MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK ÇİZGİ KARAKTER TESPİTİ

Candan TÜMER
17 14 50 104

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Doktora Programı
Danışman: Dr. Öğr. Üyesi Erdal GÜVENOĞLU

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Şubat, 2024

MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK ÇİZGİ KARAKTER TESPİTİ

Candan TÜMER
17 14 50 104
ORCID: 0000-0002-9592-7110

DOKTORA TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Doktora Programı
Danışman: Dr. Öğr. Üyesi Erdal GÜVENOĞLU
İkinci Danışman: Doç. Dr. Volkan TUNALI

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Şubat, 2024



JÜRİ VE ENSTİTÜ ONAYI

CANDAN TÜMER'ın "Makine Öğrenmesi Yöntemleri Kullanılarak Çizgi Karakter Tespiti" başlıklı tezi 08.02.2024 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Maltepe Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği" nin ilgili maddeleri uyarınca Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans/Doktora/Sanatta Yeterlik tezi oy birliğiyle/oy çokluğuyla, başarılı/başarısız olarak kabul edilmiştir.

	Unvanı. Adı ve Soyadı	İmza
Üye (Tez Danışmanı)	Dr.Öğr.Üyesi Erdal GÜVENOĞLU (Danışman) Maltepe Üniversitesi	.....
Üye	Dr.Öğr.Üyesi Engin OĞUZAY Maltepe Üniversitesi	.....
Üye	Dr.Öğr.Üyesi Safinaz Sibel ERSAN Maltepe Üniversitesi	.....
Üye	Dr.Öğr.Üyesi Emin BORANDAĞ Celal Bayer Üniversitesi	.....
Üye	Dr.Öğr.Üyesi Fatih YÜCALAR Celal Bayer Üniversitesi	.....

Prof. Dr. Selva ERSÖZ KARAKULAKOĞLU
Enstitü Müdürü

ETİK İLKE VE KURALLARA UYUM BEYANI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" ile bildirilen 6698 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

TEŞEKKÜR

Doktora eğitim sürecimdeki katkılarından ve danışmandan çok bir baba sıcaklığı hissettiğim yol gösterici danışmanım Sn. Dr. Erdal GÜVENOĞLU'na ve bu hayatta var olduğuna binlerce kez şükrettiğim, fikirleri, eleştirileri ve verdiği tüm hayat dersleriyle her seferinde beni ayağa kaldıran ikinci danışmanım Sn. Doç. Dr. Volkan TUNALI'ya teşekkürlerimi ve saygılarımı sunarım. Doktora tezimde Kral Şakir çizgi film karakterleri üzerinde çalışma izni veren ve bilimi destekleyen ilk iletişim kurduğum andan itibaren bana çok samimi yaklaşan Sn. Varol YAŞAROĞLU'na teşekkürlerimi ve saygılarımı sunarım. Kral Şakir çizgi film karakterlerinin yasal izinleri hususunda yardımlarını ve desteğini esirgemeyen Sn. Özlem CİVELEK'e içten teşekkürlerimi sunarım. Tüm eğitim hayatım boyunca benden bilgisini ve desteğini esirgemeyen Sn. Prof. Dr. Aslıhan TÜFEKCİ ve Sn. Prof. Dr. Gürkan DOĞAN hocalarıma çok teşekkür ederim. Çalışmalarım esnasında ilgisini ve önerilerini göstermekten kaçınmayan Öğrt. Vedat TOKMAK'a, her aradığımda telefonun diğer ucunda olan Dr. Esin ERGUVAN ETGİN'e, model oluşturmadaki katkılarından dolayı Oğuzhan GÜRBÜZ'e ve manevi destekleriyle beni motive eden Öğr. Gör. Dr. Çağla Gizem AKKAŞ'a teşekkür ederim. İyi bir insanın hayatınıza etkisi paha biçilemez sözünün vücut bulmuş hali olan canım arkadaşım Kübra KAVALCI'ya gece gündüz yanımda olduğu için teşekkürlerimi sunarım. Canımdan çok sevdiğim annem Sacide TURĞUT'a ve babam Mehmet YILDIZ'a, her zaman bana güvenen canım ablalarım Sema SERİN ve Canan SARI'ya, tez yazım sürecimde beni cesaretlendiren ve her daim yanımda olan eşim Harun TÜMER'e ve kendisinden çaldığım her vakit için oğlum Rüzgar TÜMER'e teşekkürü borç bilirim.

Candan TÜMER

Şubat, 2024

ÖZET

MAKİNE ÖĞRENMESİ YÖNTEMLERİ KULLANILARAK ÇİZGİ KARAKTER TESPİTİ

Candan Tümer
Doktora Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Doktora Programı
Danışman: Dr. Öğr. Üyesi Erdal Güvenoğlu
Maltepe Üniversitesi Lisansüstü Eğitim Enstitüsü, 2024

Çizgi karakterler, süper kahramanlar ve karikatür karakterleri hem çizgi filmlerde hem de çizgi romanlarda kullanılmaktadır. Bu karakterler yaratıcılarının hayal dünyalarına ya da hikayenin gidişatına göre farklı şekillerde çizilebilmektedirler. Bu nedenle makine öğrenmesinde çizgi karakter tespiti, insan tespitinden farklıdır ve otomatik olarak algılanması ve tespit edilmesi zordur. Çizgi karakterlerin otomatik olarak algılanması ve tanınması son yıllarda önemli derecede ilgi görmektedir. Bu çalışmada makine öğrenmesi tekniklerinin kullanılmasıyla çizgi karakterlerin yüksek doğrulukla tespit edilmesi ve yeni bir çizgi karakter veri setinin literatüre kazandırılması amaçlanmıştır. Araştırmada makine öğrenmesi teknikleriyle Evrişimli Sinir Ağları tabanlı görüntü sınıflandırması ve nesne tespiti yapılmıştır. Görüntü sınıflandırma yöntemi için çoklu etiketleme kullanılarak uçtan uca yeni bir mimari tasarlanmıştır. Nesne tespiti yöntemi için de YOLOv8 kullanılarak karakter tespiti yapılmıştır. Oluşturulan KralSakir çizgi karakter veri setinin en önemli özelliği ulusal olarak ilk defa ülkemizde hazırlanmasıdır. KralSakir veri seti üzerinde çalışılan modeller, performans ölçüm yöntemleri, analiz edilen metriklerin sonuçları ayrıntılı bir şekilde açıklanmıştır. Modellerin çıktıları değerlendirilerek görselleştirilmiş ve iki model arasındaki farklara değinilmiştir. Deneysel sonuçlarda, KralSakir veri setinde her iki yöntemin çizgi karakter tanımada başarılı olduğu görülmüştür.

Anahtar Sözcükler: Çizgi Karakter Tanıma, Uçtan Uca Mimari, Makine Öğrenmesi, Çoklu Sınıflandırma, Nesne Tespiti, Derin Öğrenme.

ABSTRACT

DETECTION OF COMIC CHARACTER USING MACHINE LEARNING METHODS

Candan Tümer
PhD Thesis
Department of Computer Engineering
PhD Program in Computer Engineering
Thesis Advisor: Assist. Prof. Dr. Erdal Güvenoğlu
Maltepe University Graduate School, 2024

Cartoon characters, superheroes and caricature characters are used in both cartoons and comics. These characters can be drawn in different ways according to the imagination of their creators or the course of the story. For this reason, cartoon character detection in machine learning is different from human detection and is difficult to automatically detect and recognize. The automatic detection and recognition of cartoon characters has received considerable attention in recent years. In this study, it is aimed to detect cartoon characters with high accuracy using machine learning techniques and to introduce a new cartoon character dataset to the literature. In the research, Convolutional Neural Networks based image classification and object detection were performed with machine learning techniques. For the image classification method, a new end-to-end architecture was designed using multiple labeling. For the object detection method, character detection was performed using YOLOv8. The most important feature of the KralSakir cartoon character dataset is that it is prepared nationally for the first time in our country. The models studied on the KralSakir dataset, performance measurement methods, and the results of the analyzed metrics are explained in detail. The outputs of the models are evaluated and visualized and the differences between the two models are mentioned. Experimental results show that both methods are successful in cartoon character recognition on the KralSakir dataset.

