukoz, vb.) ve mikroskobik (kırmızı kan hücreleri, beyaz kan hücreleri, bakteriler, vb.) olarak incelenmesi olmak üzere üç ana bölümden oluşmaktadır (Huysal ve Üstündağ, 2015).

Fiziksel inceleme için, idrarın miktarı, rengi, kokusu, görünümü ve yoğunluğu gibi parametreler değerlendirilir. Bu değerlendirmeler, idrar analizinde önemli bir rol oynamaktadır ve çeşitli hastalıkların teşhisinde yardımcı olabilir. Bununla birlikte, kesin bir tanı koymak için sadece fiziksel inceleme yeterli olmayabilir ve laboratuvar testleri ve diğer tıbbi değerlendirmeler gerekebilir (Memişoğullari ve ark., 2008). Kimyasal inceleme için genellikle çeşitli kimyasal maddeler kullanılır ve bu kimyasal bileşikler standartlaştırılmıştır. Bu kimyasallar, idrar üzerine damlatıldığında renk değişikliği meydana getirir ve idrarın kimyasal analizi renk değişimine göre yapılır. Mikroskobik inceleme için ise normal şartlarda idrar örnekleri laboratuvar çalışanı tarafından incelenir. Ancak günümüzde, yüksek hasta sayısı, analizi yapan personelin deneyimi ve inceleme süresinin uzunluğu gibi nedenlerle tam otomatik idrar analizörleri bu türden incelemeler için geliştirilmiştir (Huysal ve Üstündağ, 2015).

Çoğu mevcut idrar analizörleri, kenar tespiti, renk ayrımı gibi klasik görüntü işleme yöntemlerini kullanarak ölçüm yapmaktadırlar. Bu nedenle bu cihazlar, sonradan

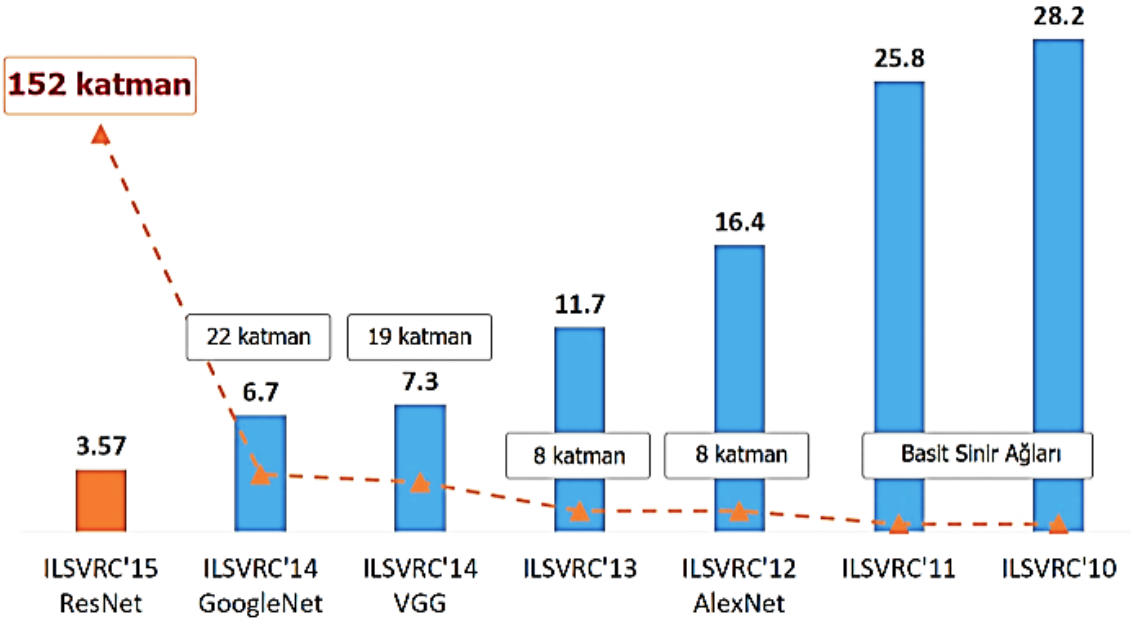
öğrenme yeteneğine sahip değillerdir. İdrar görüntüleri incelendiğinde, bazı partiküllerin birbirine çok benzeyen özelliklere sahip olduğu gözlenmektedir. Bu görüntülerin geleneksel yöntemlerle ve yüksek doğrulukla tespit edilmesi oldukça zordur (Huysal ve Üstündağ, 2015; Li ve ark., 2020). Ayrıca, hava kabarcığı gibi fiziksel nesneler de ölçüme dahil edilebilir. Bu durum, yanlış sonuçların oluşmasına ve laboratuvar çalışanlarının daha fazla çaba harcamalarına neden olur.

1.2 Derin Öğrenme

Derin öğrenme, yapay zekanın bir dalı olan makine öğrenmesinin özel bir formudur. Derin yapay sinir ağları, klasik yapay sinir ağlarının çok katmanlı ve çok nöronlu bir örneğidir. Derin öğrenme ağlarının önemli bir özelliği, probleme uygun özelliklerin çıkarılması için ek bir çaba gerektirmeden bu işi otomatik olarak yapabilmesidir (Kızrak ve Bolat, 2018). Öznitelikler, bir görüntü için piksel yoğunluk değerlerinin bir vektörü, kenar kümeleri veya nesneye özgü şekiller gibi farklı özellikler olabilir. Bu özelliklerden biri veya birkaçı, nesneyi daha iyi temsil edebilir (Şeker ve ark., 2017). Derin yapay sinir ağları, kendisine sunulan bilgiyi kullanmak yerine, öğreneceği bilgiyi kendi başına belirleme yetisine sahiptir. Bu nedenle, derin öğrenme ağları, klasik yöntemlere göre daha başarılı sonuçlar üretmektedir (Kızrak ve Bolat, 2018).

Derin öğrenme, 2012 yılında bilim dünyasında büyük bir etki yaratmıştır. ImageNet, nesne tespiti alanında düzenlenen en büyük yarışmalardan biridir. Yarışma 2010 yılında başlamış ve her yıl düzenlenmeye devam etmektedir. 2012 yılındaki yarışmada AlexNet (Krizhevsky ve ark., 2017) modeli birinci olmuştur. ImageNet'in geçmiş yarışmalarında en düşük hata oranı %26,1 iken 2012 yılında ESA modelinin hata oranı %15,3 gibi bir orana düşürülmüştür. Bu nedenle, derin öğrenme 2012 yılında büyük ilgi görmüş ve popülerlik kazanmıştır (İnik ve Ülker, 2017). Yarışma sürecinde, 2010-2015 yılları arasında katılımcıların geliştirdiği evrişimsel sinir ağları (Convolutional Neural Networks) sayesinde görüntü sınıflandırma hata oranı 7,89 kat (%28,2'den %3,57'ye) ve nesne tespiti hata oranı da 4,7 kat azalmıştır (%42,5'ten %9'a) (Bayram, 2020).

Şekil 1.1'de görüldüğü gibi AlexNet modeli ile 2012 yılında ESA ile elde edilen başarı her sene artmıştır. Bu başarının en önemli nedeni, sinir ağındaki katman sayısının artmasıdır. Katman sayısının artması, ağın daha derin bir yapıya sahip olmasını sağlar (Bayram, 2020).



Şekil 1.1. 2010-2015 yılları arasında ImageNet yarışmasında elde edilen hata oranları ve katman sayıları (Bayram, 2020)

Derin öğrenme, uzun bir geçmişe sahip bir alan olmasına rağmen, geçmiş dönemlerde başarılı bir şekilde kullanılamamıştır. Bu durumun temel nedenlerinden biri, geçmişte yeterli miktarda veriye sahip olunamamasıdır. Günümüzde, bu algoritmaların sağlıklı bir şekilde çalışabilmesi için gereken yüksek miktarda kaynaklar sağlanabilmektedir. Bu kaynakların ilki, veridir. Toplumun dijitalleşmesinin artmasıyla birlikte veri oluşturmak kolay ve maliyet açısından uygun hale gelmiştir. Bir diğer kaynak ise hesaplama gücüdür. Derin öğrenme ağları, gizli katmanların artmasıyla derinleşirken daha büyük bellek ve daha hızlı bilgisayar kaynaklarına ihtiyaç duyar. Aynı şekilde birden fazla gizli katmana sahip bir ağın eğitimi sırasında geriye yayılım gibi algoritmalar kullanılır. Bu algoritmalar kullanılırken yapılan hesaplamalar paralel olarak daha hızlı bir şekilde gerçekleştirilebilir. Bu nedenle, derin ağların eğitimi için işlemci olarak paralel hesaplama yeteneği daha yüksek olan grafik kartları kullanılmaktadır (Bayram, 2020).

Son yıllarda, grafik işlem birimi (GPU) performansının artması, derin öğrenme alanında önemli gelişmelere yol açmıştır. Evrişimsel sinir ağları (ESA), özellikle görüntü analizi alanında önemli bir rol oynamaktadır. Geleneksel yöntemlere kıyasla, ESA daha fazla özelliği otomatik olarak çıkarabilir ve özelliklerin kombinasyonunu optimize edebilir (Ji ve ark., 2019).

Bu tez çalışmasında, mevcut idrar analiz cihazlarının sorunlarına bir çözüm olarak, geleneksel görüntü işleme yöntemleri yerine daha yeni ve daha doğru çalışan bir

derin öğrenme modeli olan YOLOv7 kullanılmıştır. Bu model ile idrardaki lökosit ve eritrositler tespit edilmiştir.

2. KAYNAK ARAŞTIRMASI

Günümüzde idrardaki hücrelerin tespiti üzerine birçok çalışma yapılmıştır. Bu çalışmalarda farklı hücre tipleri, algoritmalar, veri setleri ve parametreler kullanılmıştır. Bu bölümde, günümüze kadar yapılan bu çalışmalar hakkında bilgi verilmektedir.

Liang ve ark. (2018), idrar partiküllerinin tespiti ve sınıflandırması için Faster R-CNN ve SSD (single-shot multi-box detector) yöntemlerini birlikte kullanmışlardır. Ayrıca, SSD'nin performansını artırmak için "Trimmed SSD" adında bir yöntem önermişlerdir. Çalışmada, 7 farklı sınıftan oluşan (eritrosit, lökosit, epitel hücresi, kristal, döküntü, mantar, epitel çekirdeği) ve 5376 görüntü içeren veri seti kullanılmıştır. Görüntüler klinik uzmanlar tarafından işaretlenmiştir. Tamamı 800x600 boyutundadır. Çalışma sonucunda, ortalama doğruluk oranının %84,1 olduğu belirtilmiştir.

Li ve ark. (2019), idrar sediment görüntülerini sınıflandırmak için LeNet-5 modelini kullanmışlardır. LeNet-5, aslında bankacılık sektöründe el yazısı karakterlerini tanımak için tasarlanmış bir modeldir. Bu çalışmada, LeNet-5 modeli üzerinde bazı değişiklikler yapılmış idrar sediment görüntülerinin tanınması sağlanmıştır. Bu değişikliklerden ilki, çıktı katmanının 10'dan 4'e indirilmesidir. Diğeri, çekirdek sayısının arttırılmasıdır. Bir diğeri ise aktivasyon fonksiyonu olarak sigmoid yerine ReLu fonksiyonunun kullanılmasıdır. Li ve ark. sınıflandırıcı fonksiyonu olarak Softmax kullanmışlardır. Deneylerde, 4 sınıftan oluşan 2551 idrar sediment görüntüsü ile %92 oranında bir doğruluk elde edilmiştir.

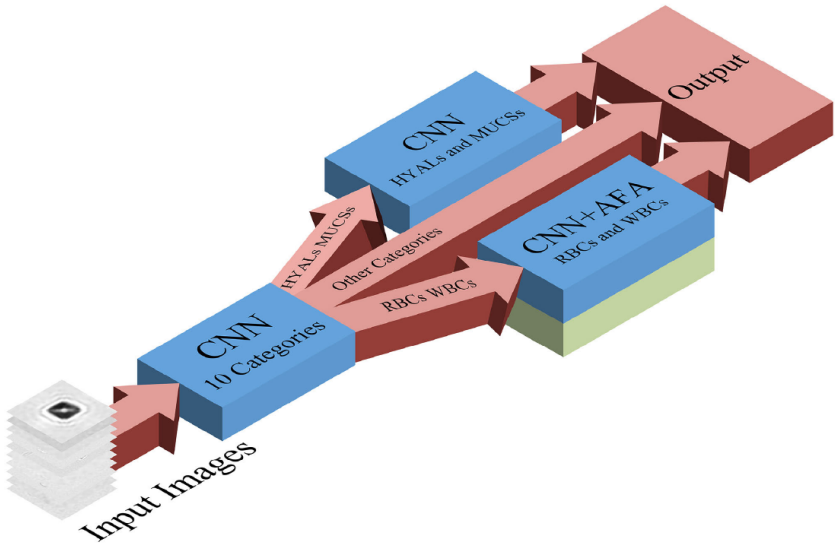
Zhang ve ark. (2019), idrar sediment partiküllerini sınıflandırmak için Adam optimizasyon yöntemini kullanarak yeni bir model önermişlerdir. Çalışmada, veri seti mikroskop altında alınan 1550 adet görüntü ile oluşturulmuştur. Önerilen modelin %97 doğruluk oranıyla, diğer derin öğrenme modellerine göre en iyi sonucu veren model olduğu belirlenmiştir.

Huagoi ve ark. (2019), idrardaki kalsiyum oksalat kristallerini derin öğrenme ile tespit etmeye çalışmışlardır. Hastanede yapılan testlerde idrarda kalsiyum oksalat tespit edilen 80 hasta için 749 görüntü kaydedilmiştir. Toplanan idrarlar yeniden mikroskop altında incelenmiş ve CCD kamerayla kalsiyum oksalat kristalinin görüntüleri kaydedilmiştir. ResNet mimarisinin kullanıldığı çalışmada, ağa özelleştirilmiş iki ek katman eklenerek ölçüm doğruluğunun artırıldığı belirtilmiştir. Yaklaşık %75 doğruluk oranıyla 344 kalsiyum oksalat kristalinin tespit edildiği ifade edilmiştir (Çizelge 2.1).

Çizelge 2.1. Huagoi ve ark. yaptığı çalışmanın sonucu (Xiang ve ark., 2019)

Partikül Tipi	Toplam Sayısı	Doğru Tahmin Sayısı	Yanlış Tahmin Sayısı	Kaçırılan Tahmin Sayısı	Doğruluk
Kalsiyum Oksalat Kristali	344	258	32	16	0.74

Qingbo ve ark. (2019) tarafından gerçekleştirilen çalışmada, veri seti hastalardan elde edilmiştir. Toplanan örneklerin mikroskop altında görüntüleri kaydedilmiştir. Her bir örnekten, 650 yüksek çözünürlüklü görüntü alınmıştır. Toplamda, 300.000 görüntü içeren bir veri kümesi kullanılmıştır. Çalışmada, BACT, BYST, CAOX, HYAL, MUCS, eritrosit (RBC), SPRM, SQEP, lökosit (WBC) ve WBCC partiküllerinin tespiti yapılmıştır. Şekil 2.1'de görüldüğü gibi, ana ağ modülü öncelikle giriş görüntülerini 10 kategoriye göre tanımaktadır. Eğer görüntüler RBC'ler veya WBC'ler olarak tanınırsa, bu tanımlama ile ilgili RBC-WBC, ikincil tanıma modülüne yönlendirilir. Eğer görüntüler HYAL veya MUCS olarak tanınırsa, bu tanımlama ile ilgili HYAL-MUCS, birincil tanıma modülüne gönderilir. Farklı dört kategoriden olan görüntüler ise doğrudan sınıflandırıcıya ulaştırılır. Çalışmada temel model olarak AlexNet modeli kullanılmıştır. Modelin doğruluk oranı %97'dir ve ortalama tanıma süresi 6ms olarak belirlenmiştir (Ji ve ark., 2019).



Şekil 2.1. Qingbo ve ark. yaptığı çalışmanın diyagramı (Ji ve ark., 2019)

Li ve ark. (2020) yaptıkları çalışmada, toplamda 16 farklı idrar partikülü içeren bir veri seti kullanmışlardır. Bu veri seti, 384 hastadan toplanan idrar örneklerinin mikroskop altındaki görüntülerinin kaydedilmesiyle oluşturulmuştur. Her bir hastadan 40

adet görüntü alınmış ve toplamda 15.360 idrar görüntüsü elde edilmiştir. Görüntüler 2048x1536 çözünürlüğünde olup, eğitim sırasında kullanılmak üzere dörtte bir oranında küçültülerek 1024x768 çözünürlüğüne düşürülmüştür. Ayrıca, veri setini zenginleştirmek amacıyla, geometrik dönüşüm, gürültü ekleme, renk değiştirme, kontrast, parlaklık ve keskinlik değişimleri gibi işlemler uygulanarak 100.000 adet görüntüyle eğitim gerçekleştirilmiştir. Eğitimde, ResNet tabanlı bir model olan RetinaNet kullanılmıştır. 50 epok süren eğitim sonucunda %88,4 doğruluk değeri elde edilmiştir.

Wenqian ve ark. (2020), farklı proje konularından elde edilen bir veri seti üzerinde çalışmışlardır. Bu veri seti toplamda 2024 ham görsele sahiptir ve eritrosit (RBC), lökosit (WBC), EP, CT, CYT, YST ve NC gibi yedi farklı idrar partikülünü içermektedir. Çalışmada, AlexNet, GoogLeNet ve ResNet ağlarının yanı sıra AlexNet mimarisinde yapılan geliştirmelerle oluşturulan bir model olan De-AlexNet de kullanılmıştır. Elde edilen sonuçlar Çizelge 2.2'de sunulmuştur. Ağların temel yapılarıyla yapılan denemelerde, ResNet mimarisinin en doğru sonuçları verdiği gözlemlenmiştir. Ayrıca, sonradan yapılan ince ayarlamalarla ağların doğruluk değerleri artırılmıştır. Sonuçlara bakıldığında yine ResNet mimarisinin en doğru sonuçları verdiği görülmektedir.

Çizelge 2.2. Wenqian ve ark. yaptığı çalışma sonucu (Liu ve ark., 2020)

CNN	Normal	İnce Ayarlı
AlexNet	88.6 ± 1.2	90.2 ± 0.8
De-AlexNet	93.2 ± 0.9	94.8 ± 0.6
GoogLeNet	94.1 ± 0.6	94.6 ± 0.5
ResNet	94.9 ± 0.5	95.3 ± 0.4

Nagai ve ark. (2022), idrar sedimentindeki kristalleri sınıflandırmak amacıyla 441 görüntüden oluşan ve 15 sınıf içeren bir veri seti kullanmışlardır. Bu veri setini ders kitapları ve soru bankalarından toplayarak elde etmişlerdir. Toplanan 441 görüntüyü zenginleştirerek 60.000'e çıkarmışlardır. Veri seti zenginleştirme sürecinde "randomcrop" dışındaki tüm olası işlemleri gerçekleştirdiklerini belirtmişlerdir. Veri seti zenginleştirme işleminin modelin doğruluğu üzerinde olumlu bir etkiye sahip olduğunu göstermişlerdir. CNN mimarilerinden Xception mimarisini kullandıklarını ve yaklaşık %90 doğruluk değeri elde ettiklerini belirtmişlerdir.

Khalid ve ark. (2022), 4 sınıflı (eritrosit, kalsiyum oksalat, sistin kalsiyum ve ürik asit) toplamda 820 idrar sediment görüntüsünden oluşan bir veri seti kullanmışlardır. İdrar sediment görüntülerini sınıflandırmak için MobileNet, VGG16, DenseNet, ResNet50V ve InceptionV3 modellerini tercih etmişlerdir. Yapılan çalışmayla MobileNet modelinin

%98,5 doğruluk oranıyla en iyi model olduğunu belirlemişlerdir. DenseNet ve InceptionV3 modelleri ile %96,5 doğruluk değeri elde etmişler ve MobileNet modelinin de benzer sonuçlar ürettiğini tespit etmişlerdir.

Ji ve ark. (2023), idrar sediment görüntülerini sınıflandırmak için yarı denetimli bir ağ modeli önermişlerdir. Özellikle düşük çözünürlüklü ve sınırlı sayıda idrar sediment görüntüsünü yüksek performansla sınıflandırmayı amaçlamışlardır. Deneylerde, 16 sınıftan oluşan 429.605 idrar sediment görüntüsünü içeren bir veri seti kullanmışlardır. Önerdikleri "US-RepNet" adlı modelle %94 doğruluk değeri elde etmişlerdir.

Atıcı ve ark. (2023), toplamda 9.004 idrar sediment görüntüsünden oluşan ve 7 sınıfı (eritrosit, lökosit, düz epitel, epitel, WBCC, mukus, kabarcık) içeren bir veri seti oluşturmuşlardır. Görüntüler, DIRUI FUS-2000 model mikroskop cihazından elde edilmiş, 800x600 boyutunda ve bmp formatında kaydedilmiştir. Toplanan görüntülerdeki 27.033 hücre uzman biyologlar tarafından işaretlenmiştir. Çalışmada, YOLOv7 modeli kullanılmıştır ve %82 sınıflandırma başarısının elde edildiği bildirilmiştir. Ayrıca, derin öğrenme modellerinde etiketleme sürecinin önemli bir adım olduğu da vurgulanmıştır.

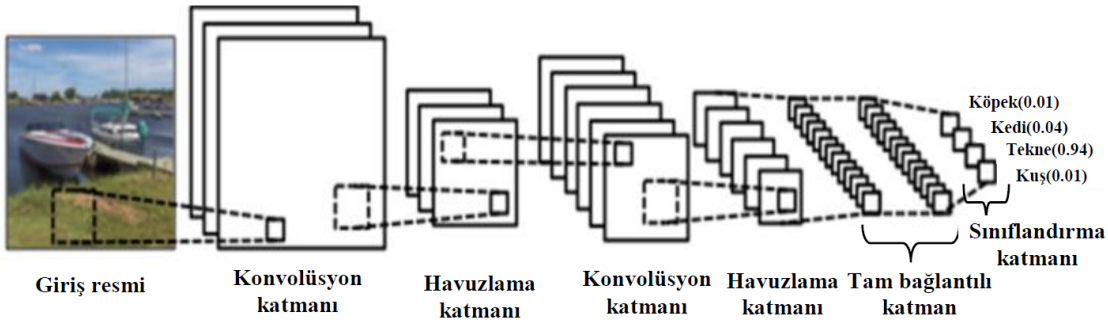
3. MATERYAL VE YÖNTEM

3.1. Evrişimsel Sinir Ağları (ESA)

ESA, çok katmanlı algılayıcının (MLP) özel bir türüdür ve insan görme sisteminin modellenmesiyle oluşturulmuştur. Yaygın olarak bilgisayarlı görme alanında başarılı sonuçlar elde ettiği için geniş çapta kullanılmaktadır. Yann LeCun ve ark. tarafından 1998 yılında yayınlanan bu model, çok katmanlı yapay sinir ağlarının ilk başarılı sonucunu vermiştir. Başlangıçta posta numaraları ve banka çeklerindeki sayıların sınıflandırılması ve okunması için geliştirilmiştir (Aktaş ve ark., 2020). Nesne tanıma, sınıflandırma, takip etme, doğal dil işleme, cümle modelleme ve tahmin problemlerinde kullanılmaktadır (Kızrak ve Bolat, 2018). ESA'da öğrenilecek olan nesnenin ayırt edici özellikleri, programcı tarafından belirlenip ağa bildirilmez. Oluşturulan ESA modeli, öğrenilecek nesnenin ayırt edici özelliklerini kendi kendine öğrenir. Bunun için evrişim katmanı ve havuzlama katmanı gibi özellikler, yapay sinir ağlarından farklı bir şekilde kullanılır (Bayram, 2020).

ESA mimarisinde başlangıçta giriş katmanları konvolüsyon (convolution) ve havuzlama (pooling) katmanlarından oluşurken, daha sonraki gelişmelerle birlikte doğrusal olmayan bir yapı elde etmek için aktivasyon katmanı eklenmiştir. Ardından, ağın ezberlemesini engellemek için daha az ağırlığa sahip düğümlerin kaldırılmasını sağlayan seyreltme (DropOut) katmanı kullanılmıştır. Son aşama, tam bağlantılı katmandan oluşur ve ardından sınıflandırma katmanı yer alır (İnik ve Ülker, 2017; Kızrak ve Bolat, 2018).

Özet olarak, ESA'lar ardışık olarak yerleştirilmiş birden fazla eğitilebilir katmandan oluşur. ESA'ya giriş verildikten sonra, mevcut katmanlarda sırasıyla Şekil 3.1'de gösterilen işlemler gerçekleştirilir ve bu işlemlerin sonunda bir sonuç elde edilir. Eğitim aşamasında elde edilen sonuç ile istenen sonuç arasındaki fark hata olarak ortaya çıkar. Bu hatayı tüm ağırlıklara aktarmak için geriye yayılım algoritması kullanılır. Her bir iterasyonda ağırlıklar güncellenir ve her iterasyonda hata azalır, böylece ağ öğrenmeye başlar (İnik ve Ülker, 2017).



Şekil 3.1. ESA'nın genel mimarisi (İnik ve Ülker, 2017)

3.1.1. Giriş Katmanı

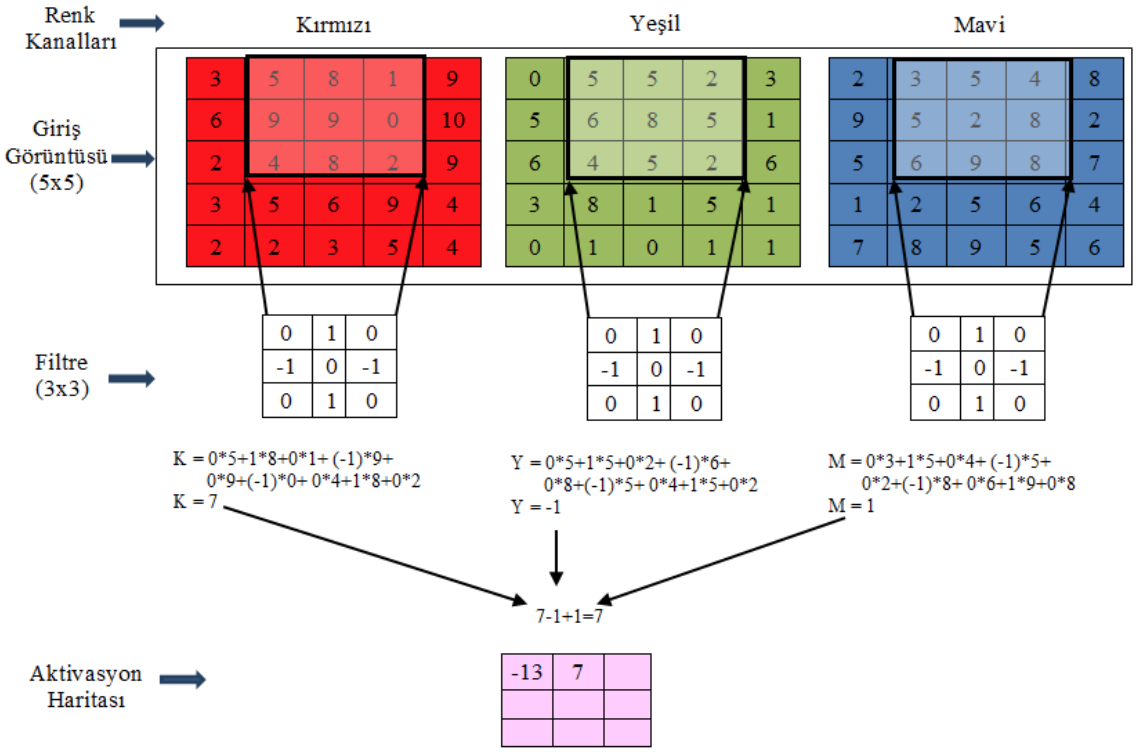
ESA'nın ilk katmanı olan giriş katmanı, verilerin ham haliyle ağa sunulduğu katmandır. Modelin başarısı için bu katman oldukça önemlidir. Giriş katmanında, giriş verisi için boyut tasarımı yapılır. Giriş görüntüsünün yüksek bir boyutta seçilmesi, fazla bellek gerektirir, eğitim ve test süresinin artmasına neden olabilir. Bununla birlikte, ağın başarısını artırabilir. Giriş görüntüsünün düşük bir boyutta seçilmesi, bellek gereksinimini azaltır ve eğitim süresini kısaltır. Ancak, oluşturulan ağın derinliği azalır ve performansı düşebilir. Görüntü analizinde, uygun bir giriş görüntü boyutu seçilmelidir. Bu boyut ağın derinliği, donanımsal hesaplama maliyeti ve ağın başarısı açısında uygun olmalıdır (İnik ve Ülker, 2017).

3.1.2. Konvolüsyon Katmanı (Convolution Layer)

ESA'nın temelini oluşturan konvolüsyon katmanı, dönüşüm katmanı olarak da bilinir. Bu işlem, bir filtre matrisinin belirli bir atlama miktarıyla tüm görüntü üzerinde hareket ettirilmesiyle gerçekleştirilir. Filtreler 2x2, 3x3, 5x5 gibi farklı boyutlarda olabilirler. Bir önceki katmandan gelen görüntülere uygulanırlar ve bu uygulama sonucunda aktivasyon haritalarını oluştururlar. Aktivasyon haritaları, her bir filtrenin özgün özelliklerinin keşfedildiği bölgelerdir. ESA'nın eğitimi sırasında, her iterasyon sonunda filtre katsayıları da değişir. Böylece, görüntü üzerindeki belirleyici noktaların vurgulanması sağlanır (İnik ve Ülker, 2017; Bayram, 2020).

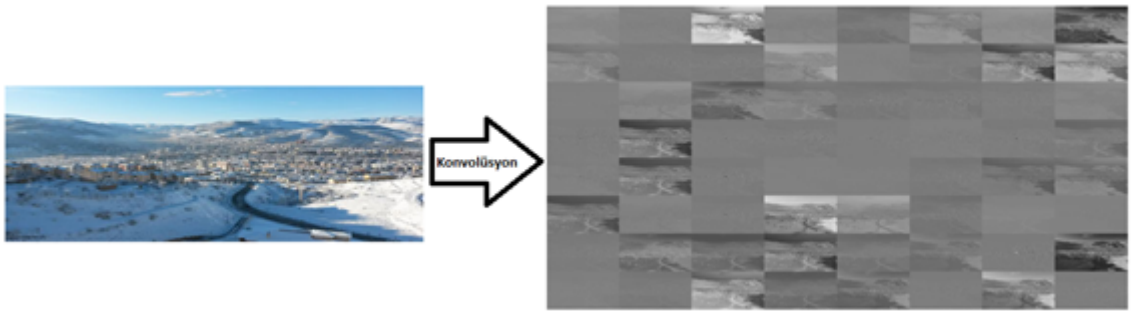
Renkli (RGB) bir görüntüye bir filtrenin uygulanması Şekil 3.2'de gösterilmiştir. Örnekte, giriş katmanının boyutu 5x5x3 olacaktır. Çünkü giriş görüntüsü 5x5 boyutundadır ve renkli olduğundan dolayı kırmızı, mavi ve yeşil renk kanalları da dahil edilerek giriş 5x5x3 boyutunda olur. Filtre örneğinde 3x3 boyutunda bir filtrenin

kullanıldığı görülmektedir. Konvolüsyon işleminde 3x3'lük bir filtre belirli bir adım ile sağa veya sola kaydırılarak görüntü üzerinde dolaştırılır. Örnekte adım sayısı 1 olarak belirlenmiştir. Bu dolaştırma işlemi, görüntü matrisinin tümünde gerçekleştirilir. Filtre katsayıları her bir renk kanalındaki değerlerle çarpılır ve bu değerlerin toplamı alınır. Her üç kanal üzerinde bu işlem gerçekleştirildikten sonra, toplamı aktivasyon haritasını oluşturur. Filtre katsayıları her renk kanalı matrisine uygulandığında farklı olabilir. Tasarımcılar, bu filtre katsayılarındaki değişiklikleri modellerine uygun olarak yaparlar (İnik ve Ülker, 2017).



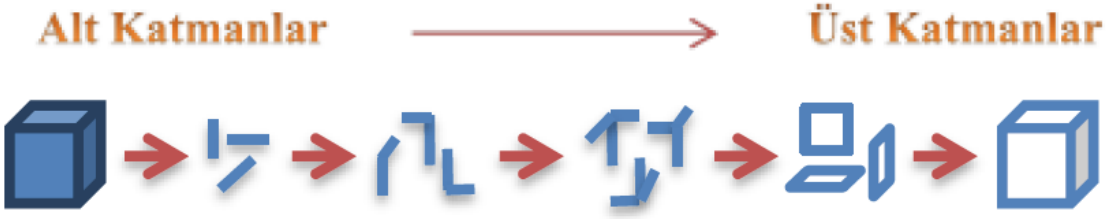
Şekil 3.2. 5x5x3 boyutundaki giriş görüntüsüne 3x3'lük filtrenin uygulandığı konvolüsyon işlemi (İnik ve Ülker, 2017)

Şekil 3.3'te, 3x3 boyutunda ve toplamda 64 filtreden oluşan bir ağ tasarlanmıştır. Her bir filtrenin giriş katmanına uygulanmasıyla oluşan görüntüler de aynı şekilde gösterilmiştir. Şekil 3.3'e bakıldığında, her bir filtrenin giriş görüntüsü üzerinde farklı bir etkisi olduğu görülmektedir (İnik ve Ülker, 2017).



Şekil 3.3. ESA'nın konvolüsyon katmanından sonra oluşan görüntüleri (İnik ve Ülker, 2017)

Birden fazla konvolüsyon katmanı tanımlanarak giriş görüntüsünün boyutu azaltılır ve ağın derinliği artırılır. Bu sayede ağın daha doğru sonuçlar üretmesi sağlanır. Her bir konvolüsyon katmanında, önceki konvolüsyon katmanlarında elde edilen görüntülere filtreler uygulanır. Bu şekilde giriş görüntüsünün en boy oranı azalırken derinlik artar. Artan derinlik sayesinde görüntünün özellikleri daha iyi bir şekilde çıkarılır. Şekil 3.4'te, alt seviye katmanlardan elde edilen özelliklerin daha az ayırt edici olduğu, alt katmanların birleşimiyle oluşan üst seviye katmanların ise daha fazla ayırt edici olduğu görülmektedir (Bayram, 2020).



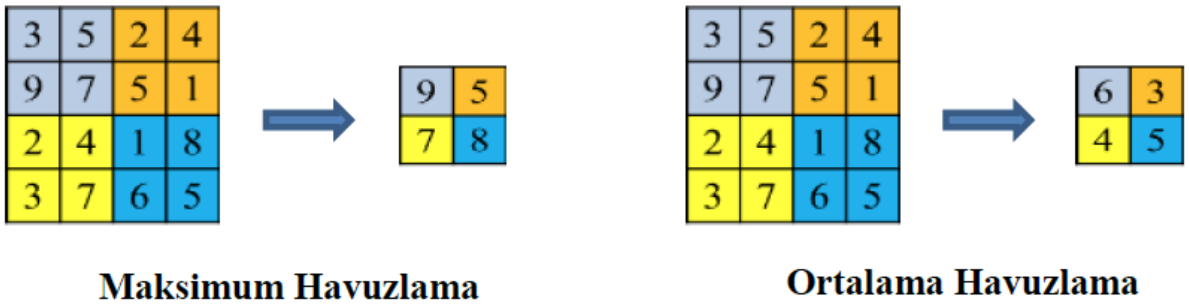
Şekil 3.4. Konvolüsyon işlemi ile katmanlar arasında elde edilen özelliklerin ayırt edicilikleri (Bayram, 2020)

3.1.3. Havuzlama Katmanı (Pooling Layer)

Havuzlama katmanı, genellikle konvolüsyon işleminden sonra gelen bir katmandır. Temel amacı, sonraki konvolüsyon katmanı için giriş boyutunu azaltmaktır. Bu katmanın işlemleri sonucunda veri boyutu azalırken bilgi kaybı oluşur. Genel olarak bu kayıp, ağın performansı için faydalı bir durumdur. Veri boyutundaki azalma, bir sonraki ağ katmanları için daha az hesaplama yükü oluşturur ve modelin ezberlemesini engeller (İnik ve Ülker, 2017; Bayram, 2020).

Havuzlama katmanında, konvolüsyon katmanında olduğu gibi filtreler tanımlanır. Bu filtreler, belirli bir atlama miktarı ile görüntüdeki pikseller üzerinde hareket ettirilir

(Şekil 3.5). Günümüzde maksimum havuzlama ve ortalama havuzlama olmak üzere iki tür filtre kullanılmaktadır. Maksimum havuzlama filtresi, piksellerin maksimum değerini seçer. Ortalama havuzlama filtresi ise piksellerin ortalama değerini hesaplayarak işlem yapar. Maksimum havuzlama yöntemi, günümüzde daha performanslı olduğu için tercih edilmektedir. Havuzlama katmanı, konvolüsyon katmanı sonucunda oluşan tüm görüntülere uygulanır. ESA'lar için havuzlama katmanı isteğe bağlıdır ve bazı mimarilerde bu katman kullanılmaz (İnik ve Ülker, 2017).



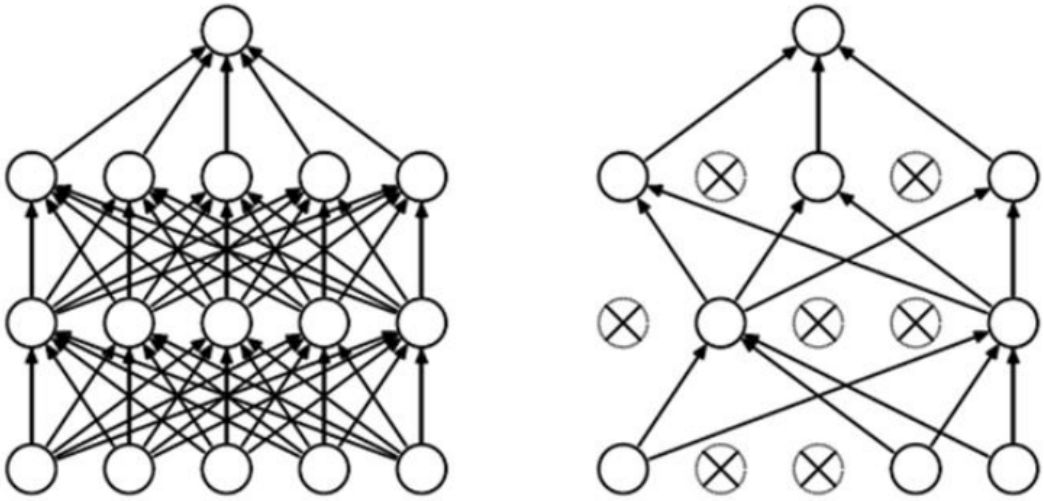
Şekil 3.5. 4x4 boyutundaki görüntüye 2x2 boyutunda maksimum ve ortalama havuzlama operatörünün uygulanması (Bayram, 2020)

3.1.4. Tam Bağlantılı Katman (Fully Connected Layer)

ESA mimarisinde, ardışık konvolüsyon ve havuzlama katmanlarından sonra tam bağlantılı bir katman bulunur. Bu katman, kendisinden önceki katmanın tüm alanlarına bağlıdır. Nöronlar, bu katmanda tam bağlantılı olarak yer alır ve her bir nöron, bir sonraki nöronla bağlantı kurar. Bu nedenle tam bağlantılı katman olarak adlandırılır (İnik ve Ülker, 2017; Doğan ve Türkoğlu, 2018).