Keywords: Comic Character Detection, End-To-End Architecture, Machine Learning, Multi-Classification, Object Detection, Deep Learning.

İÇİNDEKİLER

JÜRİ VE ENSTİTÜ ONAYI ii
ETİK İLKE VE KURALLARA UYUM BEYANI iii
TEŞEKKÜR iv
ÖZET v
ABSTRACT vi
İÇİNDEKİLER vii
TABLOLAR LİSTESİ xi
ŞEKİLLER LİSTESİ xii
KISALTMALAR xvi
1. GİRİŞ 1
1.1 Problem 2
1.2 Amaç 3
1.3 Önem 3
1.4 Varsayımlar 4
1.5 Sınırlıklar 4
1.6 Tanımlar 4
2. LİTERATÜR ARAŞTIRMASI 6
3. MATERYAL VE YÖNTEM 19
3.1 Yapay Sinir Ağları 19
3.2 Makine Öğrenmesi 22
3.3 Derin Öğrenme 23
3.4 Evrişimli Sinir Ağları 26

3.4.1 Evrişimli katmanlar 28
3.4.2 Havuzlama katmanları 29
3.4.3 Tam bağlı katmanlar 30
3.4.3.1 Aktivasyon katmanı 31
3.4.3.2 Normalleştirme katmanı 31
3.4.3.3 Düğüm seyreltme katmanı 32
3.4.3.4 Düzleştirme katmanı 33
3.4.3.5 Yoğun katman 34
3.5 Denetimli Öğrenme 35
3.6 Transfer Öğrenme 36
3.7 Hiper-Parametreler 37
3.7.1 Katman boyutu 37
3.7.2 Mini-yığın 38
3.7.3 Optimizasyon algoritmaları 38
3.7.4 Öğrenme oranı 39
3.7.5 L1-L2 düzenlileştirmesi 40
3.7.6 Eğitim tur sayısı 41
3.8 Değerlendirme Metrikleri 42
3.8.1 Karmaşıklık matrisi 42
3.8.2 Doğruluk 43
3.8.3 Kesinlik 43

3.8.4 Duyarlılık, geri çağırma 43
3.8.5 F1 skoru 44
3.8.6 Eğri altında kalan alan 44
3.8.7 mAP değeri 45
3.9 Yetersiz Uyum, Aşırı Uydurma ve İyi Form 45
3.10 Data Augmentation 50
3.11 Uçtan – Uca Mimari 51
3.12 Çok Sınıflı Görüntü Sınıflandırma 51
3.13 Nesne Tespiti 52
3.14 KralSakir Veri Setinin Özellikleri 54
3.15 Çizgi Karakter Tanıma Yöntemi Oluşturulması 60
3.15.1 Çok etiketli görüntü sınıflandırma yöntemi 60
3.15.1.1 Mimarinin kodlanması 61
3.15.1.2 Veri setinin modele tanıtılması 62
3.15.1.3 Eğitim 65
3.15.1.4 Sonuçların görselleştirilmesi 65
3.15.1.5 Sınıfların tahmin edilmesi 66
3.15.2 YOLOv8 ile nesne tespiti yöntemi 66
3.15.2.1 Mimarinin kodlanması 66
3.15.2.2 Veri setinin modele tanıtılması 67
3.15.2.3 Eğitim 71
3.15.2.4 Sonuçların görselleştirilmesi 72

3.15.2.5 Sınıfların tahmin edilmesi 72

4. BULGULAR VE TARTIŞMA 74

4.1 Çizgi Karakter Tanıma İçin Önerilen Çok Sınıflı Görüntü Sınıflandırma Yöntemi 74

4.1.1 Yöntemin mimarisi 74

4.1.2 Yöntemin eğitimi 77

4.1.3 Yöntemin değerlendirme sonuçları 77

4.1.4 Yöntemin tahmin sonuçları 82

4.2 YOLOv8 ile Nesne Tespiti Yöntemi 84

4.2.1 Yöntemin mimarisi 84

4.2.2 Yöntemin eğitimi 85

4.2.3 Yöntemin değerlendirme sonuçları 85

4.2.4 Yöntemin tahmin sonuçları 91

4.3 Yöntemlerin Karşılaştırılması ve KralSakir Veri Seti Üzerinden Değerlendirilmesi 92

5. SONUÇ ve ÖNERİLER 95

KAYNAKLAR 98

ÖZGEÇMİŞ 109

TABLOLAR LİSTESİ

Tablo 1. Literatür Taramasının Özeti 16

Tablo 2. KralSakir Veri Seti Sınıfları 55

Tablo 3. KralSakir Veri Seti Çizgi Karakter Sayıları 55

Tablo 4. Model Performans Tablosu 77

Tablo 5. Sınıflara Göre Test Seti Metrik Değerleri 82

Tablo 6. Eğitim Seti Sonuçları 93

Tablo 7. Doğrulama Seti Sonuçları 93

Tablo 8. Sonuçları Alınan Metrikler 94

ŞEKİLLER LİSTESİ

Şekil 1. Algılayıcı 20

Şekil 2. Yapay Sinir Ağı 20

Şekil 3. Yapay Zeka, Makine Öğrenmesi ve Derin Öğrenmenin İlişkisi 22

Şekil 4. Yapay Sinir Ağ Yapıları 25

Şekil 5. ESA Mimarisi Tüm Katmanları 27

Şekil 6. Evrişim Katmanı 29

Şekil 7. Havuzlama Katmanı 30

Şekil 8. Aktivasyon Katmanı 31

Şekil 9. Normalleştirme Katman Türleri 32

Şekil 10. Düğüm Seyreltme Katmanı 33

Şekil 11. Düzleştirme Katmanı 34

Şekil 12. Yoğun Katman 35

Şekil 13. Denetimli Öğrenme 36

Şekil 14. Karmaşıklık Matrisi 42

Şekil 15. Yetersiz Uyum Gösteren Öğrenme Eğrisi 46

Şekil 16. Daha Fazla Eğitim Gerektiğini Gösteren Öğrenme Eğrisi 46

Şekil 17. Aşırı Uydurma Gösteren Öğrenme Eğrisi 47

Şekil 18. İyi Uyum Gösteren Öğrenme Eğrisi 48

Şekil 19. Doğrulama Veri Kümesine Göre Çok Küçük Olabilecek Bir Eğitim Veri Kümesini Gösteren Öğrenme Eğrisi 49

Şekil 20. Eğitim Veri Kümesine Göre Çok Küçük Olabilecek Bir Doğrulama Veri Kümesini Gösteren Öğrenme Eğrisi 50

Şekil 21. Eğitim Veri Kümesinden Daha Kolay Tahmin Edilen Doğrulama Veri Kümesini Gösteren Öğrenme Eğrisi 50