3.1.5. Seyreltme Katmanı (DropOut Layer)

Seyreltme katmanı, derin öğrenme katmanlarında aşırı öğrenmeyi ortadan kaldırmak için kullanılan bir katmandır. ESA'da büyük veri setleriyle eğitim yapıldığından, ağ bazen ezberleme yapabilir. Bu katmanda, aşırı öğrenmeye neden olan bazı bağlantılar ortadan kaldırılır. Bu şekilde ağın ezber yapması engellenir ve performansı artırılır. Şekil 3.6a'da ağın orijinal yapısı, Şekil 3.6b'de ise seyreltme katmanından sonra oluşan yapı görülmektedir (İnik ve Ülker, 2017; Doğan ve Türkoğlu, 2018).



Şekil 3.6. Standart bir ESA'ya seyreltme katmanının uygulanması (İnik ve Ülker, 2017)

3.1.6. Sınıflandırma Katmanı (Classification Layer)

Sınıflandırma katmanı, tam bağlantılı katmandan sonra gelir. Bu katman, tam bağlantılı katmandan gelen değerleri alarak sınıflandırma işlemini gerçekleştirir. Katmanın çıkışında olasılıksal değerler üretilir. Bu katmanın çıkış sayısı, sınıflandırılacak nesne sayısına eşittir. Sınıflandırma yapılırken, her sınıfa olan yakınlık değeri üretilir. Derin öğrenme ağında, bu katmanda farklı sınıflandırıcılar kullanılabilir. Genellikle Softmax sınıflandırıcı tercih edilir, çünkü başarıyla çalışmaktadır. Sınıflandırmada, her bir nesne için 0-1 aralığında bir çıkış üretilir. Çıkış değeri 1'e yakın olan sınıf, ağın tahmin ettiği sınıf olarak belirlenir (İnik ve Ülker, 2017; Doğan ve Türkoğlu, 2018).

3.2. YOLO (You Only Look Once)

Günümüzde, derin öğrenme yöntemiyle nesne tespiti için kullanılan algoritmalar iki gruba ayrılabilir. İlk grup R-CNN (Girshick ve ark., 2013), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren ve ark., 2017) ve Mask R-CNN (He ve ark., 2017) gibi modelleri içermektedir. Bu modeller, nesne tespiti için iki aşamalı bir yaklaşım benimsemektedirler. İki aşamalı yapıları sebebiyle yavaş çalışırlar, ancak yüksek doğruluk oranına sahiptirler. Bu yöntemler, diğerlerine kıyasla daha fazla zaman ve

karmaşık bir yapı gerektirirler. Buna karşılık, SSD (Tek Seferde Çoklu Kutu Tespit Edici) (Shafi ve ark., 2022), YOLO (Sadece Bir Kez Bak) (Redmon ve ark., 2016) ve RetinaNet (Lin ve ark., 2017) gibi derin öğrenme modelleri, görüntüyü tek seferde işleyerek sonuç üreten modellerdir. Bu modeller, nesne tespit problemini bir regresyon problemi gibi ele alarak tek seferde sonuç üretebilmektedirler.

YOLO'nun (Redmon ve ark., 2016) ilk sürümü 2016 yılında Joseph Redmon ve ekibi tarafından geliştirilmiştir. PASCAL VOC 2012 veri setinde gerçek zamanlı nesne algılama için en iyi performansı sağlayan bir model olmuştur. YOLOv1, AlexNet tabanlı bir model olan GoogLeNet ESA'yı kullanmaktadır. YOLOv1, her görüntüyü 3x3, 5x5 gibi boyutlarda sınırlayıcı kutulara bölmektedir. Her sınırlayıcı kutu için B adet (YOLOv1'de B, 2 olarak belirlenmiştir) destek kutusu çizilmektedir. Çizilen destek kutularının güven skorları belirlenmektedir. Ardından sınıf olasılıkları hesaplanmaktadır (Redmon ve ark., 2016; Nazir ve Wani, 2023; Ye ve ark., 2023).

YOLOv2 (Redmon ve Farhadi, 2017), YOLO9000 olarak da bilinen bir modeldir. Joseph Redmon ve ekibi tarafından 2017 yılında geliştirilmiştir. Bu yeni model, YOLOv1'in üzerine inşa edilmiştir. YOLOv2, daha yüksek doğruluk oranına ve daha hızlı çalışma yeteneğine sahiptir. Yeni modelde küçük nesnelerin tespitinde doğruluk oranı artırılmıştır. YOLOv1'de görüntüler sınırlayıcı kutulara bölünüp daha sonra destek kutuları belirleniyordu. Ancak YOLOv1'de destek kutu sayısı sınırlıydı, bu da her sınırlayıcı kutuda en fazla iki nesne tespit edilebileceği anlamına geliyordu. YOLOv2'de, bu soruna çözüm olarak daha esnek destek kutuları tasarlanmıştır. Bu sayede her sınırlayıcı kutu ile en fazla iki nesne tespit edebilme sorununa çözüm getirilmiştir. YOLOv2'de bir dizi iyileştirme yapılmış ve daha güçlü ve derin bir ESA olan Darknet-19 kullanılmıştır (Redmon ve Farhadi, 2017; Nazir ve Wani, 2023; Ye ve ark., 2023).

YOLOv3 (Redmon ve Farhadi, 2018), 2018 yılında Joseph Redmon ve ekibi tarafından geliştirilmiştir. YOLOv3'e yeni eklenen 53 katmanla toplamda 106 katmana ulaşılmıştır. Ayrıca YOLOv3'te Darknet-19'a kıyasla daha büyük ve daha derin olan Darknet-53 ağı kullanılmıştır. Darknet-53, Darknet-19'dan daha güçlü ve ResNet-101 ve ResNet-152'den daha verimlidir. YOLOv3 modeli, eski modellere göre daha derin ve karmaşık bir ağ yapısına sahiptir. Bu nedenle eğitim süresi daha uzun sürmekte ve daha fazla veri kümesiyle eğitilmektedir. YOLOv3'ün en belirgin özelliği, üç farklı ölçekte tespit yapabilmesidir. Bu sayede hem büyük hem de küçük nesnelerin tespiti konusunda iyi bir performans sergilemektedir (Redmon ve Farhadi, 2018; Aktaş ve ark., 2020; Yasak, 2021; Dang ve ark., 2023; Nazir ve Wani, 2023).

YOLOv4 (Bochkovskiy ve ark., 2020), Alexey Bochkovskiy ve ekibi tarafından 2020 yılında geliştirilmiştir. COCO veri setinde en yüksek başarıya sahip gerçek zamanlı bir nesne tespit modelidir. YOLOv3'e kıyasla, ortalama hassasiyet %10 ve fps (frame per second) %12 artırılmıştır. YOLOv4'ün getirdiği yenilikler arasında, eğitim sırasında veri setinin zenginleştirilmesine olanak tanıyan "bag of speacials (BoS)" ve "bag of freebies (BoF)" bölümlerinin modelin omurga bölümüne eklenmesi bulunmaktadır. Yeni modelde, Darknet-53'ten CSPDarknet-53 omurgasına geçilmiştir. Ayrıca, modelde CSP (Cross Stage Partial Network), SPP (Spatial Pyramid Pooling), SAM (Spatial Attention Module), CmBN (Cross Mini-Batch Normalization) ve PAN (Path Aggregation Network) gibi değişiklikler ve yenilikler yapılmıştır (Bochkovskiy ve ark., 2020; Dang ve ark., 2023; Nazir ve Wani, 2023; Ye ve ark., 2023).

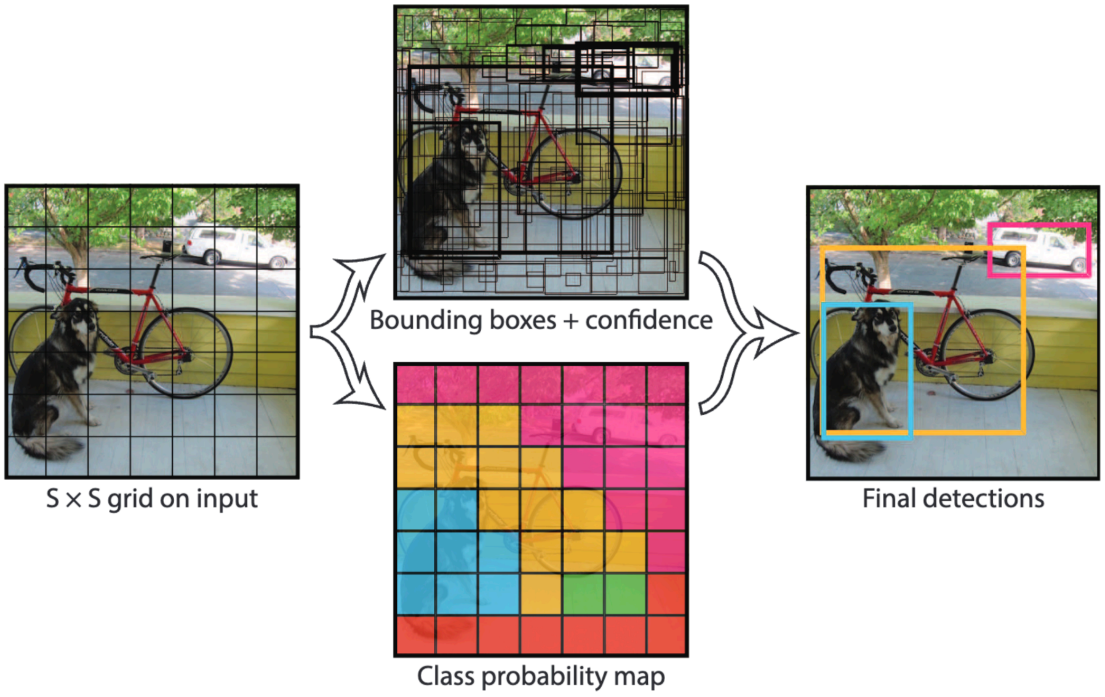
YOLOv5 (Glenn, 2020), YOLOv4'ün çıkışından yaklaşık bir ay sonra ortaya çıkmıştır. Yakın zamanda ortaya çıkmaları ve YOLOv5'in hakemli bir makaleye sahip olmaması, birçok tartışmaya yol açmıştır. Ancak YOLOv5, yenilikler getirmiş ve topluluğun güvenini kazanmıştır. YOLOv5'te, önceki sürümlerde kullanılan C programlama diliyle yazılan Darknet kütüphanesi yerine Python programlama diliyle yazılan PyTorch kütüphanesi kullanılmıştır. Python dilinin kullanım kolaylığı, geniş topluluk desteği, kurulum ve entegrasyon kolaylığı gibi avantajlarıyla popüler bir programlama dilidir. PyTorch kütüphanesiyle birlikte YOLOv5'in gelecekte daha fazla katkı ve büyüme potansiyeline sahip olması beklenmektedir. İki modelin performans karşılaştırmasının doğruluğu hakkında tartışmalar bulunmaktadır. Çünkü iki model farklı diller ve kütüphaneler kullanılarak tasarlanmıştır. Dil ve kütüphane kullanımının performans üzerindeki etkileri doğrudan hesaplanamadığından, adil bir karşılaştırma yapmak mümkün olmamaktadır. Ancak yapılan çalışmalarda, YOLOv5'in YOLOv4'e göre daha yüksek performans gösterdiği gözlemlenmiştir (Glenn, 2020; Thuan, 2021; Parlak ve Emel, 2022; Nazir ve Wani, 2023).

YOLOv6 (Li ve ark., 2022), Chuyi Li ve ekibi tarafından 2022 yılında geliştirilmiştir. YOLOv6, hız ve doğruluk gibi metrikler açısından oldukça iyi bir performans sergilemektedir. YOLOv6'ya, nicelleştirme teknikleri olan QAT (quantization-aware training) ve PTQ (post-training quantization) entegre edilerek performans artırılmış ve çıkarım hızı önemli ölçüde düşürülmeden elde edilmiştir (Li ve ark., 2022; Nazir ve Wani, 2023).

3.2.1. YOLOv7

Bu çalışmada kullanılan YOLOv7 (Wang ve ark., 2023) modeli, 2022 yılında geliştirilmiştir. YOLOv7 ve YOLOv6 modelleri benzer bir zamanda piyasaya sürülmüşlerdir. YOLOv7'nin yeni modeli, hız ve performans iyileştirmeleri sunmaktadır. YOLOv7, son derece gelişmiş ve doğru nesne tespit performansı sunarak hesaplama ve çıkarım maliyetlerini artırmadan önceki iyi bilinen nesne tespit algoritmalarını etkin bir şekilde geride bırakmaktadır. Bu, daha hızlı çıkarım işlemleriyle birlikte daha yüksek tespit doğruluğunu mümkün kılar (Gallo ve ark., 2023). Bu bölümde, YOLO algoritması detaylandırılarak açıklanacaktır.

YOLO modeli, giriş görüntüsünü Şekil 3.7'de gösterildiği gibi SxS boyutundaki (örneğin 7x7, 13x13, 26x26) sınırlayıcı kutulara bölmektedir. Giriş görüntüsü sınırlayıcı kutulara ayrıldıktan sonra her bir sınırlayıcı kutu, nesnelerin merkezinde olup olmadığını bulmakla sorumludur. Eğer herhangi bir nesnenin merkezi, kendi sınırları içerisinde ise bu nesneleri içerecek şekilde destek kutuları çizilir.



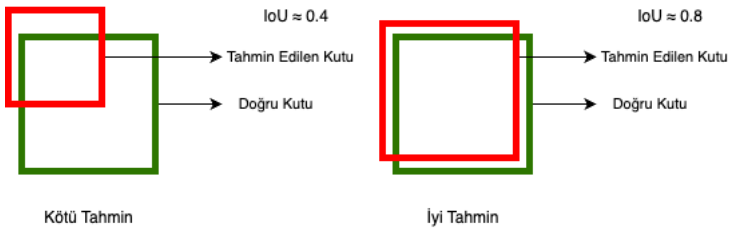
Şekil 3.7. Giriş görüntüsüne uygulanan sınırlayıcı kutu işlemi (Redmon ve ark., 2016)

Her çizilen destek kutusu için bir güven skoru hesaplanır. Hesaplanan bu güven skoruna göre fazladan çizilen destek kutuları elenir ve her nesne için sadece bir tane

destek kutusu kalması sağlanır. Denklem 3.1'de güven skorunun (GS) hesaplanmasında kullanılan formül verilmiştir:

$$GS = \Pr(nesne) * IoU_{tahmin}^{gerçek} \tag{3.1}$$

Denklem 3.1'deki Pr(nesne) destek kutusunda nesne bulunma olasılığını 0 ile 1 arasında bir değer ile ifade etmektedir. Denklemdeki bir diğer değişken olan $IoU_{tahmin}^{gerçek}$ ise destek kutusunun gerçek kutu arasındaki birleşim üzerinden kesişim değerini ifade etmektedir (Şekil 3.8).



Şekil 3.8. IoU değeri hesabı

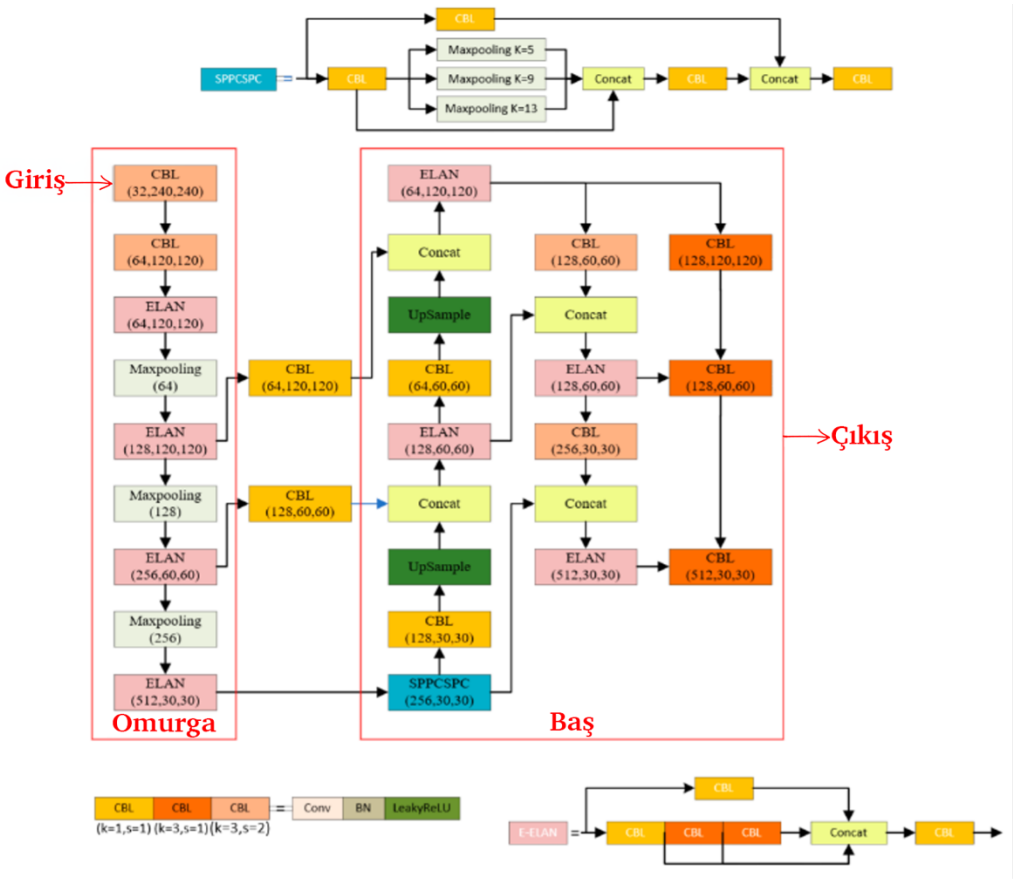
Her destek kutusu için toplamda 5 parametre hesaplanır. Bu parametreler sırasıyla x, y, w, h ve güven skorudur ve hepsi 0 ile 1 arasındadır. x ve y değerleri, nesnenin merkez konumunu ifade etmektedir. w ve h değerleri ise kutunun yüksekliğini ve genişliğini belirtmektedir. Güven skoru ise Denklem 3.1'deki denkleme eşittir ve tahmin edilen destek kutusunun güvenirliğini ifade eder. Her destek kutusu yalnızca bir nesneye aittir.

3.2.1.1. YOLOv7-tiny Modeli

YOLOv7'de birden fazla alt model bulunmaktadır. Bu alt modelleri az karmaşık modelden daha karmaşık modele doğru YOLOv7-tiny, YOLOv7, YOLOv7-X, YOLOv7-W6, YOLOv7-E6, YOLOv7-E6E ve YOLOv7-D6 şeklinde sıralayabiliriz. Bu çalışmada kullanılan YOLOv7-tiny modeli aşağıda detaylı olarak açıklanmıştır.

YOLOv7-tiny modeli 3 ana bölümden oluşmaktadır (Şekil 3.9). Bu bölümler sırasıyla giriş (input), omurga (backbone) ve baş (head) bölümleridir. Giriş bölümünde görsel alınır ve yeniden boyutlandırılır. Ayrıca veri zenginleştirme işlemleri de bu bölümde gerçekleştirilir. Omurga bölümü genellikle nesneye ait özniteliklerin çıkarıldığı bölümdür. Bu bölümde çıkarılan özniteliklerin kaybolmaması, hesaplama maliyetlerinin

düşürülmesi ve ağın daha fazla öznitelik öğrenmesi için 2 adet ara bağlantı kullanılarak baş bölümüne bilgi aktarımı sağlanmıştır. Son olarak baş bölümünde destek kutusu, güven skoru ve nesne olasılığı gibi işlemler gerçekleştirilir (Li ve ark., 2023).



Şekil 3.9. YOLOv7-tiny mimarisi (Li ve ark., 2023)

Şekil 3.9'da CBL adı verilen bu katman, görüntülerden özellik haritaları çıkarmak için kullanılır. ELAN katmanı, aktivasyon işlemlerini zenginleştirmek için genişletilmiş bir lineer ağ olarak görev yapar. Concat katmanı, farklı ölçeklerdeki bilgileri birleştirerek daha kapsamlı bir nesne tespiti yapmak amacıyla özellik haritalarını bir araya getirir. Upsample katmanı, özellik haritalarını boyutlandırmak için kullanılır ve daha küçük özellik haritalarını büyüterek nesne tespit hassasiyetini artırır. Bu katmanlar, YOLOv7-tiny modelinin daha hassas ve etkili nesne tespiti yapabilmesi için özel bir mimari oluşturur.

YOLOv7-tiny algoritması, YOLOv7'den türetilmiş olup, kademe tabanlı model ölçeklendirme stratejisini korur. Daha az parametre ve daha hızlı tespit hızıyla geliştirilmiş verimli uzun menzilli toplama ağı (ELAN) kullanarak tespit doğruluğunu iyileştirir. YOLOv7 ailesinin performans karşılaştırması Çizelge 3.1'de verilmiştir.

Çizelge 3.1. YOLOv7 ailesinin performans karşılaştırması

Model	Param.	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 (Bochkovskiy ve ark., 2020)	64.4M	640	%49.7	%68.2	%54.3	%32.9	%54.8	%63.7
YOLOR-u5 (r6.1) (Wang, Yeh, ve ark., 2021)	46.5M	640	%50.2	%68.7	%54.6	%33.2	%55.5	%63.7
YOLOv4-CSP (Wang, Bochkovskiy, ve ark., 2021)	52.9M	640	%50.3	%68.6	%54.9	%34.2	%55.6	%65.1
YOLOR-CSP (Wang, Yeh, ve ark., 2021)	52.9M	640	%50.8	%69.5	%55.3	%33.7	%56.0	%65.4
YOLOv7 (Wang ve ark., 2023)	36.9M	640	**%51.2**	**%69.7**	**%55.5**	**%35.2**	**%56.0**	**%66.7**
	%-43	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X (Wang, Yeh, ve ark., 2021)	96.9M	640	%52.7	**%71.3**	%57.4	%36.3	%57.5	%68.3
YOLOv7-X (Wang ve ark., 2023)	71.3M	640	**%52.9**	%71.1	**%57.5**	**%36.9**	**%57.7**	**%68.6**
	%-36	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny (Wang, Bochkovskiy, ve ark., 2021)	6.1	416	%24.9	%42.1	%25.7	%8.7	%28.4	%39.2
YOLOv7-tiny (Wang ve ark., 2023)	6.2	416	**%35.2**	**%52.8**	**%37.3**	**%15.7**	**%38.0**	**%53.4**
	%+2	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l (Wang, Bochkovskiy, ve ark., 2021)	8.7	320	%30.8	%47.3	%32.2	**%10.9**	%31.9	%51.5
YOLOv7-tiny (Wang ve ark., 2023)	6.2	320	**%30.8**	**%47.3**	**%32.2**	%10.0	**%31.9**	**%52.2**
	%-39	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 (Wang, Yeh, ve ark., 2021)	115.8M	1280	%55.7	%73.2	%60.7	%40.1	**%60.4**	%69.2
YOLOv7-E6 (Wang ve ark., 2023)	97.2M	1280	**%55.9**	**%73.5**	**%61.1**	**%40.6**	%60.3	**%70.0**
	%-19	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 (Wang, Bochkovskiy, ve ark., 2021)	151.7M	1280	%56.1	%73.9	%61.2	**%42.4**	%60.5	%69.9
YOLOv7-D6 (Wang ve ark., 2023)	154.7M	1280	%56.3	%73.8	%61.4	%41.3	%60.6	%70.1
YOLOv7-E6E (Wang ve ark., 2023)	151.7M	1280	**%56.8**	**%74.4**	**%62.1**	%40.8	**%62.1**	**%70.6**
	=	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

3.2.1.2. Performans Ölçütleri

Bir nesne algılama modelinin performansını değerlendirmek için birçok ölçüt kullanılır. En yaygın kullanılan ölçütlerden bazıları IoU (Intersection over Union), hassasiyet (Precision), duyarlılık (Recall), ortalama hassasiyet (Average Precision - AP) ve ortalama kesinlik değerlerinin ortalaması (mean Average Precision - mAP) olarak verilebilir.

IoU: IoU, gerçek değer (Ground Truth) ve model tahmini arasındaki benzerliği ölçen bir değerlendirme metriğidir. IoU, nesne algılama performansının doğruluğunu değerlendirmek için yaygın olarak kullanılan bir ölçüttür. Daha yüksek IoU değerleri, modelin gerçek nesneyi daha iyi tahmin ettiğini gösterir. Bu metrik, Denklem 3.2'de gibi hesaplanır.

$$IoU = \frac{Alan(Gerçek\ \cap \text{Tahmin})}{Alan(Gerçek\ \cup \text{Tahmin})} \tag{3.2}$$

Hassasiyet (Precision): Hassasiyet, doğru olarak tahmin edilen pozitif örneklerin, tüm tahmin edilen pozitif örnekler üzerindeki oranıyla hesaplanır. Burada, doğru pozitif, gerçek pozitif örneklerin doğru bir şekilde tahmin edildiği durumları temsil eder. Yanlış pozitif ise gerçek negatif örneklerin yanlış bir şekilde pozitif olarak tahmin edildiği durumları temsil eder. Hassasiyet, bir modelin ne kadar kesin olduğunu ölçen bir metriktir. Yani, tahmin edilen pozitif örneklerin gerçekten pozitif olma olasılığını gösterir. Hassasiyet değeri ne kadar yüksekse, modelin pozitif tahminlerinin ne kadar doğru olduğunu gösterir. Bu metrik Denklem 3.3'te verildiği gibi hesaplanır.

$$Hassasiyet = \frac{Doğru\ Pozitif}{(Doğru\ Pozitif + Yanlış\ Pozitif)} \tag{3.3}$$

Duyarlılık (Recall): Duyarlılık gerçek sınıftaki tüm örneklerin içinde doğru olarak tahmin edilen pozitif örneklerin oranıyla elde edilir. Burada, doğru pozitif, gerçek pozitif örneklerin doğru bir şekilde tahmin edildiği durumları temsil eder. Yanlış negatif ise gerçek pozitif örneklerin yanlış bir şekilde negatif olarak tahmin edildiği durumları temsil eder. Duyarlılık, bir modelin ne kadar eksiksiz olduğunu ölçen bir metriktir. Yani, gerçek pozitif örneklerin ne kadarının doğru bir şekilde tespit edilebildiğini gösterir.

Duyarlılık değerinin yüksek olması modelin gerçek pozitifleri ne kadar iyi tespit ettiğini gösterir. Bu metrik Denklem 3.4'te verildiği gibi hesaplanır.

$$Duyarlılık = \frac{Doğru\ Pozitif}{(Doğru\ Pozitif + Yanlış\ Negatif)} \tag{3.4}$$

Ortalama Hassasiyet (AP): AP, nesne algılama performansını değerlendirmek için kullanılan hassasiyet ve duyarlılık metriklerini içeren bir ölçüttür. AP, Hassasiyet-Duyarlılık eğrisini özetleyen ve 0 ile 1 arasındaki duyarlılık değerlerinin ortalamasını içeren bir sayısal metriktir. AP, bir modelin nesne algılama performansını genel olarak özetler. Yüksek AP değerleri, modelin hem yüksek hassasiyet hem de yüksek duyarlılık sergilediğini gösterir. AP değerinin yüksekliği, modelin nesneleri doğru bir şekilde tespit etme ve doğru tahminleme yeteneğinin o kadar yüksek olduğunu gösterir. Bu metrik Denklem 3.5'te verildiği gibi hesaplanır.

$$AP = \sum \big(Duyarlılık(i) - Duyarlılık(i-1)\big) * Hassasiyet(i) \tag{3.5}$$

Ortalama Kesinlik Değerlerinin Ortalaması (mAP): mAP değeri, her sınıfın ortalama hassasiyet değerlerinin toplanması ve sınıf sayısına bölünmesiyle elde edilir. mAP değeri, genel bir performans ölçütü olarak kullanılır ve bir modelin nesne algılama yeteneğini sınıflar arasında karşılaştırmak için kullanılır. Yüksek mAP değerleri, modelin farklı sınıfları başarılı bir şekilde tespit etme yeteneğini gösterir. Bu metrik Denklem 3.6'da verildiği gibi hesaplanır.

$$mAP = \frac{1}{S}\sum_{j=1}^{S} AP_j \tag{3.6}$$

3.3. Python Programlama Dili

Python, 1990 yılında Hollandalı programcı Guido Van Rossum tarafından geliştirilmeye başlanmış bir programlama dilidir. Bağımsız bir platforma sahip yorumlanabilir ve interaktif bir üst seviye dildir. Diğer dillerden farklı olarak derleyiciye ihtiyaç duymadan doğrudan yorumlanabilir. Böylece derleme sürecine ihtiyaç

duyulmadan farklı işletim sistemlerinde ve farklı mimarilerde sorunsuz bir şekilde çalışan yazılımlar geliştirilebilir.

Python'un önemli bir özelliği, açık kaynak kodlu olmasıdır. Bu özellik sayesinde Python sürekli olarak geliştirilir ve geniş bir kullanıcı topluluğu tarafından desteklenir. Açık kaynak yapısı, herhangi bir kullanıcının Python'un kaynak kodunu inceleyebilmesini ve gerektiğinde değişiklik yapabilmesini mümkün kılar.

Python, sade bir yazım kuralına sahip olan dinamik bir programlama dilidir. Girintilere dayalı sözdizimi, kodun okunabilirliğini artırır ve dilin öğrenilmesini kolaylaştırır. Python'un modüler yapısı, sınıf kullanımını destekler ve her türlü veri girişini kolaylaştırır. Bu da programlama sürecini daha verimli hale getirir.

Python, farklı platformlarda (Linux, Mac, Windows vb.) çalışabilme yeteneğine sahiptir. Bu, Python'un geniş bir kullanıcı tabanına ulaşmasını ve çeşitli projelerde kullanılmasını sağlar. Python, masaüstü uygulamaları, web uygulamaları, bilimsel ve matematiksel hesaplamalar gibi çeşitli uygulama alanlarında kullanılabilir.

Python'un geniş bir geliştirici topluluğu ve geniş bir kütüphane desteği bulunmaktadır. Bu, Python programlama dili ile çeşitli uygulamaların geliştirilebileceği anlamına gelir. İhtiyaç duyulan kütüphaneler, "PIP" adı verilen paket yöneticisi aracılığıyla kolayca yönetilebilir.

3.4. PyTorch Kütüphanesi

PyTorch, Facebook Yapay Zekâ Araştırma Laboratuvarı FAIR (Facebook Artificial Intelligence Researchers) tarafından geliştirilen, açık kaynaklı bir makine öğrenmesi kütüphanesidir. PyTorch, sinyal işleme, makine öğrenmesi ve görüntü işleme konularında hazır algoritmalar sunmanın yanı sıra derin öğrenme ve katmanlı sinir ağlarının modellenmesi konusunda önemli bir altyapı sağlar. Esnek ve kolay kullanım sunan kütüphanelere sahip olması, kompleks sinir ağlarının modellemesini mümkün kılar. PyTorch, CPU ve GPU'lar arasında etkili bir şekilde paralel işlem yapabilir.

PyTorch'un yüksek performans elde etmek için büyük bir kısmı C++ diliyle yazılmıştır. Ancak farklı bir dilde yazılmış olmasına rağmen, Python ekosistemiyle sıkı bir entegrasyona sahiptir. PyTorch'un temel libtorch kütüphanesi, tensör veri yapısını, GPU ve CPU operatörlerini ve temel paralel işlemleri içerir. PyTorch kütüphanesinin diğer kütüphanelerle karşılaştırması Çizelge 3.2'de sunulmuştur (Paszke ve ark., 2019).

Çizelge 3.2. Derin öğrenme modellerinin kütüphaneler üzerindeki verimlilikleri (yüksek değer daha iyi)

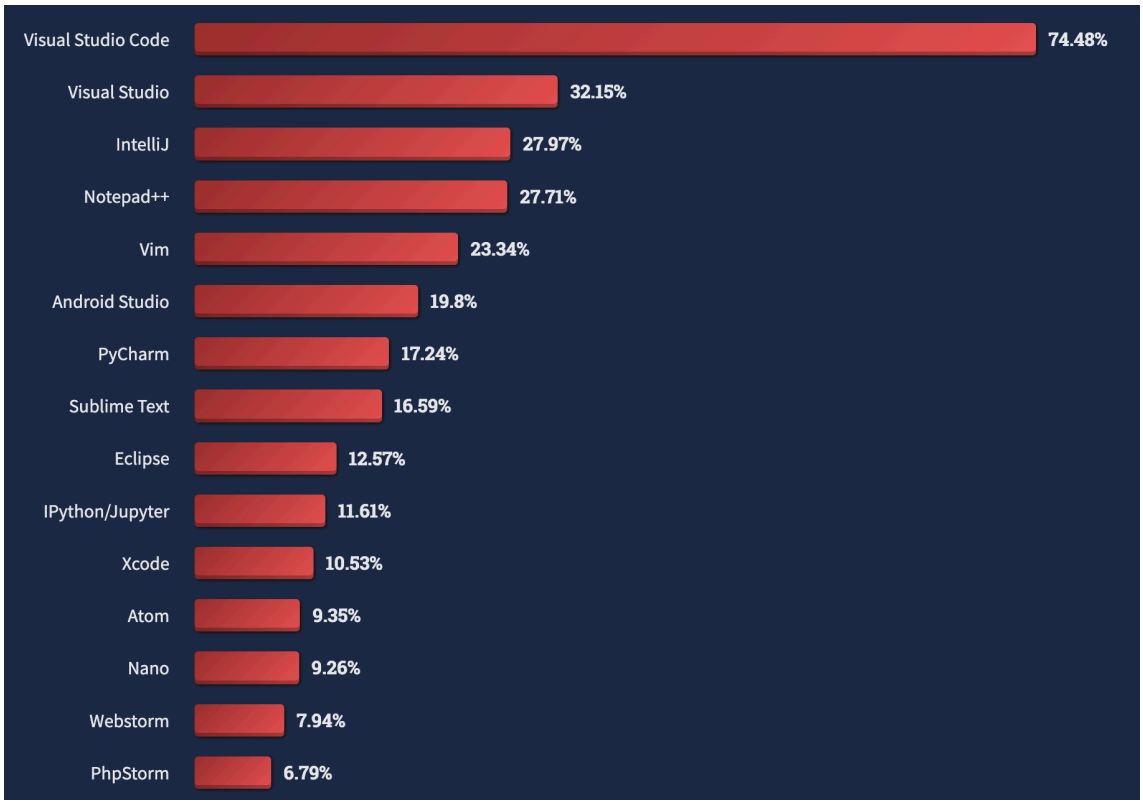
Framework	AlexNet	VGG-19	ResNet-50	MobileNet	GNMTv2	NCF
Chainer	778 ± 15	N/A	**219** ± 1	N/A	N/A	N/A
CNTK	845 ± 8	84 ± 3	210 ± 1	N/A	N/A	N/A
MXNet	**1554** ± 22	113 ± 1	218 ± 2	444 ± 2	N/A	N/A
PaddlePaddle	933 ± 123	112 ± 2	192 ± 4	**557** ± 24	N/A	N/A
TensorFlow	1422 ± 27	66 ± 2	200 ± 1	216 ± 15	9631 ± 1.3%	4.8e6 ± 2.9%
PyTorch	1547 ± 316	**119** ± 1	212 ± 2	463 ± 17	**15512** ± 4.8%	**5.4e6** ± 3.4%

Bu çalışmada, YOLOv7 kütüphanesinin bir bağımlılığı olarak PyTorch'un 1.10 sürümü kullanılmıştır.

3.5. Visual Studio Code

Visual Studio Code (VSC), hafif ancak güçlü bir metin editörü olarak işlev gören bir masaüstü uygulamasıdır. Windows, macOS ve Linux işletim sistemlerinde kullanılabilir. Tam özellikli bir IDE'ye kıyasla daha minimalist bir arayüz sunar. Ancak kullanıcı taleplerine göre hata ayıklama işlevleri, Git entegrasyonu gibi birçok özellik eklenerek genişletilebilir.

VSC metin editörü, platformdan bağımsız masaüstü uygulamaları oluşturmak için açık kaynaklı Electron kütüphanesi kullanılarak geliştirilmiştir. Performans açısından Sublime ve Atom gibi metin editörlerinden daha üstün olduğu belirtilmektedir. 2022 Stackoverflow anketine göre (Şekil 3.10), kullanıcılar tarafından en çok tercih edilen araçlar arasında yer almaktadır.



Şekil 3.10. Stackoverflow anket sonuçlarına göre en çok kullanılan IDE (*Stack Overflow Developer Survey 2022*, 2023)

VSC, birçok popüler programlama dilini destekler. Python, C++, C#, JavaScript, TypeScript, HTML, CSS, Go, Ruby, Rust gibi dillerle uyumlu olarak çalışabilir. Aynı zamanda geliştiricilere zengin özelliklere sahip bir kod düzenleme deneyimi sunar. Otomatik kod tamamlama, kod renklendirme, hata kontrolü ve hızlı gezinme gibi özellikler, geliştiricilerin işlerini kolaylaştırır.

Uzantı ekosistemi, VSC'nin işlevselliğini genişletmek için kullanılabilir. Hazır uzantılar aracılığıyla farklı amaçlara yönelik özellikler eklenirken, kullanıcılar istedikleri özellikleri ekleyebilir veya geliştirebilirler.

VSC'nin bütünleşmiş terminalleri, uygulamaların çalıştırılmasını, hata ayıklamayı ve komutları doğrudan editörden çalıştırmayı mümkün kılar. Ayrıca Git entegrasyonu sayesinde kod depolama ve sürüm kontrolü kolaylaşır. Commit'ler, dal birleştirme işlemleri ve diğer Git işlemleri doğrudan VSC üzerinden gerçekleştirilebilir.