Şekil 22. ESA Görüntü Sınıflandırma Mimarisi 52

Şekil 23. YOLO Mimarisi 53

Şekil 24. YOLO Tespit Sistemi 54

Şekil 25. Çizgi Karakterlerin Etiketlenen Görüntü Sayı Dağılım Grafiği 56

Şekil 26. Veri Setinden Rastgele Seçilen Görüntüler 57

Şekil 27. Kral Şakir Karakteri Görselleri 58

Şekil 28. Remzi Karakteri Görselleri 58

Şekil 29. Necati Karakteri Görselleri 59

Şekil 30. Canan Karakteri Görselleri 59

Şekil 31. Oluşturulan CSV Dosyası 63

Şekil 32. Veri Seti Kontrol Satırları 64

Şekil 33. YOLOv8 Görev Dosyaları 67

Şekil 34. Makesense.ai Sitesi Proje Ekranı 68

Şekil 35. Makesense.ai Sitesi Etiket Oluşturma İşlemi 68

Şekil 36. Makesense.ai Sitesi Etiketleme İşlemi 69

Şekil 37. Etiketlemesi Yapılan Görsellerin Dışarı Aktarımı 69

Şekil 38. Etiketleme Sonrası Sınıflarıyla Birlikte Oluşturulan .txt Belgesi 70

Şekil 39. Dosyalara Uygun Şekilde Etiketlerin Eklenmesi 70

Şekil 40. YAML Dosyası 71

Şekil 41. Detect Klasöründe Sonuçların Görselleştirilmesi 72

Şekil 42. Oluşturulan Modelin Mimarisi 75

Şekil 43. Model Özeti 76

Şekil 44. Eğitim Sırasında Ekrana Yansıyan Değerler 77

Şekil 45. Ortalama Metrik Sonuçları 78

Şekil 46. Metriklerin Sonuçlarının Eğitim Boyunca Değerleri 78

Şekil 47. Tüm Metriklerin Grafiksel Gösterimi 80

Şekil 48. Karmaşıklık Matrisi 81

Şekil 49. Görüntü Sınıflandırma Tahmin Sonuçları 1 83

Şekil 50. Görüntü Sınıflandırma Tahmin Sonuçları 2 83

Şekil 51. Model Özeti 84

Şekil 52. Modelin Eğitimi 85

Şekil 53. Metrik Sonuçlarının CSV Dosyası 86

Şekil 54. F1 Skor Grafiği 87

Şekil 55. Kesinlik Grafiği 87

Şekil 56. Duyarlılık Grafiği 88

Şekil 57. Kesinlik - Duyarlılık Grafiği 88

Şekil 58. Karmaşıklık Matrisi 89

Şekil 59. Kayıp Değerleri Grafikleri 90

Şekil 60. Nesne Tanıma Tahmin Sonuçları 1 91

Şekil 61. Nesne Tanıma Tahmin Sonuçları 2 91

KISALTMALAR

AdaGrad : Adaptive Gradient Algorithm, Uyarlamalı Gradyan Algoritması

Adam : Adaptive Moment Estimation, Uyarlanabilir Moment Tahmini

AUC : Area Under Curve, Eğri Altındaki Alan

AP : Average Precision, Ortalama Kesinlik

BP : Back Propagation, Geri Yayılım

CAST : Character labeling in Animation using Self-supervision by Tracking, İzleme ile Özdenetim Kullanarak Animasyonda Karakter Etiketleme

CSV : Comma Separated Values, Virgülle Ayrılmış Değerler

COCO : Common Objects in Context, Bağlam İçinde Ortak Nesneler

ÇKA : Multilayer Perceptron, Çok Katmanlı Algılayıcılar

ESA : Convolutional Neural Networks, Evrişimli Sinir Ağı

FDM : Fuzzy Diffusion Maps, Bulanık Difüzyon Haritaları

FN : False Negative, Yanlış Negatif

FP : False Positive, Yanlış Pozitif

FSM : Frequent Subgraph Mining, Sık Alt Grafik Madenciliği

GPU : Graphics Processing Unit, Grafik İşlemci Birimi

IoU : Intersection over Union, Birlik Üzerinde Kesişim

LPP : Locality Preserving Projections, Yerelliği Koruyan Projeksiyonlar

LR : Learning Rate, Öğrenme Oranı

mAP **:** Mean Average Precision, Ortalama Kesinlik

MTL **:** Multi-Task Learning, Çoklu Görev Öğrenimi

QBE **:** Query By Example, Örnekle Sorgulama

R-CNN **:** Region with Convolutional Neural Network, Bölgesel Tabanlı ESA

RAG **:** Region Adjacency Graphs, Bölge Bitişiklik Grafikleri

RAM **:** Random Access Memory, Rasgele Erişimli Bellek

ReLU **:** Rectified Linear Unit, Düzeltilmiş Lineer Birim

RMSprop **:** Root Mean Squared Propagation, Karesel Ortalamanın Karekök Yayılımı

ROC **:** Receiver Operating Characteristic Curve, Alıcı İşletim Karakteristik Eğrisi

SGD **:** Stochastic Gradient Descent, Stokastik Gradyan İnişi

SSD **:** Single Shot MultiBox Detector, Tek Atışlı MultiBox Dedektör

SIFT **:** Scale-Invariant Feature Transform, Ölçekle Değişmeyen Özellik Dönüşümü

TP **:** True Positive, Doğru Pozitif

TN **:** True Negative, Doğru Negatif

YSA **:** Artificial Neural Network, Yapay Sinir Ağı

YOLO **:** You Only Look Once, Sadece Bir Kez Bakarsın

YZ **:** Artificial Intelligent, Yapay Zeka

1. GİRİŞ

Bilgisayarla görme, disiplinler arası bir bilim alanıdır ve insanların sahip olduğu görsel sisteme benzer bir yaklaşımı kullanarak görsel verilerin detaylı bir şekilde anlaşılması için bilgisayarların kullanılmasını içermektedir. Bu yapay zekanın (YZ) çalışma alanlarından biridir ve uygulamaları çeşitli problemleri ele almaktadır (Xu, Wang, Shou, Sadick, ve Wang, 2021). 1950'li yıllardan itibaren gelişen YZ aslında insanlar gibi düşünen makineler yapma bilimidir (Chollet, 2021). YZ teknolojisi büyük miktardaki veriyi insanlardan farklı ve çok daha kısa sürede işleyebilme yetisine sahiptir. Öğrenebilen bir algoritmaya sahip olan YZ, bizim varlığımıza en az şekilde ya da hiç ihtiyaç duymadan kendi kendine karar verebilen ve kendini sürekli iyileştirebilen sistemler ya da makinelerdir. YZ'nin çalışma alanlarından biri olan makine öğrenmesi; algoritmalar ve eğitimler aracılığıyla deneyimleyerek öğrenebilen tahmine dayalı bir analiz türüdür (Sharma, Sharma, ve Neeru, 2021). Makine öğrenmesinin alt çalışma alanı olan derin öğrenme ise insan beyni ve düşünme sisteminin taklit edilerek veri kümelerinden nasıl sonuç çıkaracağına yönelik bir analiz türüdür. Derin öğrenme algoritmalarında birçok sınıflandırma modeli bulunmaktadır ve en sık aynı zamanda en popüler olarak kullanılanı evrişimli sinir ağlarıdır (ESA). ESA'lar görsel sınıflandırma problemlerinde güçlü bir yapay sinir ağı (YSA) tekniğidir (Chollet, 2021). ESA'lar el yazısı, rakam ve karakter tanıma gibi nesne tespiti görevleri için geliştirilmiştir. ESA'nın araştırmalarda daha çok kullanılmasının nedeni, otomatik bir özellik çıkarıcı olarak çalışmasıdır. ESA, verilerdeki kenarlar gibi düşük seviyeli özelliklerden yüzler ve nesneler gibi daha yüksek seviyeli özelliklere kadar olası tüm özellikleri çıkarabilmektedir. Özellikler, belirli bir nesne sınıfının diğer nesne sınıflarından farklı olan özelliklerinin yanı sıra bu özelliklerin ölçüm veya işleme yoluyla çıkarılabilen koleksiyonudur. Görüntü özelliği çıkarma; veri madenciliği, bilgisayarla görme ve tıbbi teşhis gibi birçok alanda araştırmacıların ilgisini çekmektedir (Sunitha, Sharmila, ve Srinivasan, 2019). Görüntü özelliklerinden görüntü tanıma aslında bir sınıflandırma işlemidir. Sınıflandırma, makine öğrenimi algoritması tarafından yapılan gözlemlerden sonuçlar çıkararak verileri kategorize etme işlemidir. Ayrıca bu işlemden sonra gelen yeni verilerin hangi kategoriye ait olduğunun belirlenmesine de yardımcı olmaktadır.

Bu çalışmada çizgi karakter sınıflandırması yapılmış ve yeni bir veri seti hazırlanmıştır. Geliştirilen modellerin verimli bir şekilde çalıştığı deneysel sonuçlarla gösterilmiştir. Kral Şakir (Yaşaroğlu, 2023) çizgi film karakterlerinin yaratıcısı Ulusal Sanatçımız Varol Yaşaroğlu (Yaşaroğlu, 2023) tarafından yaratmış olduğu çizgi karakterler üzerinde çalışılmasına izin verilmiş ve yapılan bu bilimsel çalışma desteklenmiştir.

Çizgi karakterleri tanıma (tespiti) görevi için ulusal çizgi film karakterlerine yönelik ilk defa bir veri seti oluşturulmuştur. Sınıflandırma problemi için yeni bir model geliştirilerek uçtan uca (end-to-end) bir mimari oluşturulmuş aynı zamanda nesne tanıma yöntemi de çalışılmıştır.

Çalışmanın devamında literatür incelemesi yapılmış; var olan modeller ve kullanılan tekniklerden bahsedilmiştir. Daha sonra çalışmada önerilen modeller, performans ölçüm yöntemleri, analiz edilen metriklerin sonuçları ayrıntılı bir şekilde açıklanmıştır. Çalışmanın bulgular kısmında modellerin çıktıları değerlendirilerek görselleştirilmiş ve iki yöntem arasındaki farklara değinilmiştir.

1.1 Problem

Çizgi karakterler hem çizgi filmlerde hem de çizgi romanlarda kullanılmaktadır. Bu karakterler yaratıcılarının hayal dünyalarına ya da hikayenin gidişatına göre farklı şekillerde çizilebilmektedirler. Makine öğrenmesinde çizgi karakter tespiti, insan tespitinden farklıdır ve otomatik olarak algılanması ve tespit edilmesi zor olduğu için insan tespitine dayalı yöntemler doğrudan çizgi karakterlere uygulanamamaktadır (Rigaud, Burie, Ogier, ve Karatzas, 2014). Bunun yanı sıra veri seti oluşturulması çok vakit alan ve zahmetli bir iş olduğundan dolayı çizgi karakterlere ait oluşturulmuş veri setlerinin sayısı oldukça sınırlıdır. Bu sebepler sonucunda çizgi karakter yüz algılama veya karakter tespiti konusunda az sayıda çalışma yapılmaktadır. Ayrıca bu karakterlerin yaratıcıları tarafından çalışmalarda karakterlerin telif haklarından kaynaklı kullanılmasına izin verilmemesi, nitelikli çalışma yapılması ya da yeni bir yöntem denenmesi önünde ciddi bir engel teşkil etmektedir. Ülkemizde buna yönelik bilimsel çalışmalar yapılacak bir veri seti bulunmamaktadır. Çalışmada oluşturulan KralSakir çizgi karakter veri setinin en önemli özelliği ulusal olarak ilk defa ülkemizde hazırlanmasıdır.

1.2 Amaç

Bu çalışmada yeni bir çizgi karakter veri seti hazırlanması ve literatüre kazandırılması amaçlanmaktadır. Aynı zamanda makine öğrenmesi teknikleriyle uçtan uca bir mimari tasarlanmış ve nesne tespiti de çalışılarak çizgi karakter tanımanın zorlu problemlerine farklı bir bakış açısıyla çözüm kazandırılması amaçlanmıştır. Araştırma, bilgisayarla görme alanındaki çalışmalara katkı sağlayarak alan araştırması yapılmasının kolaylaştırılmasının yanı sıra ülkemizde ilk kez yapıldığından dolayı daha sonra yapılacak çalışmalara örnek teşkil etmeyi de amaçlamaktadır.

1.3 Önem

Günümüzde bilgisayarla görü teknolojisi; biyometrik tarayıcılar, yüz tanıma teknolojileri, artırılmış gerçeklik, sağlık ekipmanları, tarım ve üretim gibi birçok alanda kullanılmaktadır. Çalışmanın önemi; bilgisayarla görme, görüntü işleme ve multimedya analiz sistemlerinin alandaki araştırmalarına katkı sağlayacak olmasıdır. Çizgi karakter endüstrisindeki gelişmelerle birlikte çizgi karakter tanımlamada bilgisayarla görü teknolojisi tekniklerinin yaygın bir şekilde kullanılması bu alandaki çalışmalara duyulan ihtiyacı artırmıştır. Literatüre bakıldığında çizgi karakter tanıma ile ilgili sınırlı sayıda çalışma vardır ve daha önce Türkiye'de çizgi karakterlerle ilgili herhangi bir çalışmaya rastlanılmamıştır. Bu nedenle çalışma önemlidir. Aynı zamanda çalışma daha sonra yapılacak olan çalışmalarda aşağıdaki ihtiyaçların giderilmesine yönelik bir başlangıç noktası olması açısından önemlidir.

- Benzer çizgi karakterlerin webde arama yapmak için resim arama motorlarına dahil edilmesi.
- Görme engelli kişilerin çizgi filmleri anlamasına yardımcı olmak için ekran okuyucularla entegrasyonu.
- İçerik kontrol yazılımlarının sosyal medyadaki uygunsuz çizgi film görüntülerini sansürlemesine yardımcı olmak.
- Çizgi karakterlerin telif hakkı açısından korunması.

1.4 Varsayımlar

Bu tezde kullanılan yöntemlerden çıktıları literatürde var olan metriklere göre değerlendirilmiştir. Yöntemler oluşturulan veri seti üzerinde çalıştırılmış ve deneysel sonuçların çıktılarına göre başarılı oranlar elde edilene kadar düzenlenmiştir.

1.5 Sınırlıklar

Çizgi karakter ya da karikatür tanıma çalışmalarındaki en önemli kısıt karakterin yaratıcıları tarafından telifle korunuyor olmasıdır. Sanatçılardan karakterlerinin kullanımı için izin almak aşılması kolay olmayan engellerden biridir. Bu çalışmada tek bir çizgi filme bağlı karakterlerden yalnızca dört tanesi ile çalışılmıştır. Çok etiketli sınıflama yöntemi için ayrı, nesne tespiti yöntemi için de ayrı tekniklerle hazırlanan iki virgülle ayrılmış değerler (CSV) dosyası oluşturulmuştur. Yani veri seti üzerinde iki ayrı yöntemle iki ayrı çalışma hazırlığı yapılmıştır. Üzerinde çalışılmış olan veri seti uzun bir zaman aralığında toplanmış ve karakter sayıları belirli bir doygunluğa ulaştıktan sonra model oluşturulmaya geçilmiştir. Sınıf sayılarının kısıtlı olması veri seti oluşturmak için harcanan zamanla orantılıdır. Modelin eğitilmesi aşaması da uzun ve zorlu bir süreçtir. Epok (iterasyon/eğitim tur sayısı) sayıları artıkça eğitim için ayrılan süre ile ilgili bir uzama mevcuttur. Daha fazla iyileşmenin olmadığı epok sayıları ile modeller sınırlı tutulmuştur. Görüntü analizinde kullanılan teknolojilerin her bakımdan iyi olmak zorunda olması da bir kısıttır. Yüksek RAM (Random Access Memory) ve GPU (Graphics Processing Unit) desteği ile çalıştırılan model maliyet açısından yüksek rakamlara mal olmaktadır. Google Colab'te çalışılmış hatta pro sürümü ve plus sürümü denenmiştir fakat ihtiyacın belirli bir ölçüde karşılandığı bir ortamı bulunmaktadır. Çalışmanın ilerlemesi yönünde Google Colab'ten RAM ve GPU kullanma desteği ücret karşılığında alınmış fakat o da yeterli olmamıştır.

1.6 Tanımlar

Bu tezde kullanılan yöntemlerden birisi çok etiketli görüntü sınıflandırmadır. Çok etiketli görüntü sınıflandırma (multi-label image classification) her bir örneğe birden fazla etiketin atanabileceği sınıflandırma probleminin bir çeşididir. Çok sınıflı sınıflandırma problemi olarak analiz edilen veri seti için yeni bir model oluşturulmuştur. Aynı zamanda nesne tanıma modeli de çalışılmıştır. Nesne tanıma, dijital görüntülerde belirli bir

sınıftaki görsel nesnelerin (insanlar, hayvanlar veya arabalar gibi) örneklerini tespit etmekle ilgilenen önemli bir bilgisayarla görme görevidir. Çalışılan her iki yöntemin çıktıları değerlendirilmiştir.

2. LİTERATÜR ARAŞTIRMASI

Çizgi karakterlerin tespiti, literatürde var olan karakter analizi yöntemlerinden bağımsız bir şekilde incelenmiştir. Literatür incelendiğinde bazı araştırmacıların kısıtlı veri setleri üzerinde çeşitli yöntemlerle çalıştığı ve birçok model önerdiği gözlemlenmiştir. Aynı zamanda bazı araştırmacıların az da olsa kendi veri setini hazırlayarak literatüre kazandırdığı çalışmalar mevcuttur. Bu bölümde çizgi karakter tanıma konusuyla ilgili literatürde yer alan çalışmalar araştırılarak yıllara göre geniş bir özet sunulmaya çalışılmıştır.