VSC, birçok kullanışlı araç ve eklentiyle birlikte gelir. Bütünleşmiş terminal, otomatik kaydetme, çoklu imleç desteği, görev çalıştırıcıları gibi özellikler, geliştiricilere daha verimli bir çalışma ortamı sunar.

Tüm bu özellikleriyle VSC, geliştiriciler arasında popüler bir seçenek haline gelmiştir. Hem hafif olması hem de genişletilebilir olması, farklı programlama dilleriyle çalışan ve farklı gereksinimlere sahip kullanıcılar tarafından tercih edilir.

Bu çalışmada, bütün kodlama işlemleri için VSC çalışma ortamı kullanılmıştır. Kodlar Python dilinde yazılmıştır.

3.6. Docker

Docker, yazılım geliştirme ve dağıtım süreçlerini hızlandırmak, standartlaştırmak ve daha verimli hale getirmek amacıyla kullanılan bir konteynerizasyon platformudur. Konteynerizasyon, uygulamaların ve bağımlılıklarının bir araya getirilip izole edilerek, farklı ortamlarda sorunsuz bir şekilde çalışabilmesini sağlayan bir teknoloji olarak tanımlanabilir. Docker, bu kavramı etkili bir şekilde uygulayan ve yaygınlaştıran bir aracıdır.

Docker, üç temel bileşenden oluşur: Docker Motoru (Docker Engine), Docker Hub ve Docker Compose. Docker Motoru, işletim sistemi düzeyinde sanallaştırma teknolojilerini kullanarak, konteynerlerin oluşturulmasını, yönetilmesini ve çalıştırılmasını sağlayan çekirdek bileşenidir. Docker Motoru, izolasyon özellikleri sayesinde konteynerlerin birbirinden bağımsız olarak çalışmasını sağlar.

Docker Hub, Docker konteynerlerinin paylaşılması, dağıtılması ve bulunabilirliği için merkezi bir depo ve kaynak sunar. Docker Hub, kullanıcıların hazır konteyner görüntülerini paylaşmalarını ve kullanmalarını kolaylaştırır.

Docker Compose, çoklu konteyner uygulamalarını tanımlamak ve yönetmek için kullanılan bir araçtır. Docker Compose dosyaları, birden fazla konteynerin nasıl çalıştırılacağını, bağlantılarını ve yapılandırmalarını tanımlar. Bu, karmaşık uygulamaların kolayca oluşturulmasını ve yönetilmesini sağlar.

Docker, yazılım geliştirme süreçlerinde tekrarlanabilirliği artırır, uygulama bağımlılıklarının sürdürülebilirliğini sağlar ve uygulama dağıtımını hızlandırır. Ayrıca, Docker konteynerleri, farklı işletim sistemleri ve altyapılarda sorunsuz çalışabilir, bu da taşınabilirlik sağlar. Bu nedenle, özellikle bulut tabanlı ve mikro servis tabanlı uygulamaların geliştirilmesi ve yönetilmesi için tercih edilen bir teknoloji haline gelmiştir (*Docker Docs*, 2023).

NVIDIA Docker (nvidia-docker), NVIDIA grafik işlem birimlerini (GPU'lar) içeren sistemlerde Docker konteynerlerini kullanmayı ve GPU hızlandırmasını etkin bir

şekilde entegre etmeyi mümkün kılan bir araçtır. Bu teknoloji, özellikle derin öğrenme, yapay zeka ve bilimsel hesaplama gibi GPU yoğun iş yüklerini çalıştırmak isteyen geliştiriciler ve araştırmacılar için önemlidir. NVIDIA Docker, GPU hızlandırmalı iş yükleri için Docker konteynerlerini optimize etmek amacıyla geliştirilmiş bir araçtır. CUDA ve CuDNN gibi NVIDIA'nın GPU hızlandırma kütüphanelerini içeren konteynerlerin sorunsuz oluşturulmasını ve çalıştırılmasını sağlar (*Cuda Architecture Overview*, 2023).

Bu çalışmada NVIDIA Docker, manuel kurulum sırasında bağımlı olunan kütüphane ve teknolojilerin uyumsuz versiyonlarının yanlışlıkla kurulma riskini önlemek için kullanılmıştır. Eğer uyumsuz bir versiyon yanlışlıkla kurulursa, modelin eğitimi ya hiç başlamaz ya da sorunlu bir şekilde devam eder ve model öğrenmesi gerçekleşmez. Bu tür durumlardan kaçınmak amacıyla YOLOv7 kaynak kodunun içerisindeki Docker dosyası kullanılarak eğitimler gerçekleştirilmiştir. Kullanılan Docker dosyası EK-1'de mevcuttur.

3.7. Label Studio

Label Studio, çok çeşitli projeleri, kullanıcıları ve veri türlerini destekleyen bir açık kaynak veri etiketleme aracıdır. Bu platform, özellikle yapay zeka ve makine öğrenimi alanlarında veri etiketleme süreçlerini optimize etmek isteyen araştırmacılar, veri bilimcileri ve yapay zeka mühendisleri için güçlü bir kaynaktır (*Label Studio Documentation*, 2023).

Label Studio, metin, görüntü, ses ve video gibi farklı veri türlerini etiketlemek için kullanılabilir. Bu, yapay zeka ve makine öğrenimi modellerinin eğitimi için zorunlu bir aşamadır. Çünkü bu modeller doğru sonuçlar elde etmek için etiketlenmiş verilere ihtiyaç duyarlar. Platform, farklı etiketleme işlemleri için kullanılabilen çok çeşitli etiketleme araçları sunar. Bu araçlar, nesne tanıma, duygu analizi, metin sınıflandırma ve daha birçok görev için özelleştirilebilir.

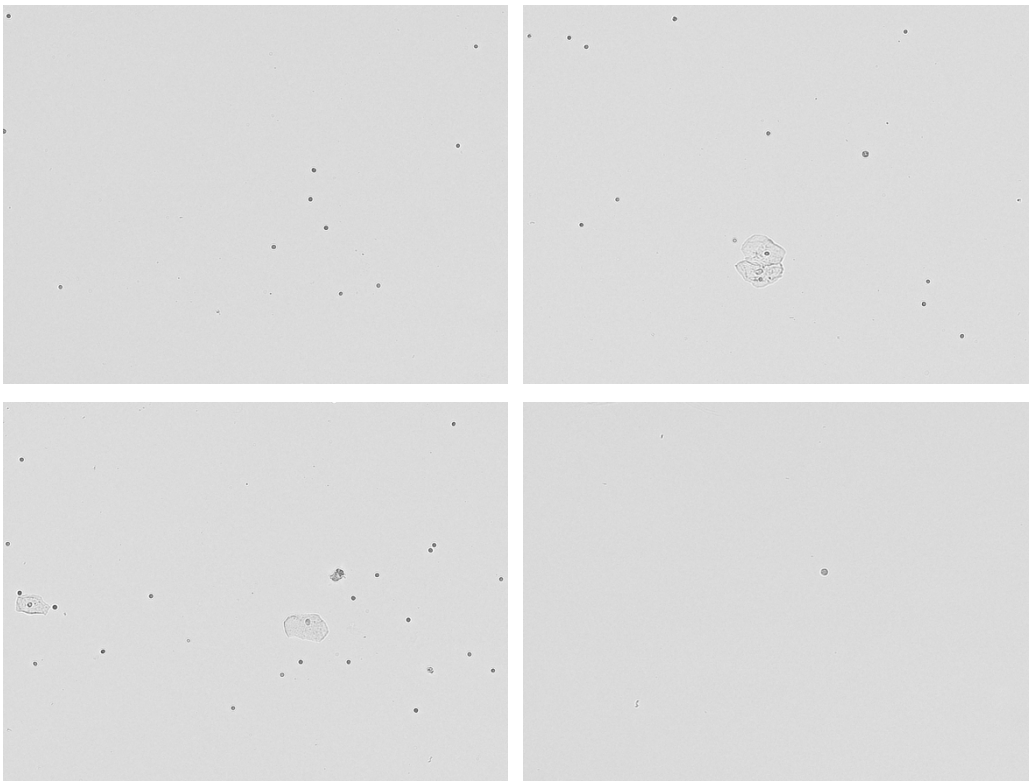
Label Studio, kullanıcıların projelerine ve özel gereksinimlerine uyacak şekilde özelleştirilebilir. İş akışları oluşturulmasına olanak tanır. Bu, farklı projeler için uyarlanabilir bir çözüm sunar. Birden fazla kişi veya ekip, Label Studio üzerinden aynı projede iş birliği yapabilir. Bu, büyük veri kümelerini hızlı bir şekilde etiketlemeyi kolaylaştırır ve veri kalitesini arttırır.

Etiketlenmiş verilerin görselleştirilmesi, etiketleme sürecini daha anlaşılır ve verimli hale getirir. Label Studio, bu görselleştirmeyi destekler ve veri analitiğini kolaylaştırır. Platform, diğer veri işleme araçları ve platformlarıyla entegre edilebilir. Bu, etiketlenmiş verilerin daha geniş bir yapay zeka iş akışına kolayca dahil edilmesini sağlar.

Açık kaynaklı Label Studio, kullanıcıların kaynak kodunu inceleyebileceği, özelleştirebileceği ve ücretsiz olarak kullanabileceği anlamına gelir. Bu, platformun geniş bir topluluk tarafından geliştirilmesine ve desteklenmesine imkan tanır. Label Studio, etiketlenmiş verilerin kalitesini izlemek ve gerektiğinde düzeltmek için kullanışlı araçlar sunar. Bu da veri kalitesini kontrol etme sürecini iyileştirir.

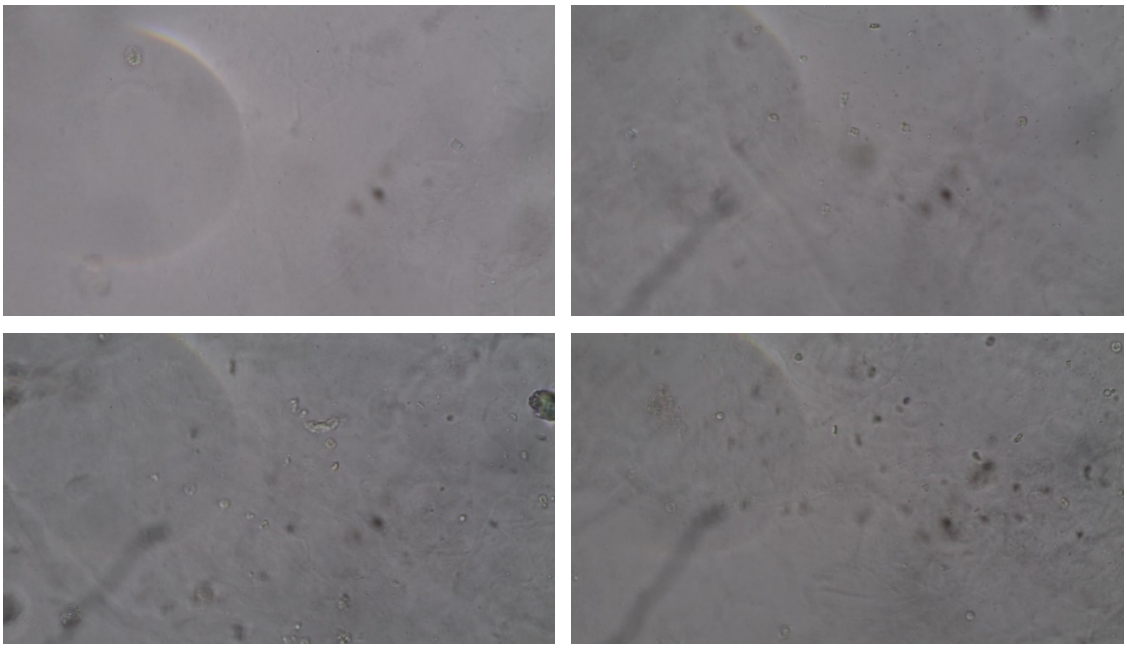
3.8. Veri Seti

Bu çalışmada, iki farklı veri seti birleştirilerek kullanılmıştır. İlk veri kümesi Atıcı ve ark. (2023) tarafından kullanılan görselleri içermektedir. Görüntüler, DIRUI FUS-2000 model mikroskop cihazından elde edilmiştir. Atıcı ve ark. (2023) çalışmalarında, toplamda 9.004 görsel kullanarak 7 farklı partikülün tespitini gerçekleştirmişlerdir. Bu çalışmanın odak noktası eritrosit ve lökosit tespiti olduğu için, veri kümesine yalnızca en az bir eritrosit veya lökosit içeren görseller dahil edilmiştir. Bu yaklaşım, dahil edilen her bir görselin en az bir eritrosit veya lökosit içerdiği anlamına gelmektedir. Ancak bu görseller aynı zamanda diğer hücre tiplerini de içerebilir. Bu prosedür sonucunda, toplamda 5.393 görsel veri kümesine dahil edilmiştir (Şekil 3.11.). Bu görsellerin yaklaşık %20'si test, %5'i doğrulama için kullanılmıştır. Test kümesi, modelin eğitim sonrası başarısını değerlendirmek için kullanılırken doğrulama kümesi modelin eğitim süreci sırasında YOLOv7 tarafından başarımını ölçmek için kullanılmıştır.



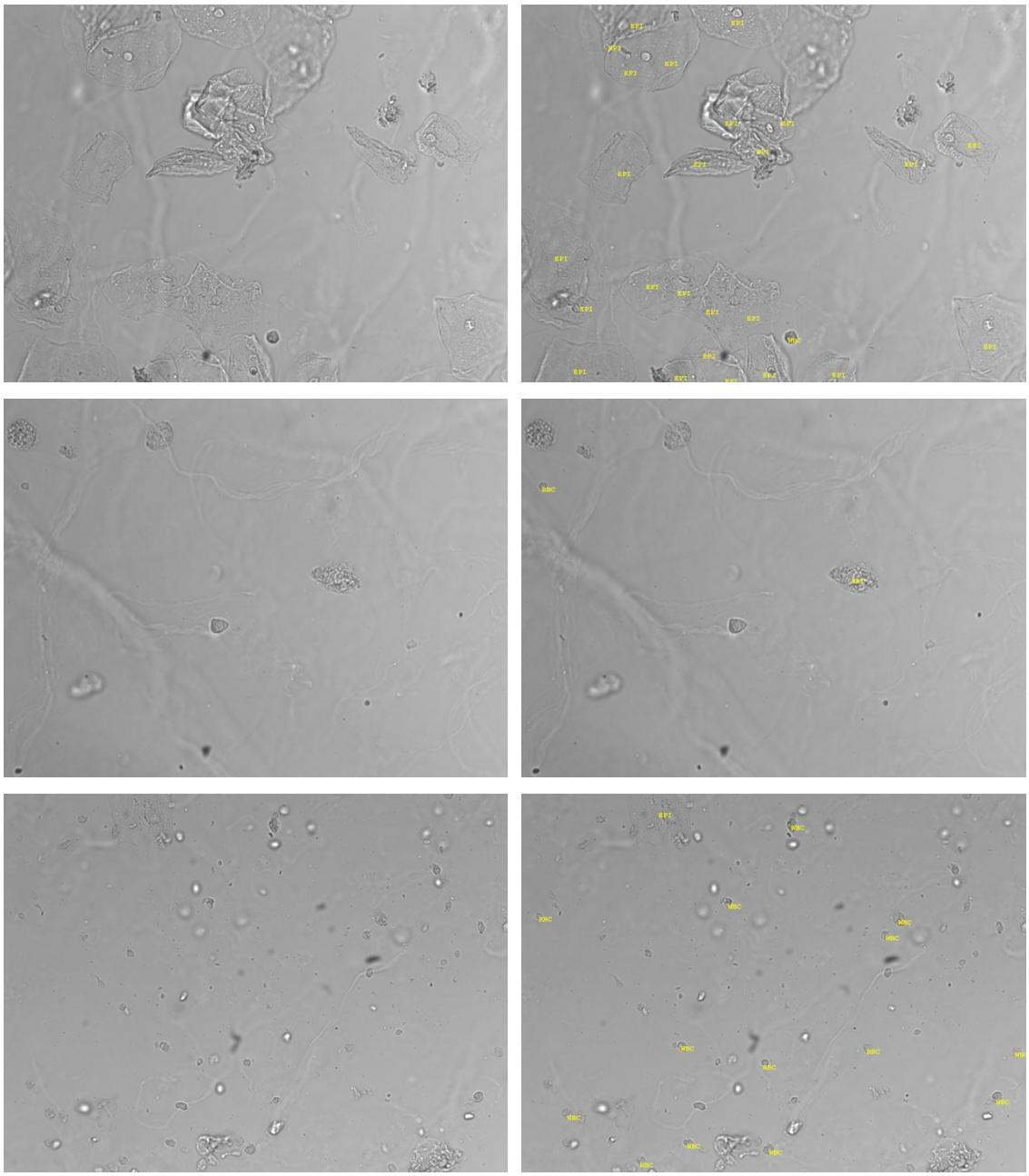
Şekil 3.11. Atıcı ve ark. (2023) tarafından kullanılan örnek görüntüler

Bu tez çalışmasında kullanılan ikinci veri kümesi ise Goswami ve ark. (2021) tarafından hazırlanmıştır. Bu veri kümesinde eritrosit (RBC), lökosit (WBC) ve epitel hücreleri işaretlenmiştir ve toplamda 366 adet görüntü bulunmaktadır. Bu çalışmanın odak noktası eritrosit ve lökosit tespiti olduğu için, veri kümesine yalnızca en az bir eritrosit veya lökosit içeren görseller dahil edilmiştir. Bu yaklaşım, dahil edilen her bir görselin en az bir eritrosit veya lökosit içerdiği anlamına gelmektedir. Ancak bu görseller aynı zamanda diğer hücre tiplerini de içerebilir. Bu prosedür sonucunda, toplamda 358 adet görüntü veri kümesine dahil edilmiştir (Şekil 3.12.). Bu görüntülerin yaklaşık %14'ü test, %10'u doğrulama için ayrılmıştır. Test kümesi, modelin eğitim sonrası performansını değerlendirmek için kullanılmış, doğrulama kümesi ise eğitim sırasında modelin performansını ölçmek için YOLOv7 tarafından kullanılmıştır.



Şekil 3.12. Goswami ve ark. (2021) tarafından kullanılan örnek görüntüler

Çalışma kapsamında üretilen model günümüzde kullanılan idrar analiz cihazıyla karşılaştırılmıştır. Karşılaştırma için Urised Labumat 2 model idrar analiz cihazı kullanılmıştır. Bu cihaz, ham görüntüleri bitmap formatında hem işaretli hem de işaretsiz olarak kaydetme olanağı sunmaktadır. Bu nedenle adil bir karşılaştırma yapmak için bu cihazdan 170 adet işaretli ve işaretsiz görüntü elde edilmiştir (Şekil 3.13.).



(a) (b)

Şekil 3.13. Urised marka idrar analizör cihazından aldığımız görüntüler. (a) Orijinal görüntüler, (b) Urised Labumat 2 cihazının işaretlemiş olduğu görüntüler

Bu çalışmada, model eğitimi aşamasında 4,309 adet, eğitim aşamasında doğruluğu ölçmek için 307 adet ve çalışma sonunda modeli test etmek için 1,305 adet görüntüden oluşan bir veri kümesi oluşturulmuştur. Bu veri kümesi, modelin genel performansını değerlendirmek için kapsamlı bir analiz imkânı sunmaktadır. Veri kümesine ait detaylar Çizelge 3.3'te verilmiştir.