Sun ve Kise (2010;2013) yaptıkları çalışmalarla, Viola–Jones (2004) tespit çerçevesinin yerel özelliklerini kullanarak çizgi roman (Japon çizgi romanları) sayfalarındaki karakterleri ve yüzleri tespit etmek için yeterli olduğunu kanıtlamışlardır (Sun ve Kise, 2010; Sun ve Kise, 2013; Viola ve Jones, 2004).

Sun ve Kise (2010) karikatür yüzlerinin görüntülü yayınlarda telif hakkının korunması konusunda bir çalışma yapmışlardır. Çalışmalarında tek renkli çizgilerle temsil edilen çizgi çizimlere odaklanmışlardır. Kısmi kopyalar ve benzer kopyalar her zaman çizgi çizimlerin intihallerinde uygulandığından, şüpheli görüntülerden benzer kısmi kopyaları tespit etmek için nesne tespiti (object detection) ve görüntü alma tekniğini birleştirmeyi önermişlerdir. İlk olarak, bir kademeli sınıflandırıcı ile ROI'leri (İlgili Bölgeler - Region of Interest) tespit etmişler ve daha sonra bir özellik eşleştirme yöntemi kullanarak telif hakkıyla korunan görüntülerden karşılık gelen kaynak parçaları tespit etmişlerdir. Yapmış oldukları çalışmanın deneysel sonuçları, karmaşık arka planlardan benzer kısmi kopyaları tespit etmek için önerilen yöntemin etkinliğini kanıtlamıştır (Sun ve Kise, 2010).

Buna karşılık, Takayama, Johan ve Nishita (2012) yapmış oldukları araştırmada, Viola–Jones (2004) tespit çerçevesi de dahil olmak üzere insanların yüzleri için kullanılan yüz tespiti ve yüz tanımlama tekniklerinin renkli çizgi roman karakterlerine uygulanamayacağını göstermişlerdir (Takayama, Johan, ve Nishita, 2012; Viola ve Jones, 2004). Çünkü çizgi karakter yüzleri ve vücutları; insan yüzleri ve vücutlarındaki organların pozisyonları, boyutları, şekilleri hatta renk tonları açısından önemli ölçüde

farklılık göstermektedir. Takayama, Johan ve Nishita (2012) çalışmalarında ten rengi bölgelerini ve kenarlarını kullanan başka bir yüz algılama yöntemi önermişlerdir (Takayama, Johan, ve Nishita, 2012).

Khan ve diğerleri (2012) de özellikle çizgi karakter tespiti yapabilmek için nesne tespiti çalışmalarında, renk özniteliklerinden yararlanmışlardır (Khan, ve diğerleri, 2012).

Yu ve Seah (2011) çalışmalarında, çizgi film veri kümelerinin düşük boyutlu uzayını ortaya çıkaran FDM'i (Bulanık Yayılma Haritaları - Fuzzy Diffusion Maps) önermektedirler. Bulanık tutarlı ilişki FDM'de özelliklerin tercih sırasını ağırlık hesaplamasında kullanılan tercih derecesine dönüştürmek için kullanılmaktadır. ROC eğrilerini (Alıcı İşletim Karakteristik Eğrisi - Receiver Operating Characteristic Curve) kullanan deneysel sonuçlar, bulanık tutarlı ilişkinin ağırlık dağıtımında mükemmel performansa sahip olduğunu göstermektedir. Benzerliği değerlendirmek için FDM kullanılarak difüzyon mesafesi uygulanmaktadır. İzomapta kullanılan jeodezik mesafenin ve LPP'de (Yerelliği Koruyan Projeksiyonlar - Locality Preserving Projections) kullanılan yerel komşu mesafesinin aksine, yayılma mesafesi, iki veriyi birbirine bağlayan tüm uzunluktaki yolların kümesini içermektedir. Deneysel sonuçlar, diğer spektral yöntemlerle karşılaştırıldığında FDM'in karikatür tanıma ve karikatür kümelemede daha istikrarlı bir performansa sahip olduğunu göstermektedir. FDM'in karikatür veri seti için düşük boyutlu bir uzay oluşturabildiği ve benzerliği doğru bir şekilde değerlendirebildiği görülmektedir. FDM'de benimsenen yayılma mesafesi veriler arasındaki tüm yolları dikkate aldığından dolayı gürültü bozulmasına karşı toleranslıdır. Bu, yüz veri kümesinden farklı olarak çizgi film veri kümesinin dağılımının yerel ilişki koruma yöntemiyle temsil edilemeyeceği anlamına gelmektedir (Yu ve Seah, 2011).

Grafikler, belirli bir koleksiyondaki öğeler arasındaki ikili ilişkileri modellemek için kullanılan popüler veri yapılarıdır. Görüntü işlemede, komşuluk grafikleri genellikle bölümlere ayrılmış bölgelerin ilişkilerini belirlemek için kullanılmaktadır. Bu tür grafiklerin karşılaştırılması büyük ölçüde incelenmiştir ve grafik eşleştirme stratejileri, benzer kalıpları verimli bir şekilde bulmak için gerekli görülmüştür. Graf teorisini kullanan Ho, Rigaud, Burie ve Ogier (2013) çalışmalarında, çizgi romanlarda tekrar eden çizgi karakterleri tanımak için bir yöntem önermişlerdir. Kullandıkları yöntemde her

panel, atfedilen bir bitişiklik grafiği ile temsil edilmiş ve bu grafik kümesi arasında gereksiz yapıları bulmak için tam olmayan bir grafik eşleştirme stratejisi geliştirilmiştir. Bu sayede, aynı karakterin göründüğü farklı panellerde benzer alt grafiklerle temsil edilmesi mümkün olmuştur. Ana fikir, aynı karakter, göründüğü farklı panellerde benzer alt grafiklerle temsil edilmiştir. İki aşamalı eşleştirme işlemi ise bir düğüm eşleştirme adımından ve bir uç doğrulama adımından oluşmaktadır. Yaklaşımlarının özgünlüğü, karakterler için önceden herhangi bir nesne modeline gerek olmamasıdır. Algoritma, hikayenin ana karakterlerine karşılık gelen tüm tekrar eden yapıları otomatik olarak algılamaktadır. Herhangi bir modele gerek duyulmadan algoritmanın tüm gereksiz yapıları kendi başına algılaması, çalışmanın üstünlüğü olarak belirtilmiştir. Bu yaklaşım, bir çizgi romanda çizilen gereksiz çizgi karakterleri tespit etmek için kullanılmıştır, ancak herhangi bir fazla nesneyi tespit etmek için de kullanılabileceği bildirilmiştir (Ho H. N., Rigaud, Burie, ve Ogier, 2013). Ho, Rigaud, Burie ve Ogier (2014) bir diğer araştırmalarında, etkin bir şekilde her bir paneli, renk bölgelerinin düğümleri olarak kullanıldığı atfedilen bir bitişiklik grafiği olarak temsil ederek çizgi karakterleri tespit etmişlerdir. Çalışma, en sık görülen renk grubu görünümlerini otomatik olarak yerelleştirmek ve bunları ana karakterler olarak etiketlemek için fazla renk yapılarını bulmaktadır (Ho H. N., Rigaud, Burie, ve Ogier, 2014).

Sun, Bruie, Ogier ve Kise (2013) de çalışmalarında, en çok tekrarlanan öğeleri bulmak için fazlalık bilgi sınıflandırmasıyla SIFT (Ölçekle Değişmeyen Özellik Dönüşümü - Scale-Invariant Feature Transform) tanımlayıcısını kullanmaktadırlar. Bir tür Japon çizgi romanı olan Manganın telif hakkı koruması için tam ve benzer kısmi kopyaların tespitine yönelik olarak çalışma yapmışlardır. Bu amaçla, çizgi roman sayfalarındaki karakterlerin yerinin belirlenmesi ve karakterin tespiti olarak iki tür ilgi alanına dayalı olarak hem tam hem de benzer kopyaları belirlemek için içerik altyapılı görüntü alma yöntemlerinin uygulanmasını önermişlerdir. Yöntem, yalnızca karmaşık arka plana sahip görüntülerden kısmi kopyaları bulmakla kalmamış aynı zamanda tam kopya tespiti için telif hakkıyla korunan Manga sayfalarının ilgili kopyalanan bölümlerini ve benzer kopya tespiti için kopyalanan Manga karakterlerini rapor edebilmişlerdir. Çalışmada, olası dönüşümler ile ilgili kararlı öznitelikler üzerinde durulmuş ve bunları belirlemek için bir çerçeve önerilmiştir. Spesifik olarak, bazı ayırt edici özellikler, bir eğitim veri seti temelinde karakterleri sınıflandırmak için dedektörler olarak seçilmiştir. Dedektörlere bağlı olarak

aynı karakterlerin farklı sahnelerdeki çizimleri tespit edilebilmektedir. Metodoloji altı çizgi roman başlığı üzerinde denenmiş ve doğrulanmıştır. Farklı sahneler için müthiş değişikliklere rağmen, önerilen yöntem %70 çizgi roman karakterinin tespit edilmesini sağlamıştır. Bununla beraber deneysel sonuçlar, basılı kısmi kopyaların saptanması için önerilen yöntemin yüksek performansını kanıtlamıştır. Ayrıca, 10.000'den fazla Manga sayfası içeren bir veri tabanı kullanılarak, elle çizilmiş olanların %85'i ve benzer kısmi kopyaların %77'si nispeten yüksek hassasiyetle tespit edilmiştir (Sun, Burie, Ogier, ve Kise, 2013).

Iwata, Ito ve Kise (2014) çalışmalarında, Sun'un (2010) yöntemini amaca uygun olacak şekilde değiştirerek yeni bir Manga karakteri bulma yöntemi önermişlerdir. Çalışmalarındaki amaçları, manga görüntüleri için içerik tabanlı erişim yöntemi elde etmektir. Yapmış oldukları araştırmada yerel özellik çıkarımı ve yaklaşık en yakın komşular araması kullanılarak çizgi karakter tespiti için iyi sonuçlar elde ettiklerini göstermişlerdir (Iwata, Ito, ve Kise, 2014).

Bilgisayarlı görme alanında insan tespiti aktif bir araştırma alanıdır. Bunu çizgi roman hikayelerindeki ana karakterler gibi insan benzeri çizimlere genişletmek önemlidir. Çizgi roman analizi; grafikleri, metinleri, nesneleri ve insanları tanımanın kesiştiği çok yeni bir araştırma alanıdır. Çizgi karakterlerin otomatik algılanması çizgi roman anlayışı için doğru ve önemli bir adımdır. Riguad, Burie, Ogier ve Karatzas (2014) çalışmalarında, içerik tabanlı çizim alımı ve renk paleti kullanarak çizgi roman karakter alımı için renk tabanlı bir yaklaşım sunmaktadırlar. Basit bir kullanıcı etkileşimine dayalı yeni bir çizgi roman karakteri algılama çerçevesi önermişlerdir. Tek bir nesne rengi örneğinden, sunulan çerçevenin karakter temsilinin çoğunu, duruş ve ölçekten bağımsız olarak alabildiğini göstermişlerdir. Yazarlar önerdikleri yaklaşımda, sorgu renklerinin ayırt edilebilirliğine dayanan, gölge veya tarama aygıtı nedeniyle biraz farklı renge sahip benzer çizgi roman karakterleri tespiti için daha fazla uğraşılması gerektiğinden bahsetmişlerdir (Rigaud, Burie, Ogier, ve Karatzas, 2014).

Le, Luqman, Burie ve Ogier (2015) araştırmalarında, örnekle sorgulama (QBE, Query By Example) modelini kullanarak çizgi karakter görüntü taraması için sık alt grafik madenciliği (FSM, Frequent Subgraph Mining) tekniklerini kullanmışlardır. Atfedilen

bölge bitişik grafikleri (RAG, Region Adjacency Graphs) ve bunların sık görülen alt grafiklerini kullanarak karikatür sayfa görüntülerini belirlemek için bir şema önermişlerdir. İlk önce tüm çizgi roman cildinin her bir panelinin grafik yapılarını ve yerel özelliklerini çıkarmış, ardından farklı yerel özellik kategorilerini, atfedilen farklı RAG katmanlarına ayırmışlardır. Daha sonra, her bir katman için sık alt grafların listesi, FSM tekniği kullanılarak elde edilmiştir. İndeksleme ve içerik tabanlı görüntü alma sistemleri amacıyla, çizgi karakter tanıma ve sıralama, sorguyu temsil eden graflar ile keşfedilen sık alt graflar arasındaki izomorfizm kontrol edilerek yapılmıştır. Önerilen yaklaşımın deneysel sonuçları, QBE modelini kullanarak çizgi görüntülerin güvenilir erişim sonuçlarını elde edebileceğini göstermektedir (Le, Luqman, Burie, ve Ogier, 2015; Le, Luqman, Burie, ve Ogier, 2015).

Matsui ve diğerleri (2017) çalışmalarında Mangaya özgü bir görüntü-tanımlama çerçevesi önermişlerdir. Önerilen sistem, etkin kenar boşluğu etiketleme, ekran tonu kaldırmalı kenar yönlendirme histogramı, özellik açıklaması ve ürün niceleme kullanarak yaklaşık en yakın komşu aramasından oluşmaktadır. Sorgulama için sistem, çizim tabanlı bir ara yüz kullanmaktadır. Ara yüze dayalı olarak, iki etkileşimli yeniden sıralama şeması sunulmuştur: alaka düzeyi geri bildirimi ve sorgu rötuşu. Değerlendirme için, profesyonel Manga sanatçıları tarafından çizilen 21142 sayfalık 109 çizgi romandan oluşan Manga109 adlı yeni bir manga görüntüleri veri seti oluşturulmuştur. Deneysel sonuçlar, önerilen çerçevenin verimli ve ölçeklenebilir olduğunu göstermiştir. Verimli marj etiketleme, kenar oryantasyon histogramı, özellik tanımı ve ürün niceleme kullanarak yaklaşık en yakın komşu aramasından oluşan çalışmalarında, eskiz tabanlı sorgu modelini kullanarak çizgi karakterleri de tespit etmektedirler (Matsui, ve diğerleri, 2017).

Çizgi karakterlerin yüz tespiti, çizgi karakter alma, otomatik karakter sınıflandırma ve çizgi roman analizi gibi çoğu uygulamada gerekli bir adımdır. Bununla birlikte, mevcut yöntemler basit çizgi film görüntüleri veya küçük boyutlu çizgi karakter veri kümeleri için geliştirilmiştir ve algılama performansının iyileştirilmesi gerekmektedir. Qin ve diğerleri (2017) çalışmalarında, çizgi karakterlerin yüz tespiti için hızlı bölgesel tabanlı evrişimli sinir ağı (R-CNN - Region Based Convolutional Neural Network) yöntemi önermişlerdir. Alana katkılarının iki yönlü olduğundan bahsetmektedirler. İlk olarak, yüz

algılamanın ikili sınıflandırma (binary classification) görevi için, ampirik olarak sigmoid sınıflandırıcının softmax sınıflandırıcıdan daha iyi sonuçlar gösterdiğini bildirmişlerdir. İkinci olarak, karakter yüz algılama değerlendirmesi için toplamda 3375 çizgi roman sayfasından oluşan JC2463 ve AEC912 olmak üzere iki çizgi roman veri seti oluşturmuşlardır. Deneysel sonuçlar, önerilen yöntemin sadece mevcut yöntemlerden daha iyi olduğunu değil, aynı zamanda farklı çizim stillerine sahip çizgi karakter görüntüleri için de çalıştığını göstermiştir (Qin, Zhou, He, Wang, ve Tang, 2017).

Çizgi roman karakterlerinin tespiti, sadece çizgi romanlar için daha etkili bir indeksleme ve erişim sağlamakla kalmayıp, çizgi romanların dijital formatlarını daha iyi oluşturmak için yeterli bir anlayış sunarak çizgi roman analizi alanında ilgi çekici bir konu haline gelmiştir. Son yıllarda, çizgi romanlardan karakterleri çıkarmak/tespit etmek için önerilen birkaç yöntem makul bir performans göstermiştir. Ancak, diğer çalışmalarla karşılaştırmadan veya standart bir veri kümesi üzerinde deneyler yapmadan yöntemleri değerlendirmek için hala erkendir.