Çizelge 3.3. Eğitim, doğrulama ve test veri seti

Veri Seti	Eğitim	Doğrulama	Test	Boyut
Atıcı ve ark. (2023)	4046	269	1078	800x600
Goswami ve ark. (2021)	263	38	57	1280x720
Urised Labumat 2	-	-	170	1280x960
Toplam	4309	307	1305	-

Bu tez çalışması kapsamındaki görüntüler, Necmettin Erbakan Üniversitesi Fen ve Mühendislik Bilimleri Bilimsel Araştırmalar Etik Kurulu Başkanlığının 12 Ocak 2023 tarihli ve 2023/01 sayılı Etik Kurul Kararı'yla toplanmış ve kullanılmıştır.

4. ARAŞTIRMA BULGULARI VE TARTIŞMA

4.1. Model Tespiti

Veri Seti alt bölümünde, iki ayrı hazır veri setinin birleştirildiği ve tek bir veri seti haline getirilerek kullanıldığı belirtilmişti. Birleştirme işlemine birtakım deneysel çalışmalar sonucunda karar verilmiştir. Birleştirme işleminden önce her iki veri kümesi de ayrı ayrı eğitilmiş ve elde edilen modeller değerlendirilmiştir. Modeller, kendi test verilerinde başarılı sonuçlar üretmişlerdir. Ancak Urised Labumat 2 model idrar analiz cihazından alınan görüntüler üzerinde hedeflenen sonuca ulaşamamışlardır. Bu nedenle iki veri kümesi birleştirilmiş ve benzer görüntülerin tespiti önlenmeye çalışılmıştır. Veri kümeleri birleştirildikten sonra mAP değerlerinde biraz düşüş gözlenmiştir. Buna karşılık Urised Labumat 2 model idrar analiz cihazından alınan görüntüler üzerinde başarının arttığı görülmüştür. Bu nedenle, daha sonraki deneyler birleştirilmiş veri kümesi kullanılarak yapılmıştır. Tüm eğitimler, AMD Ryzen 5600X işlemci, Nvidia RTX3080 10 GB ekran kartı ve 32 GB 3600MHz CL16 RAM belleğe sahip kişisel bilgisayarda gerçekleştirilmiştir. Çizelge 4.1'de birleştirilmiş veri kümesi üzerinde gerçekleştirilen çalışmalar ayrıntılı olarak sunulmuştur. Daha detaylı tablo EK-2'de verilmiştir.

Çizelge 4.1. Birleştirilmiş veri seti ile yapılan bazı çalışmalar.

Sıra	Model	Epok	Grup Boyutu	Görsel Boyutu
1	YOLOv7-tiny	150	32	800
2	YOLOv7	150	2	800
3	YOLOv7-tiny	150	50	640
4	YOLOv7-tiny	150	50	640
5	YOLOv7-tiny	150	50	640

Çizelge 4.1'e bakıldığında, epok sayısının her zaman 150 olarak seçildiği görülmektedir. Bu seçimin nedeni, önceki çalışmalarda 300 epok gibi daha yüksek bir değer seçildiğinde modelin ezberlemeye başladığının ve mAP değerinin düştüğünün gözlenmesidir.

Çizelge 4.1'deki her bir çalışmaya ait mAP değerleri Çizelge 4.2 ve Çizelge 4.3'te ayrıntılı olarak sunulmuştur. Bu değerlerden en doğru modelin 1 numaralı model olduğu görülebilmektedir. Bu nedenle, bundan sonra yapılan deneyler ve karşılaştırmalarda bu model kullanılmıştır.

Çizelge 4.2. Eritrosit sınıfı için mAP değerleri

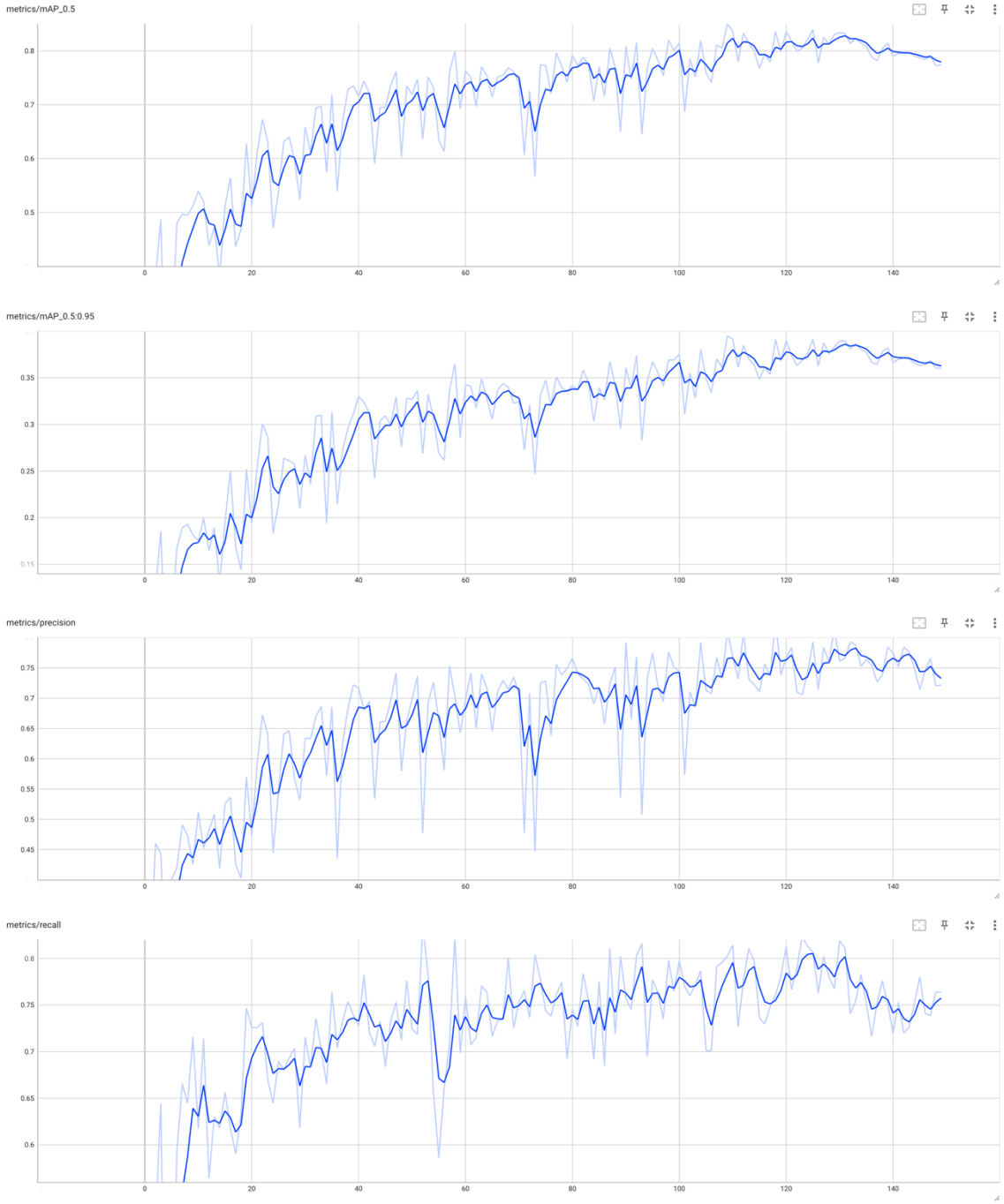
Sıra No	Atıcı ve ark. (2023)	Goswami ve ark. (2021)	Urised Labumat 2
1	**0.902**	0.813	**0.413**
2	0.885	0.795	0.381
3	0.644	0.694	0.248
4	0.806	0.764	0.291
5	0.795	**0.835**	0.301

Çizelge 4.3. Lökosit sınıfı için mAP değerleri

Sıra No	Atıcı ve ark. (2023)	Goswami ve ark. (2021)	Urised Labumat 2
1	**0.832**	0.652	0.573
2	0.653	0.602	0.594
3	0.270	0.545	0.562
4	0.485	0.597	**0.601**
5	0.521	**0.733**	0.481

4.2. Sonuçların Diğer Çalışmaların Sonuçları İle Karşılaştırılması

Eğitim, yaklaşık olarak 75 dakika sürmüştür. Bu süreçte model sürekli olarak öğrenme eğilimi göstermiştir. Ancak son 20 epok boyunca mAP grafiğinde bir düşüş gözlenmiştir. Eğitim, eritrosit ve lökosit sınıfları için, sırasıyla, 0.879 ve 0.67 mAP değeri ile tamamlanmıştır. Eğitim sürecine ait grafikler Şekil 4.1'de sunulmuştur. Eğitim sonunda eritrosit sınıfı için daha yüksek bir mAP değeri elde edilmiştir. Bu farkın nedeni, Atıcı ve ark. (2023)'nın veri setinde eritrositin yaklaşık %60 oranında iken lökositin yaklaşık %22 oranında bulunmasıdır.



Şekil 4.1. YoloV7-tiny eğitimi sırasında epok numarasına göre mAP 0.5, mAP 0.95, hassasiyet ve duyarlılık değerlerinin grafikleri

Şekil 4.1'deki grafikler sırasıyla eğitim sırasında her epokda hesaplanan mAP 0.5, mAP 0.5:0.95, hassasiyet ve duyarlılık ölçütlerini içermektedir. mAP grafikleri, modelin nesne algılama yeteneğini genel olarak özetler. Yüksek mAP değerleri, modelin hem yüksek hassasiyet hem de yüksek duyarlılık sergilediğini göstermektedir. mAP 0.5 grafiği, modelin nesneleri en az %50 örtüşme (IoU) ile doğru bir şekilde tespit ettiğini gösterirken, mAP 0.5:0.95 grafiği, farklı eşik değerleri (0.5 ile 0.95 arası) kullanarak bir

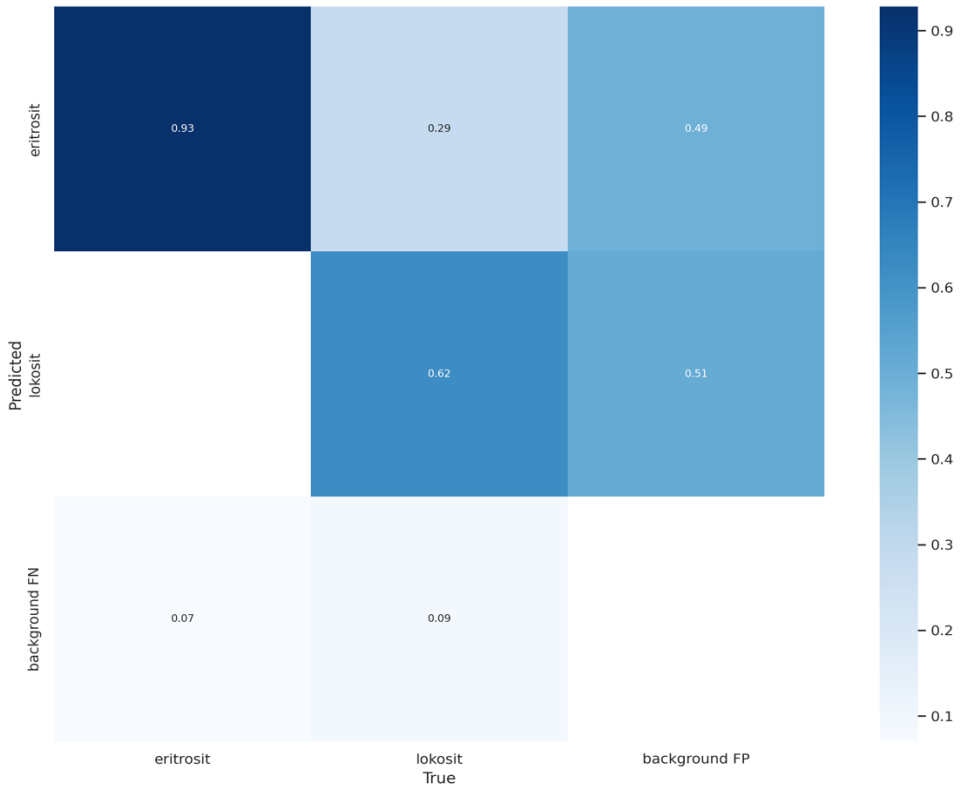
nesnenin doğru bir şekilde tespit edildiğini göstermektedir. mAP 0.5:0.95 grafiği, modelin farklı hassasiyet seviyelerinde ne kadar başarılı olduğunu daha ayrıntılı bir şekilde değerlendirmemize olanak tanır. Hassasiyet grafiği, modelin yaptığı tahminlerin ne kadar kesin olduğunu gösterirken, duyarlılık grafiği, modelin ne kadar eksiksiz olduğunu gösteren bir metriktir.

Derin öğrenme sürecinin sonucunda elde edilen karışıklık matrisi oldukça önemli bir değerlendirme aracıdır. Bu matris, modelin sınıflandırma performansını açıkça gösterir ve tahmin edilen sınıflar ile gerçek sınıflar arasındaki doğru sınıflandırma oranlarını içerir. Normalleştirilmiş değerler üzerinden sunulan bu matris, önemli bilgiler sağlar.

Şekil 4.2'de yer alan karışıklık matrisi incelendiğinde, eritrosit sınıfına yönelik sonuçların oldukça dikkat çekici olduğu görülmektedir. Modelin eritrositleri tahmin etme yeteneği %93'lük bir doğruluk oranına sahiptir. Ancak, söz konusu sınıfın tahmini sırasında %29'luk bir lökosit hata oranı gözlenmiştir. Ayrıca, eritrosit sınıfının %49'u yanlışlıkla işaretlenmiştir. Bu sonuçlar, modelin eritrositleri doğru bir şekilde tespit etme yeteneğine sahip olduğunu, ancak bazı durumlarda lökosit ve arka plan sınıflarıyla karıştırıldığını göstermektedir.

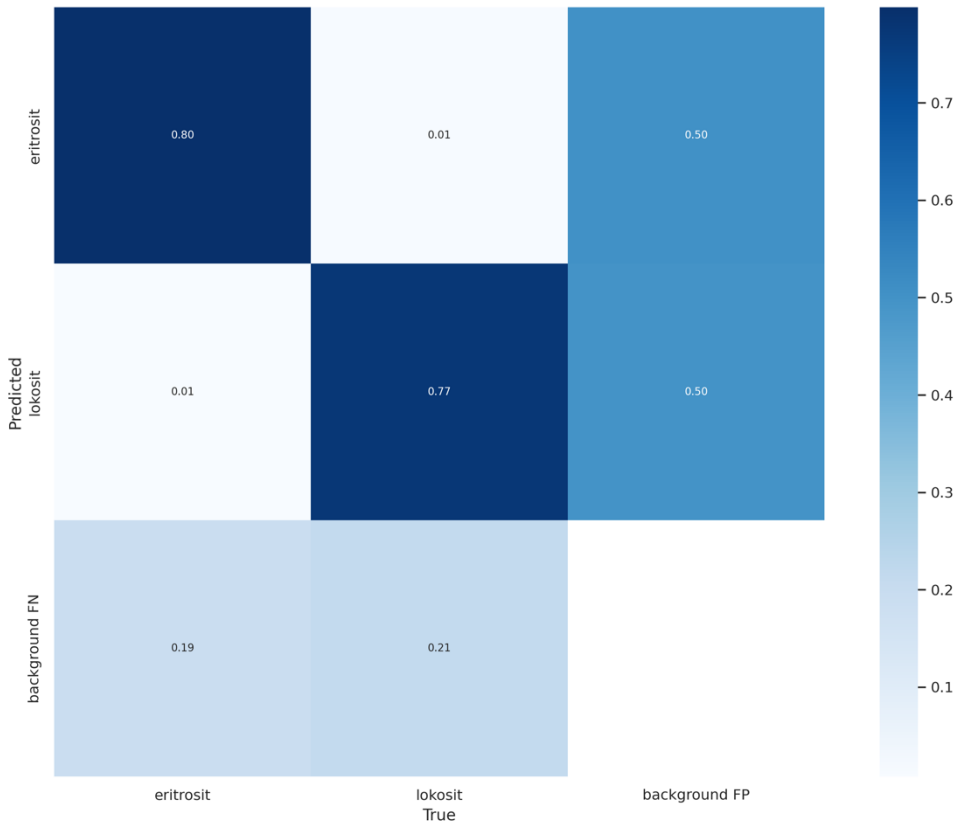
Modelin lökosit sınıfını tahmin etme yeteneği eritrosite göre daha zayıftır. Tahminlerin sadece %62'si doğru bir şekilde lökosit olarak sınıflandırılmıştır. Tahmin edilen lökositlerin, eritrosit olarak yanlışlıkla etiketlenmediği görülmektedir. Lökosit sınıfının %51'i ise yanlışlıkla sınıflandırılmıştır

Karışıklık matrisi, derin öğrenme tabanlı sınıflandırma modelinin performansını ayrıntılı bir şekilde gösterir. Analiz sonuçları, eritrosit sınıfının daha yüksek doğruluk oranlarına sahip olduğunu, lökosit sınıfının ise daha düşük doğruluk değerlerine sahip olduğunu göstermektedir. Bu sonuçlar, modelin lökositleri tanıma konusunda ve daha fazla eğitim verisiyle desteklenmesi gerektiğini ifade etmektedir.



Şekil 4.2. YOLOv7-tiny modelinin karışıklık matrisi

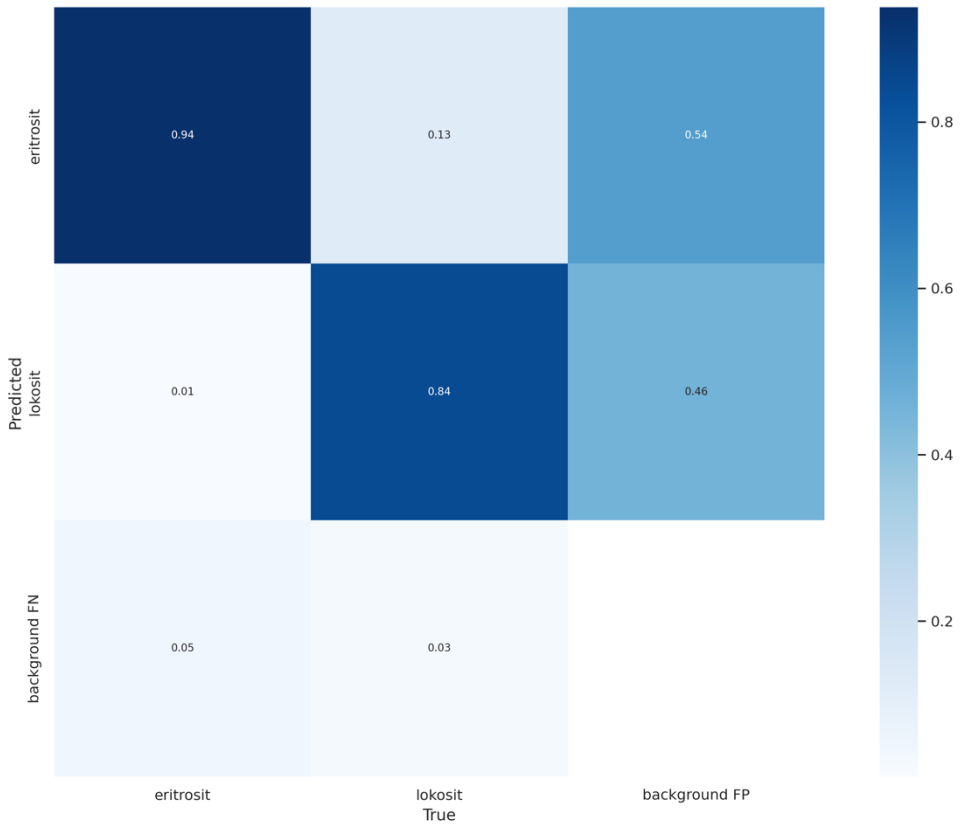
Çalışmada kullanılan veri setinin oluşturulmasında iki farklı veri kümesinin birleştirildiği daha önce ifade edilmişti. Elde edilen modelin performansını değerlendirmek amacıyla bu model, iki farklı veri kümesinin test görüntüleri üzerinde test edilmiş ve bu testler sonucunda elde edilen karışıklık matrisleri Şekil 4.3 ve Şekil 4.4'te sunulmuştur.