Nguyen, Rigaud, ve Burie (2017) yapmış oldukları çalışmada, derin öğrenmenin son ve önemli gelişmelerinden yararlanarak derin öğrenmeyi çizgi roman karakteri algılamaya uygulamışlardır. Önerilen veri kümelerine dayalı olarak çizgi karakter dedektörünü eğitmek için derin öğrenme kullanmışlardır. YOLOv2 mimarisiyle PASCAL VOC kullanarak Sequencity612, Fahad18, Ho42 ve Sun60 veri setleriyle ayrı ayrı deneysel çalışmalar yapmışlardır. Önceden yapılan çalışmaların sonuçlarına göre kendi yöntemlerinin daha iyi performans gösterdiğini kanıtlamışlardır. (Nguyen, Rigaud, ve Burie, 2017).

Manga görüntülerinde nesne tespiti için ESA kullanılmaya başlanmıştır. Yanagisawa, Yamashita ve Watanabe (2018) tarafından ESA kullanarak Manga görüntülerinden nesne tespiti yöntemi üzerine yapılan bir çalışmada ise, hızlı ve daha hızlı R-CNN'ler karşılaştırılarak Manga nesne tespitinin etkinliği incelenmiştir. En son algılama yöntemi olan tek atışlı multiBox dedektör (SSD - Single Shot MultiBox Detektör), bir görüntüdeki küçük bölgeler için nesne sınıflandırması ve kutu ayarlaması yapmışlardır. Burada Manga nesneleri; panel düzeni, konuşma balonu, karakter yüzü ve metin olarak ifade edilmiştir. Deneysel sonuçlar, hızlı R-CNN'nin panel düzeni ve konuşma balonu için, daha hızlı R-

CNN ise karakter yüzü ve metin için etkili olduğunu göstermiştir (Yanagisawa, Yamashita, ve Watanabe, 2018).

Nguyen, Rigaud ve Burie (2018) diğer bir çalışmalarında önermiş oldukları veri kümesi ve ayrıca önceki çalışmalardan elde edilen mevcut veri kümeleri üzerinde deneyler yaparak, çizgi karakter tespit etmede kullandıkları yöntemin mevcut yöntemlerden önemli ölçüde daha iyi çalıştığını kanıtlamışlardır. Yine aynı çalışmada, görüntüleri XML benzeri bir metin dosyasına kodlamak ve indekslemek için en gelişmiş görüntü analiz yöntemlerinin nasıl birleştirileceğini göstermişlerdir. İçerik açıklama dosyası daha sonra çizgi roman görüntülerini ilgili içerikleriyle ve bilgileriyle kolayca indekslenebilen panellere karşılık gelen alt görüntülere otomatik olarak bölmek için kullanılabilmektedir. Bu, belirli çizgi roman karakterleri tarafından söylenen anahtar kelimelerde gelişmiş arama, doğal dil işlemeyi kullanarak aksiyon ve sahne alımına izin vermektedir. Panel, balon, metin, çizgi karakter tespiti ve çizgi karakter yüz tespiti gibi çalışmalarda geleneksel yaklaşımlar ve derin öğrenme modelleri bir arada kullanılarak algılama ve ayrıca LSTM (Uzun Kısa Süreli Bellek - Long Short Term Memory) modeli kullanılarak metin tanıma yapılmıştır. Araştırmada, çevrimiçi kütüphane içeriğinden oluşan bir veri seti üzerinden değerlendirmeler sunulmuş ve ayrıca yeni bir genel veri seti de önerilmiştir (Nguyen, Rigaud , ve Burie, 2018).

Nguyen, Rigaud, ve Burie (2019) bir diğer çalışmalarında, çizgi roman görüntü analizi için işlem süresini, araştırmalarında önerdikleri yöntemlerle azaltmayı amaçlamışlardır. Görev başına bir model kullanmak yerine Comic MTL (Çoklu Görev Öğrenimi - Multi-Task Learning) adı verilen birden çok görevi öğrenebilen bir model önermişlerdir. Tespit ve segmentasyon görevlerine ek olarak, balonlar ve çizgi karakterler için ilişki analizi görevini Comic MTL modeline entegre etmişlerdir. Yapılan deneyler, paneller, balonlar ve çizgi karakterlerin tespiti için ek açıklamaları ve ayrıca balon ile karakter arasındaki ilişkileri içeren DCM772 ve eBDtheque genel veri kümeleri üzerinde gerçekleştirilmiştir. Comic MTL modelinin balonlar ve konuşmacıları (çizgi karakterler) arasındaki ilişkileri algılayabildiğini bunun yanında panel ve çizgi karakter tespiti gibi diğer görevlerin de iyi bir performansla uygulandığını göstermişlerdir. Ayrıca umut verici sonuçlarla balon segmentasyonunu da gerçekleştirebildiğini başarıyla kanıtlamışlardır (Nguyen, Rigaud, ve Burie, 2019).

Dijitalleştirilmiş Manga (Japon çizgi roman) kitaplarına, verimli ve etkili erişimin acilen gerekli olduğunu düşünen Chu ve Li (2019) çalışmalarında Manga karakterleri yüz tespiti için bir ESA yöntemi önermişlerdir. Konu olarak zorlu ama nispeten çalışma zamanı keşfedilmemiş bir alandır. Bir Manga sayfası verildiğinde, önce seçici arama şemasına dayalı olarak aday bölgeleri tespit etmişlerdir. Daha sonra çeşitli görünümlerdeki Manga yüzlerini tespit etmek için üç ESA önermişlerdir. Tüm nesne bölgesinden ve birkaç yerel bölgeden bilgi çıkarılmış ve çok ölçekli bilgileri erken birleştirme veya geç birleştirme şeklinde birbirlerine entegre etmişlerdir. Chu ve Li (2019) tarafından çalışmalarında önerilen yöntemler, insan yüzleri için tasarlanmış son teknoloji yüz algılama modüllerine kıyasla ikna edici bir performans göstermiştir (Chu ve Li, 2019).

Dutta ve Bisvas (2019) ise çalışmalarında, Bengalce çizgi roman sayfası görüntülerinden panellerin/çizgi karakterlerin ESA tabanlı çıkarımı üzerine odaklanmışlardır. Çizgi roman görüntülerinden hem panellerin hem de çizgi karakterlerin çıkarılması için YOLO (You Only Look Once) ve ESA'nın özelliklerine dayanan bir panel/çizgi karakter yerelleştirme mimarisi önerilmişlerdir. Dutta ve Bisvas 4130 görüntüden oluşan Bengali Çizgi Roman Görüntüsü veri seti (BCBId) ile alana katkı sağlamışlardır. Önerdikleri yöntemle, BCBId, eBDtheque, Manga 109, DCM772 veri setlerinde ve halka açık çeşitli karikatür veri setlerinde dikkate değer sonuçlar elde ettiklerini rapor etmişlerdir. (Dutta ve Biswas, 2019).

Son zamanlardaki derin öğrenme uygulamaları trendinde uygun bir veri setinin olmaması nedeniyle Aizawa ve diğerleri (2020) tarafından çalışılıp, Manga109 Japon çizgi romanı (94 yazar ve 21142 sayfa) veri setini referans alan yeni bir Manga109 veri seti geliştirilmiştir. Oluşturdukları veri setini, akademik kullanım için yazar izinleri alınarak kamuya açık hale getirmişlerdir. Bu veri setinde çerçeveleri, konuşma metinlerini, karakter yüzlerini ve karakterin bedenlerini dikkatlice açıklamışlardır. Önerilen veri seti, makine öğrenmesi algoritmalarında ve bunların değerlendirilmesinde faydalı olacak çok sayıda Manga görüntüsü ve ek açıklama sağlamaktadır. Akademik kullanıma ek olarak, endüstriyel kullanım için de daha fazla izin alınmıştır. Yazarlar çalışmalarında, mevcut derin öğrenme yöntemlerini uygulayan ve veri kümesi tarafından mümkün kılınan birkaç multimedya işleme uygulaması (metin algılama, eskiz tabanlı erişim ve karakter yüzü tespiti) örneği sunmuşlardır (Aizawa, ve diğerleri, 2020).