Şekil 4.3. Goswami ve ark. (2021) oluşturduğu veri setinin test kümesindeki karışıklık matrisi

Şekil 4.3'te sunulan karışıklık matrisinden anlaşıldığı üzere, eritrosit ve lökosit sınıflarına ait sonuçlar birbirlerine yakın değerleri göstermektedir. Modelin eritrositleri tahmin etme yeteneği %80'lik bir doğruluk oranına sahiptir. Bununla birlikte, eritrosit sınıfının %50'si yanlışlıkla işaretlenmiştir. %19'luk bir kısım ise eritrosit olduğu halde hiç işaretlenmemiştir. Bu sonuçlar, modelin eritrositleri doğru bir şekilde tanımlama yeteneğine sahip olduğunu, ancak bazı durumlarda arka plan sınıflarıyla karıştırıldığını veya görmezden gelindiğini göstermektedir.

Modelin lökosit sınıfını tahmin etme yeteneği biraz daha zayıftır. Tahminlerin sadece %77'si gerçekten lökosit olarak doğru bir şekilde sınıflandırılmıştır. Lökosit sınıfının %50'si yanlışlıkla işaretlenmiştir. %21'lik bir kısım ise doğru olduğu halde yanlışlıkla hiç işaretlenmemiştir. Bu sonuçlar, modelin lökositleri doğru bir şekilde tanımlama yeteneğine sahip olduğunu, ancak bazı durumlarda arka plan sınıflarıyla karıştırıldığını veya görmezden gelindiğini göstermektedir.



Şekil 4.4. Atıcı ve ark. (2023) oluşturduğu veri setinin test kümesindeki karışıklık matrisi

Şekil 4.4'teki karışıklık matrisinden eritrosit sınıfına yönelik sonuçların oldukça dikkat çekici olduğu görülmektedir. Modelin eritrositleri tahmin etme yeteneği %94'lük bir doğruluk oranına sahiptir. Ancak, eritrosit sınıfının tahmini sırasında %13'lük bir lökosit hatası oranı gözlenmiştir. Ayrıca, eritrosit sınıfının %54'ü yanlışlıkla işaretlenmiştir. Bu sonuçlar, modelin eritrositleri doğru bir şekilde sınıflandırma yeteneğine sahip olduğunu, ancak bazı durumlarda lökosit ve arka plan sınıflarıyla karıştırıldığını göstermektedir.

Modelin lökosit bu sınıfını tahmin etme yeteneği biraz daha zayıf tır. Tahminlerin sadece %84'ü doğru bir şekilde gerçekten lökosit olarak sınıflandırılmıştır. Tahmin edilen lökositlerin, eritrosit olarak yanlışlıkla etiketlenmediği görülmektedir. Lökosit sınıfının %46'sı ise yanlışlıkla arka plan olarak sınıflandırılmıştır. Bu sonuçlar, modelin lökositleri belirleme konusunda biraz daha iyileştirme ve hassaslaştırmaya ihtiyaç duyduğunu göstermektedir.

Atıcı ve ark. (2023) tarafından yapılan çalışmanın sonuçları ile elde edilen sonuçlar karşılaştırıldığında, her iki çalışmada da eritrosit sınıfı için %94'lük doğruluk oranı elde edilmiştir. Lökosit sınıfına gelindiğinde, bu sonuçlar %78'den %84'e

yükselmiştir. Geliştirilen model lökosit sınıfını tanıma konusunda daha üstün performans sergilemektedir. Ayrıca, Atıcı ve ark. (2023) çalışmalarında YoloV7 modelini kullanmışlardır. Bu çalışmada ise daha hafif ve hızlı çalışan YoloV7-tiny modeli kullanılmıştır. Bu nedenle, geliştirilen model çok daha hızlı tespit yapabilmektedir. Test veri kümelerine ait hassasiyet, duyarlılık ve mAP değerleri Çizelge 4.4 ve Çizelge 4.5'te verilmiştir.

Çizelge 4.4. Eritrosit sınıfı için performans metrikleri

Veri Seti	Hassasiyet	Duyarlılık	mAP 0.5
Atıcı ve ark. (2023)	0.874	0.859	0.902
Goswami ve ark. (2021)	0.761	0.724	0.813

Çizelge 4.5. Lökosit sınıfı için performans metrikleri

Veri Seti	Hassasiyet	Duyarlılık	mAP 0.5
Atıcı ve ark. (2023)	0.769	0.848	0.832
Goswami ve ark. (2021)	0.677	0.66	0.652

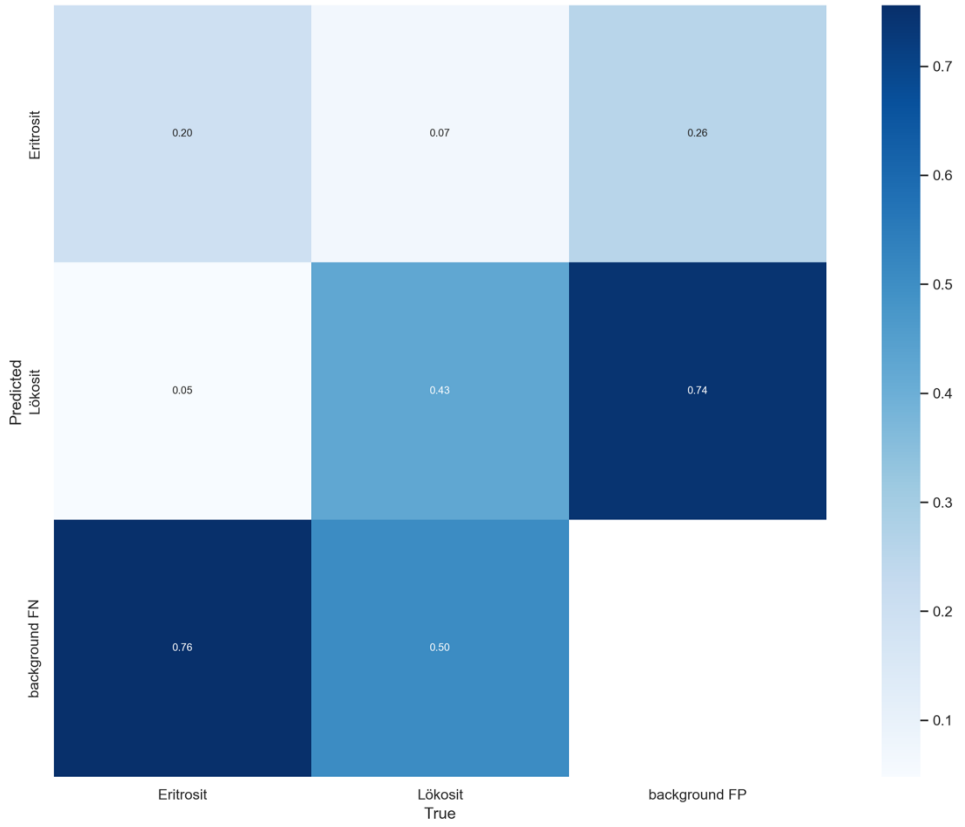
4.3. Sonuçların İdrar Analizör Cihazının Sonuçları İle Karşılaştırılması

Urised Labumat 2 model tam otomatik idrar analiz cihazından alınan 170 adet görüntü uzman laborantlar tarafından işaretlendi. İşaretleme sürecinde açık kaynak olarak sunulan Label Studio yazılımı kullanıldı. Yeni bir sunucu oluşturularak Label Studio sunucuda çalıştırıldı. Bu yazılıma görüntüler yüklendi ve çalışma ortamı hazır hale getirildi. Laborantlar, kendi kullanıcı adları ve şifreleri ile sisteme giriş yaptılar ve görsel işaretleme işlemini gerçekleştirdiler. İşaretleme yapılırken yalnızca %100 emin olunan hücreler işaretlendi.

Urised Labumat 2 model idrar analiz cihazı tarafından işaretlenmiş görüntülerin de alındığı daha önce belirtilmişti (Şekil 3.13). Bu görüntüler de Label Studio üzerinde cihazın işaretlemeleri baz alınarak işaretlenmiştir.

Urised Labumat 2 model cihazın işaretlemelerine ait karışıklık matrisini çıkarmak için Label Studio'dan JSON formatında bir çıktı alındı. JSON dosyası, her bir işaretlemenin koordinatlarını ve hangi sınıfa ait olduğunu içerir. Laborantlar tarafından işaretlenen görüntüler ile makine tarafından işaretlenen görüntüler üzerinde IoU değerini hesaplayan ve bu değere eşit veya daha yüksek olanları doğru kabul edip buna göre bir

karışıklık matrisi oluşturan Python betiği hazırlandı. Hazırlanan betikte IoU değeri YoloV7'deki test betiğinde varsayılan olarak kullanılan %65 değeri kullanılmıştır. Betiğin ürettiği karışıklık matrisi Şekil 4.5'te görülmektedir.

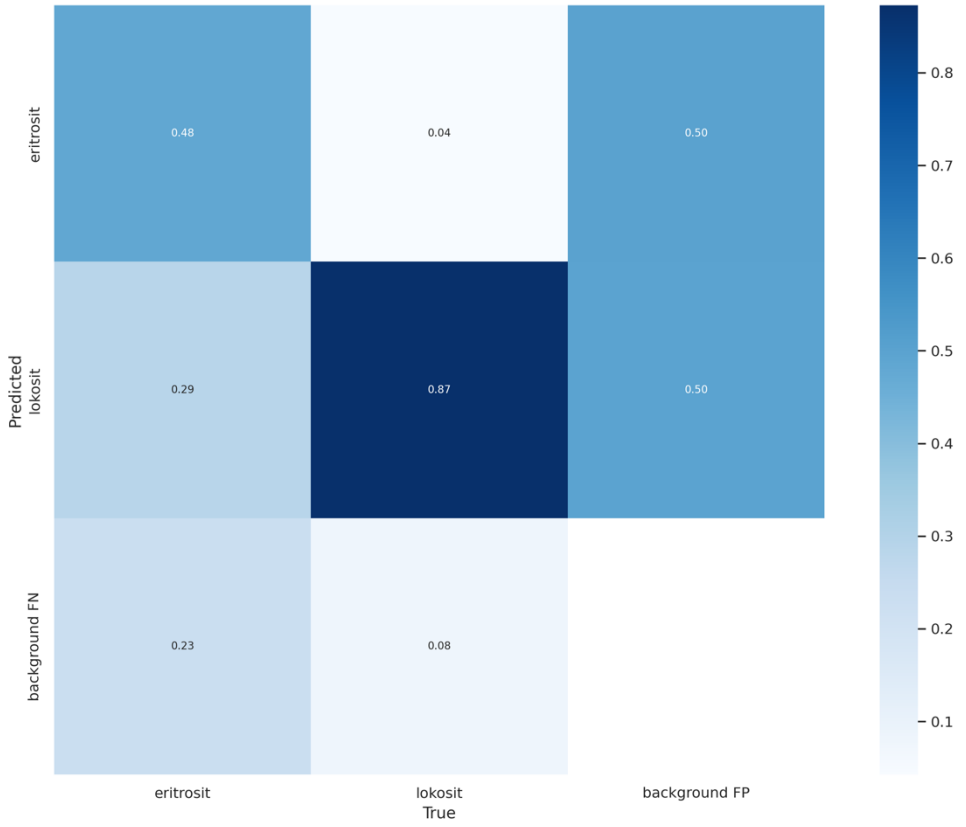


Şekil 4.5. Urised Labumat 2 model cihazının karışıklık matrisi

Şekil 4.5'te yer alan karışıklık matrisinden de anlaşılacağı üzere, eritrosit sınıfındaki sonuçlar oldukça düşüktür. Cihazın eritrositleri tahmin etme yeteneği %20'lik bir doğruluk oranına sahiptir. Eritrosit sınıfının %26'sı yanlışlıkla işaretlenmiştir ve %76'sı cihaz tarafından hiç tespit edilememiştir. Bu sonuçlar, cihazın eritrositleri doğru bir şekilde tanıma yeteneğinin çok düşük olduğunu ve genellikle hiç tespit edemediğini göstermektedir.

Cihazın lökosit sınıfını tahmin etme yeteneği biraz daha yüksektir. Tahminlerin %43'ü gerçekten lökosit olarak doğru bir şekilde sınıflandırılmıştır. Lökosit sınıfının %74'ü yanlışlıkla işaretlenmiş ve %50'si cihaz tarafından hiç tespit edilmemiştir. Bu sonuçlar, cihazın lökositleri eritrositlere göre biraz daha iyi bir şekilde tanıma yeteneğine sahip olduğunu göstermektedir.

Üretilen modelin 170 görüntüden oluşan veri seti üzerindeki karışıklık matrisini elde etmek için laborantların işaretlediği görseller YOLO formatında kaydedildi ve Python betiği çalıştırılarak karışıklık matrisi oluşturuldu (Şekil 4.6).



Şekil 4.6. Modelin 170 görüntüden oluşan veri setindeki karışıklık matrisi

Şekil 4.6'daki karışıklık matrisinden de görülebildiği gibi eritrosit sınıfındaki sonuçlar biraz düşüktür. Modelin eritrositleri tahmin etme yeteneği %48'lik bir doğruluk oranına sahiptir. Eritrosit sınıfının %50'si yanlışlıkla işaretlenmiş ve %23'lük bir kısım model tarafından hiç tespit edilememiştir. Bu sonuçlar, modelin eritrositleri doğru bir şekilde sınıflandırma yeteneğinin biraz düşük olduğunu ve bu alanda daha fazla iyileştirmeye ihtiyaç duyulduğunu göstermektedir.

Modelin lökosit sınıfını tahmin etme yeteneği biraz daha iyidir. Tahminlerin %87'si gerçekten lökosit olarak doğru bir şekilde sınıflandırılmıştır. Ancak lökosit sınıfı, %29'luk bir oranda eritrosit sınıfı ile karıştırılmıştır. Lökosit sınıfının %50'si yanlışlıkla işaretlenmiş ve %1'lik bir kısım cihaz tarafından hiç tespit edilememiştir. Bu sonuçlar, modelin lökositleri eritrositlere göre daha iyi bir şekilde tanıma yeteneğine sahip olduğunu ve kabul edilebilir bir başarı oranı elde ettiğini göstermektedir.

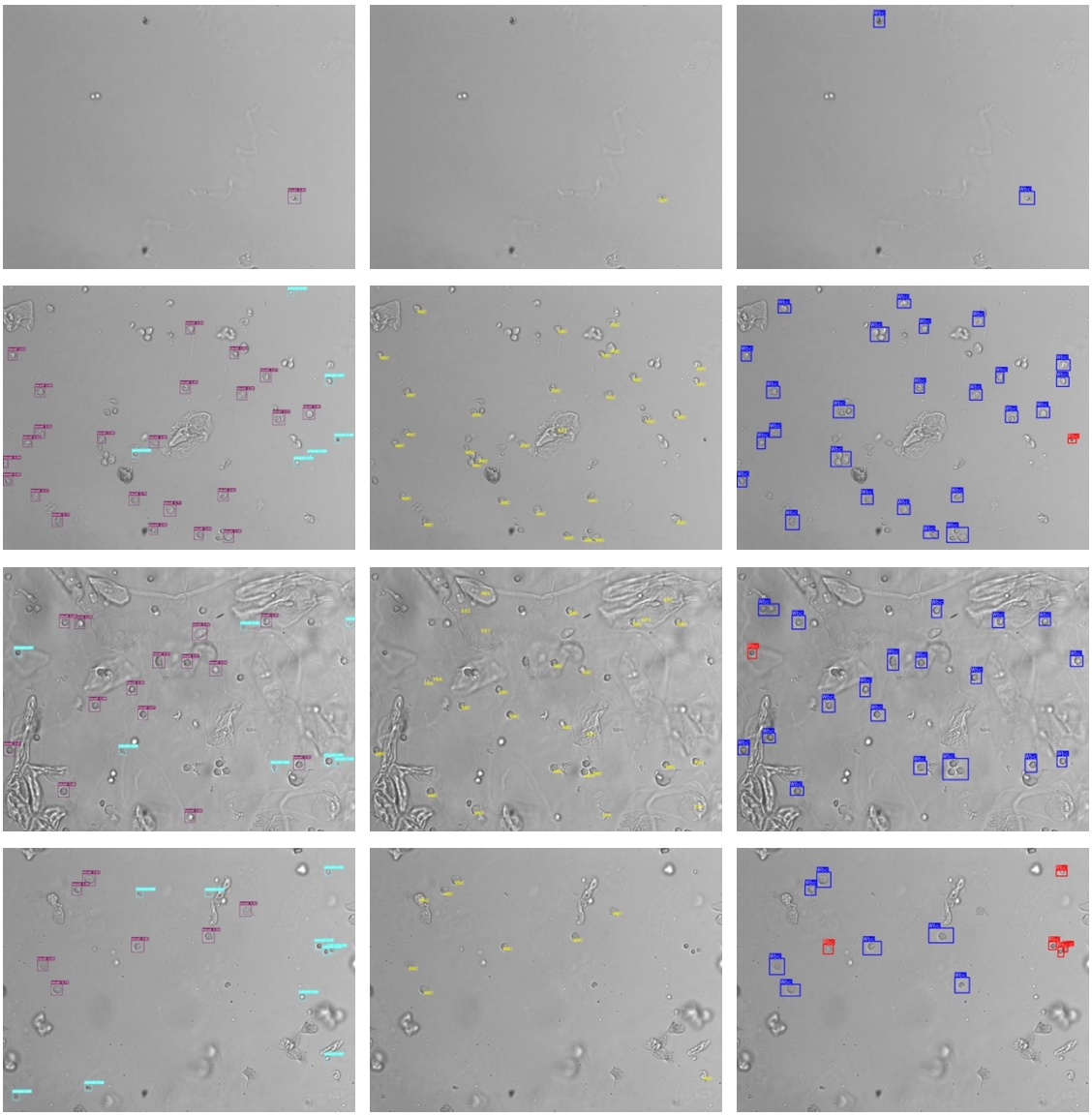
Karışıklık matrisi üzerinden yapılan karşılaştırmalar, Urised Labumat 2 model cihazının elde ettiği sonuçlardan daha yüksek sonuçların elde edildiğini göstermektedir. Modelin, Urised Labumat 2 model cihaza ait veri kümesindeki performans metrikleri Çizelge 4.6'da verilmiştir.

Çizelge 4.6. Modelin Urised Labumat 2 model cihaza ait veri kümesindeki performans metrikleri

Sınıf	Hassasiyet	Duyarlılık	mAP@.5
Eritrosit	0.285	0.503	0.37
Lökosit	0.596	0.53	0.52

Çizelge 4.6'da yer alan performans metriklerini incelediğimizde, eritrosit sınıfı için düşük hassasiyet değeri, modelin yanlış pozitif tahminler yaptığını göstermektedir. Aynı sınıf için düşük duyarlılık ise modelin bazı eritrositleri kaçırdığını ifade eder. Lökosit sınıfı için ise kabul edilebilir bir hassasiyet değeri elde edilmiş olsa da modelin yanlış pozitif tahminleri azaltması gerekmektedir. Lökositlerin düşük duyarlılığı, bazı lökositleri tespit edemediğini göstermektedir. Ayrıca, mAP değerleri, modelin genel performansının iyileştirilmesi gerektiğini ifade eder. Bu sonuçlar, modelin performansını artırmak için gereken alanları ve potansiyel iyileştirme stratejilerini belirlemek için önemli ipuçları sunmaktadır.

Geliştirilen model ile Urised Labumat 2 model cihaz göre daha doğru sonuçlar elde edilmektedir. Bu üstünlüğe rağmen, Urised Labumat 2 veri kümesindeki performansın düşük olduğu açıkça görülmektedir. Bunun nedeni, eğitim veri kümesinde Urised Labumat 2 model cihaza ait hiçbir görsel kullanılmamasıdır. Bunu gidermek için, Urised Labumat 2 model cihazdan daha fazla görüntü toplanabilir ve bunlar eğitim kümesine dahil edilebilir. Şekil 4.7'de model, cihaz ve laborant işaretlemelerine ait örnek görseller birlikte sunulmuştur.



(a) (b) (c)

Şekil 4.7. Örnek karşılaştırma görüntüleri. (a) Üretilen model, (b) Urised Labumat 2 cihazı, (c) Laborantın işaretlemeleri

5. SONUÇLAR VE ÖNERİLER

5.1. Sonuçlar

Bu tez çalışmasında, günümüz idrar analiz cihazlarındaki problemlere derin öğrenme yöntemi ile çözüm bulunmaya çalışılmıştır. İnsan idrarındaki partiküller, YOLOv7-tiny derin öğrenme modeli kullanılarak başarıyla tespit edilmiştir. Atıcı ve ark. (2023) ile Goswami ve ark. (2021) tarafından hazırlanan veri kümeleri birleştirilerek yeni bir veri kümesi oluşturulmuştur. Uygulanan deneyler sonucunda, Goswami ve ark. (2021) tarafından hazırlanan veri kümesi için eritrosit ve lökosit tespitinde, sırasıyla, %80 ve %77 doğruluk oranları elde edilmiştir. Atıcı ve ark. (2023) tarafından hazırlanan veri kümesinde ise eritrosit ve lökosit tespitinde, sırasıyla, %94 ve %84 doğruluk oranlarına ulaşılmıştır. Ayrıca, geliştirilen modelin sonuçları günümüzde yaygın olarak kullanılan Urised Labumat 2 model idrar analiz cihazının sonuçları ile de karşılaştırılmıştır. Karşılaştırma sonucunda, geliştirilen modelin eritrosit ve lökosit tespitinde, sırasıyla, %48 ve %87 doğruluk oranlarına ulaştığı ve idrar analiz cihazına göre daha üstün bir performans gösterdiği tespit edilmiştir.

5.2. Öneriler

Modelin doğruluğunu arttırmak için yapılabilecek işlemler ve öneriler aşağıda listelenmiştir:

- **Veri Seti İyileştirmesi:** Eğitim verilerindeki hatalı veya yanlış etiketlenmiş örnekler tespit edilip düzeltilebilir. Ayrıca, veri setindeki anlamsız veya gereksiz görüntüler çıkarılabilir.
- **Daha Fazla Eğitim Verisi:** Modelin doğruluğunu artırmak için daha fazla eğitim verisi toplanabilir. Daha büyük bir veri seti, modelin farklı varyasyonları daha iyi öğrenmesini sağlar. Bu da modelin genelleme yeteneğini artırır.
- **Hata Analizi ve İyileştirmeler:** Modelin hatalı sınıflandırma yaptığı örnekler tespit edilip bu hatalar giderilebilir. Eğitim sürecinde karşılaşılan hataların nedenleri belirlenebilir ve modelin bu hataları yapma olasılığı azaltılabilir.
- **Urised Labumat 2 Veri Seti Güncellemesi:** Urised Labumat 2 cihazına ait veri kümesindeki başarıyı artırmak için cihazdan yeni görseller toplanabilir ve

eğitim kümesine dahil edilebilir. Bu işlem, modelin Urised Labumat 2 cihazından gelen verileri daha iyi anlamasına ve daha hassas sonuçlar üretmesine yardımcı olabilir.

6. KAYNAKLAR

Aktaş, A., Doğan, B. ve Demir, Ö. (2020) "Derin Öğrenme Yöntemleri ile Dokunsal Parke Yüzeyi Tespiti", *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 35(3), 1685-1700.

Atıcı, H., Kocer, H. E., Sivrikaya, A. ve Dağlı, M. (2023) "Analysis of Urine Sediment Images for Detection and Classification of Cells", *Sakarya University Journal of Computer and Information Sciences*, 6(1), 37-47.

Bayram, F. (2020) "Derin Öğrenme Tabanlı Otomatik Plaka Tanıma", *Politeknik Dergisi*, 23(4), 955-960.

Bochkovskiy, A., Wang, C. ve Liao, H. M. (2020) "YOLOv4: Optimal Speed and Accuracy of Object Detection".

Cuda Architecture Overview (2023). https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/arch-overview.html#arch-overview (Erişim: 04 Eylül 2023).

Dang, F., Chen, D., Lu, Y. ve Li, Z. (2023) "YOLOWeeds: A Novel Benchmark of YOLO Object Detectors for Multi-class Weed Detection in Cotton Production Systems", *Computers and Electronics in Agriculture*, 205, 107655.

Docker Docs (2023). https://docs.docker.com/get-started/overview/ (Erişim: 04 Eylül 2023).

Doğan, F. ve Türkoğlu, İ. (2018) "Öğrenme Algoritmalarının Yaprak Sınıflandırma Başarımlarının Karşılaştırılması", *SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES*, 1.

Gallo, I., Rehman, A. U., Dehkordi, R. H., Landro, N., La Grassa, R. ve Boschetti, M. (2023) "Deep Object Detection of Crop Weeds: Performance of YOLOv7 on a Real Case Dataset from UAV Images", *Remote Sensing*, 15(2).

Girshick, R. (2015) "Fast R-CNN", *Proceedings of the IEEE international conference on computer vision*, 1440-1448.

Girshick, R., Donahue, J., Darrell, T. ve Malik, J. (2013) "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 580-587.

Glenn, J. (2020) *YOLOv5*. https://github.com/ultralytics/yolov5 (Erişim: 03 Haziran 2023).

Goswami, D., Aggrawal, H. O., Gupta, R. ve Agarwal, V. (2021) "Urine Microscopic Image Dataset", *arXiv preprint arXiv:2111.10374*.

He, K., Gkioxari, G., Dollár, P. ve Girshick, R. (2017) "Mask R-CNN", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2), 386-397.

Huysal, K. ve Üstündağ, Y. (2015) "Otomatik İdrar Analizörleri: Mikroskobik Bakı", *Türk Klinik Biyokimya Dergisi*, 13(2), 83-87.

İnik, Ö. ve Ülker, E. (2017) "Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri", *Gaziosmanpaşa Bilimsel Araştırma Dergisi*. Gaziosmanpasa Universitesi, Fen Bilimleri Enstitüsü, Fen Edebiyat Fakültesi Bina Girişi, Taşlıçiftlik-TOKAT, 6, 85-104.

Ji, Q., Jiang, Y., Wu, Z., Liu, Q. ve Qu, L. (2023) "An Image Recognition Method for Urine Sediment Based on Semi-supervised Learning", *IRBM*, 44(2), 100739.

Ji, Q., Li, X., Qu, Z. ve Dai, C. (2019) "Research on Urine Sediment Images Recognition Based on Deep Learning", *IEEE Access*, 7, 166711-166720.

Khalid, Z. M., Hawezi, R. S. ve Amin, S. R. M. (2022) "Urine Sediment Analysis by Using Convolution Neural Network", *8th IEC 2022 - International Engineering Conference: Towards Engineering Innovations and Sustainability*, 173-178.

Kızrak, M. A. ve Bolat, B. (2018) "Derin Öğrenme ile Kalabalık Analizi Üzerine Detaylı Bir Araştırma", *Bilişim Teknolojileri Dergisi*, 11(3), 263-286.

Krizhevsky, A., Sutskever, I. ve Hinton, G. E. (2017) "ImageNet Classification with Deep Convolutional Neural Networks", *Commun. ACM*, 60(6), 84-90.

Label Studio Documentation (2023). https://labelstud.io/guide/get_started.html#What-is-Label-Studio (Erişim: 04 Eylül 2023).

Li, C., Li, Lulu, Jiang, H., Weng, K., Geng, Y., Li, Liang, Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, Xiaoming ve Wei, Xiaolin (2022) "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications", *arXiv preprint arXiv:2209.02976*.

Li, Q., Yu, Z., Qi, T., Zheng, L., Qi, S., He, Z., Li, S. ve Guan, H. (2020) "Inspection of Visible Components in Urine Based on Deep Learning", *Medical Physics*, 47(7), 2937-2949.

Li, T., Jin, D., Du, C., Cao, X., Chen, H., Yan, J., Chen, N., Chen, Z., Feng, Z. ve Liu, S. (2019) "The Image-based Analysis and Classification of Urine Sediments Using a LeNet-5 Neural Network", *https://doi.org/10.1080/21681163.2019.1608307*, 8(1), 109-114.

Li, Z., Zhou, Y., Zhao, C., Guo, Y., Lyu, S., Chen, J., Wen, W. ve Huang, Y. (2023) "Design of a Cargo-Carrying Analysis System for Mountain Orchard Transporters Based on RGB-D Data", *Applied Sciences*. Basel, 13(10), 6059.

Liang, Y., Kang, R., Lian, C. ve Mao, Y. (2018) "An End-to-End System for Automatic Urinary Particle Recognition with Convolutional Neural Network", *Journal of Medical Systems*, 42(9), 1-14.

Lin, T. Y., Goyal, P., Girshick, R., He, K. ve Dollar, P. (2017) "Focal Loss for Dense Object Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2), 318-327.

Liu, W., Li, W. ve Gong, W. (2020) "Ensemble of Fine-tuned Convolutional Neural Networks for Urine Sediment Microscopic Image Classification", *IET Computer Vision*, 14(1), 18-25.

Memişoğullari, R., Yıldırım, H. A., Orhan, N. ve Yavuz, Ö. (2008) "Böbrek Biyopsisi Kadar Bilgi Veren Tetkik: Rutin İdrar Analizi", *Düzce Tıp Fakültesi Dergisi*, 3, 77-84.

Nagai, T., Onodera, O. ve Okuda, S. (2022) "Deep Learning Classification of Urinary Sediment Crystals with Optimal Parameter Tuning", *Scientific Reports 2022 12:1*, 12(1), 1-9.

Nazir, A. ve Wani, Mohd. A. (2023) "You Only Look Once - Object Detection Models: A Review", *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, 1088-1095.

Parlak, İ. E. ve Emel, E. (2022) *Alüminyum Döküm Hatalarının Derin Öğrenme Yaklaşımıyla Tespiti ve Sınıflandırılması*, *PQDT - Global*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. ve Chintala, S. (2019) "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Advances in Neural Information Processing Systems*, 32.

Perazella, M. A. (2015) "The Urine Sediment as a Biomarker of Kidney Disease", *American Journal of Kidney Diseases*, 66(5), 748-755.

Redmon, J., Divvala, S., Girshick, R. ve Farhadi, A. (2016) "You Only Look Once: Unified, Real-Time Object Detection", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779-788.

Redmon, J. ve Farhadi, A. (2017) "YOLO9000: Better, Faster, Stronger", *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January, 6517-6525.

Redmon, J. ve Farhadi, A. (2018) "YOLOv3: An Incremental Improvement", *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R. ve Sun, J. (2017) "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137-1149.

Shafi, I., Mazahir, A., Fatima, A. ve Ashraf, I. (2022) "Internal Defects Detection and Classification in Hollow Cylindrical Surfaces Using Single Shot Detection and MobileNet", *Measurement*, 202, 111836.

Stack Overflow Developer Survey 2022 (2023). https://survey.stackoverflow.co/2022/#section-most-popular-technologies-integrated-development-environment (Erişim: 06 Haziran 2023).

Suhail, K. ve Brindha, D. (2021) "A Review on Various Methods for Recognition of Urine Particles Using Digital Microscopic Images of Urine Sediments", *Biomedical Signal Processing and Control*, 68, 102806.

Şeker, A., Diri, B. ve Balık, H. H. (2017) "Derin Öğrenme Yöntemleri Ve Uygulamaları Hakkında Bir İnceleme", *Gazi Mühendislik Bilimleri Dergisi (GMBD)*, 3(3), 47-64.

Thuan, D. (2021) *Evolution of Yolo Algorithm and Yolov5: The State-of-the-Art Object Detection Algorithm*.

Wang, C., Bochkovskiy, A. ve Liao, H. M. (2023) "YOLOv7: Trainable Bag-of-freebies Sets New State-of-the-art for Real-time Object Detectors", *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7464-7475.

Wang, C.-Y., Bochkovskiy, A. ve Liao, H.-Y. M. (2021) "Scaled-yolov4: Scaling cross stage partial network", *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, 13029-13038.

Wang, C.-Y., Yeh, I.-H. ve Liao, H.-Y. M. (2021) "You Only Learn One Representation: Unified Network for Multiple Tasks", *arXiv preprint arXiv:2105.04206*.

Xiang, H., Chen, Q., Wu, Y., Xu, D., Qi, S., Mei, J., Li, Q. ve Liu, X. (2019) "Urine Calcium Oxalate Crystallization Recognition Method Based on Deep Learning", *2019 International Conference on Automation, Computational and Technology Management, ICACTM 2019*, 30-33.

Yasak, S. S. (2021) *Coğrafyada Yapay Zeka Uygulamaları: YOLO V3 ile Gerçek Zamanlı Kayaç Tespit Uygulaması Örneği*.

Ye, G., Qu, J., Tao, J., Dai, W., Mao, Y. ve Jin, Q. (2023) "Autonomous Surface Crack Identification of Concrete Structures Based on the YOLOv7 Algorithm", *Journal of Building Engineering*, 73, 106688.

Zhang, X., Jiang, L., Yang, D., Yan, J. ve Lu, X. (2019) "Urine Sediment Recognition Method Based on Multi-View Deep Residual Learning in Microscopic Image", *Journal of Medical Systems*, 43(11), 1-10.

EKLER

EK-1 Çalışmada kullanılan Docker dosyası

```
FROM nvcr.io/nvidia/pytorch:21.08-py3

RUN apt update \
        && apt install -y zip htop screen libgl1-mesa-glx

RUN apt upgrade --no-install-recommends -y openssl tar

RUN python3 -m pip install --upgrade pip wheel
RUN pip install --no-cache seaborn thop

WORKDIR /data
CMD ["./run.sh"]
```

EK-2 Birleştirilmiş veri seti ile yapılan bazı çalışmaların detayları

Sıra	Model	Parametreler	Veri Arttırma	Epok	Grup Boyutu	Görsel Boyutu
1	YOLOv7-tiny	lr0: 0.01 lrf: 0.01 momentum: 0.937 weight_decay: 0.0005 warmup_epochs: 3.0 warmup_momentum: 0.8 warmup_bias_lr: 0.1 box: 0.05 cls: 0.5 cls_pw: 1.0 obj: 1.0 obj_pw: 1.0 iou_t: 0.2 anchor_t: 4.0 fl_gamma: 0.0 optimizer: Adam	hsv_h: 0.015 hsv_s: 0.7 hsv_v: 0.4 degrees: 0.0 translate: 0.1 scale: 0.5 shear: 0.0 perspective: 0.0 flipud: 0.5 fliplr: 0.5 mosaic: 1.0 mixup: 0.0 copy_paste: 0.0 paste_in: 0.0 loss_ota: 0	150	32	800
2	YOLOv7	lr0: 0.01 lrf: 0.01 momentum: 0.937 weight_decay: 0.0005 warmup_epochs: 3.0 warmup_momentum: 0.8 warmup_bias_lr: 0.1 box: 0.05 cls: 0.5 cls_pw: 1.0 obj: 1.0 obj_pw: 1.0 iou_t: 0.2 anchor_t: 4.0 fl_gamma: 0.0 optimizer: Adam	hsv_h: 0.015 hsv_s: 0.7 hsv_v: 0.4 degrees: 0.0 translate: 0.1 scale: 0.5 shear: 0.0 perspective: 0.0 flipud: 0.5 fliplr: 0.5 mosaic: 1.0 mixup: 0.0 copy_paste: 0.0 paste_in: 0.0 loss_ota: 0	150	2	800
3	YOLOv7-tiny	lr0: 0.01 lrf: 0.3 momentum: 0.937 weight_decay: 0.0005 warmup_epochs: 5.0 warmup_momentum: 0.8 warmup_bias_lr: 0.1 box: 0.05 cls: 0.3 cls_pw: 1.0 obj: 1.0 obj_pw: 1.0 iou_t: 0.2 anchor_t: 4.0 fl_gamma: 0.0 optimizer: Adam	hsv_h: 0.015 hsv_s: 0.7 hsv_v: 0.4 degrees: 0.0 translate: 0.3 scale: 0.5 shear: 0.0 perspective: 0.0 flipud: 0.5 fliplr: 0.5 mosaic: 1.0 mixup: 0.0 copy_paste: 0.0 paste_in: 0.0 loss_ota: 0	150	50	640
4	YOLOv7-tiny	lr0: 0.006 lrf: 0.3 momentum: 0.937 weight_decay: 0.0005 warmup_epochs: 5.0	hsv_h: 0.015 hsv_s: 0.7 hsv_v: 0.4 degrees: 0.0 translate: 0.3	150	50	640

		warmup_momentum: 0.8 warmup_bias_lr: 0.1 box: 0.05 cls: 0.3 cls_pw: 1.0 obj: 1.0 obj_pw: 1.0 iou_t: 0.2 anchor_t: 4.0 fl_gamma: 0.0 optimizer: Adam	scale: 0.5 shear: 0.0 perspective: 0.0 flipud: 0.5 fliplr: 0.5 mosaic: 1.0 mixup: 0.0 copy_paste: 0.0 paste_in: 0.0 loss_ota: 0			
5	YOLOv7-tiny	lr0: 0.006 lrf: 0.3 momentum: 0.937 weight_decay: 0.0005 warmup_epochs: 5.0 warmup_momentum: 0.8 warmup_bias_lr: 0.1 box: 0.05 cls: 0.3 cls_pw: 1.0 obj: 1.0 obj_pw: 1.0 iou_t: 0.2 anchor_t: 4.0 fl_gamma: 0.0 optimizer: SGD	hsv_h: 0.015 hsv_s: 0.7 hsv_v: 0.4 degrees: 0.0 translate: 0.3 scale: 0.5 shear: 0.0 perspective: 0.0 flipud: 0.5 fliplr: 0.5 mosaic: 1.0 mixup: 0.0 copy_paste: 0.0 paste_in: 0.0 loss_ota: 0	150	50	640