Çizgi roman karakter tespiti makine öğrenimi alanında heyecan verici bir çalışma alanı olarak geçmektedir ve bu konuda son zamanlarda yeterli performansı sağlamak için birçok yöntem önerilmiştir. Aynı zamanda, bu yöntemlerin çoğu, modellerinin performanslarını bazı standart veri kümeleriyle karşılaştırmadan değerlendirmek için birkaç yüz görüntü ve daha az sınıf içeren özel veri kümelerini kullanmaktadır. Mashood Nasir ve diğerleri (2020) araştırmalarında, bir yarışmadan alınan standart kamuya açık bir veri kümesini kullanmışlardır ve ESA'nın derinlemesine eğitimini geliştirmek ve etkinleştirmek için dengesiz veri kümesi için genel bir veri dengeleme tekniği önermişlerdir. Ek olarak, süper kahramanları verimli bir şekilde sınıflandırmak için özel bir 17 katmanlı ESA da önermişlerdir. Hesaplanan sonuçlar, %97 genel sınıflandırma doğruluğuna ulaşmıştır (Mashood Nasir, ve diğerleri, 2020).

Kim ve diğerleri (2020) animasyon filmlerindeki karakterlerin tespiti için bir çalışma yapmışlardır. Animasyon filmlerdeki karakterlerin tespitindeki iki büyük zorluktan bahsetmişlerdir: aynı karakter; konusu çeşitli, benzersiz stillerle ifade edilebilir ve karakter nesnelerinin doğası ve tasarımı üzerinde stilistik veya diğer kısıtlamalar olmadığını ileri sürmüşlerdir. Çalışmalarında bu zorlukların üstesinden gelmek için, animasyon filmlerindeki karakterleri tespit etmek ve tespit edilen karakterlerin insan mı yoksa insan olmayan tipler mi olduğunu belirlemek için animasyon uyarlamalı bölge tabanlı ESA modelini sunmuşlardır. İki aşamalı bir nesne algılayıcı olan daha hızlı R-CNN modeline tek bir model kullanarak animasyon filmlerinden benzersiz stil çeşitliliğini öğrenmek için hiyerarşik bir animasyon adaptasyon modülü eklemişler ve karakter sınıfını belirlemede görsel olarak önemli olan bölgelere odaklanmak için çift dedektör mimarisini kullanmışlardır. Önermiş oldukları model çizgi karakter sözlüğü oluşturma görevi için kullanılan yöntemlere kıyasla etkili bir performans sunmuştur. Modelleri tüm karakter türlerinin ortak görsel temsillerini bulabilmekte ve animasyonlu karakterleri etkili bir şekilde tespit etmektedir (Kim, Lee, Seo , Im, ve Lee, 2020).

Derin öğrenmeye dayalı yüzle ilgili modellerdeki kayda değer başarıya rağmen, bu modeller hala gerçek insan yüzleriyle sınırlıdır. Çünkü iyi organize edilmiş bir veri setinin olmamasından dolayı animasyon yüzleri alanı daha az çalışılmıştır. Kim ve diğerleri (2021), animasyon yüzü alanındaki araştırmaları artırmak için kontrol edilebilir sentetik animasyon modelleri aracılığıyla büyük ölçekli bir ünlüler animasyonu veri setini

(AnimeCeleb) sunmuşlardır. Veri üretme sürecini kolaylaştırmak için, açık bir 3D yazılıma ve geliştirilmiş bir açıklama sistemine dayalı yarı otomatik bir boru hattı inşa etmişlerdir. Bu, zengin ek açıklamalara sahip, çok pozlu ve çok stilli animasyon yüzlerini içeren büyük ölçekli bir animasyon yüz veri setini oluşturmaya yol açmıştır. Yapmış oldukları deneylerde veri setlerinin animasyonla ilgili çeşitli görevlere uygulanabilir olduğunu kanıtlamışlardır (Kim, Park, Lee, Chung, ve Lee, 2021).

Kotlicki (2021) tarafından, çizgi roman karakterleri alanında HaarCNN adı verilen derin öğrenme ve kademeli sınıflandırıcılar ile çizgi roman karakterlerinin tespiti ve tanınması üzerine bir araştırma yürütülmüştür. Araştırmacı çalışmasının mevcut yöntemlerden daha iyi potansiyele sahip olduğunu doğrulamak için standart yöntemlerle karşılaştırmalar yapmıştır. Bu amaçla, çizgi romanlardan çıkarılan 750 adet görüntü değerlendirilip %85'in üzerinde kesinlik ve tespit edilen yüzlerin yaklaşık %80'inin geri çağrılması ve %80'in üzerinde doğru ana karakter tanıma elde edilmiştir. Görüntüler yaklaşık 200 saniyede işlenmiştir. Araştırmaya göre karakter açıklamalarının potansiyel olarak tatmin edici sonuçlarının, üretken çekişmeli ağlar gibi derin öğrenme alt alanlarında avantajlı olabileceği öngörülmüştür (Kotlicki, 2023).

Nir, Rapoport ve Shamir (2022) yaptıkları bir çalışmada mevcut bilgisayarla görme ve derin öğrenme çözümlerinin, doğal görüntüler üzerinde eğitildikleri için animasyonlu içeriklerde genellikle başarısız olduklarından bahsetmişlerdir. Bu nedenle araştırmalarında, belirli animasyon içeriklerine uygun, anlamsal temsili iyileştirmek için bir yöntem sunmuşlardır. Kendi kendine denetim kullanarak animasyonlu içerik için uygun bir stile özgü semantik temsili öğrenmek üzere bir yöntem sunmuşlardır. Öz denetim, çoklu nesne takibine ve belirli bir stile dayanan bir eşleme geliştirmek için üçlülerden oluşan bir veri kümesi oluşturmaya dayanmaktadır. Anlamsal temsil alanında kümeleme kullanmak, animasyon filmleri için otomatik olarak karakter sözlükleri oluşturmaya, karakterlerin çok sınıflı sınıflandırıcıları için eğitim kümeleri toplanmaya ve animasyon videolarındaki karakterlerin yoğun etiketlemesini otomatik olarak gerçekleştirmeye olanak tanımıştır. Böylelikle, kullanıcıların ve medya şirketlerinin CAST (İzleme ile Özdenetim Kullanarak Animasyonda Karakter Etiketleme - Character labeling in Animation using Self-supervision by Tracking) kullanarak hem video hem de dizi düzeyinde bilgileri otomatik olarak indeksleyip ortaya çıkarabilen bir hizmetten

faydalanmasına, animasyon içeriklerini çok daha kolay analiz etmesine, yeniden kullanmasına, aramasına ve para kazanmasına olanak tanıyacak bir çalışma yapmışlardır. Çeşitli animasyon stillerinde karakterlerden oluşan bir koleksiyonun sonuçlarını sunmuşturlar (Nir, Rapoport, ve Shamir, 2022).

Bu çalışmadaki literatür taraması, çeşitli görsel medya türlerinde karakter algılama ve nesne tespiti üzerine yapılan önemli araştırmalara genel bir bakış sunmaktadır. Her bir çalışma, farklı metodolojiler ve uygulamalar kullanarak bu alandaki mevcut zorluklara çözümler getirmeye çalışmıştır. Çizgi karakter tespiti konusunda çeşitli yaklaşımların ve tekniklerin kullanıldığı görülmektedir. Her bir yöntem, çizgi karakterlerin özelliklerini ve görünümlerini dikkate alarak tasarlanmıştır. Bu tezde çizgi romanlardaki karakterlerden ziyade daha fazla çeşitlilik sağlayan çizgi filmlerdeki karakterleri tanımaya odaklanılmıştır.

Tablo 1. Literatür Taramasının Özeti

Kaynak	**Kullanılan Teknik**	**Tespit Türü**	**Veri Seti**	**Doğrulama Metrikleri**
Sun ve Kise (2010)	ROI Tespit	Yüz Tespiti	Veri Seti Oluşturulmuş	Precision, Recall
Yu ve Seah (2011)	FDM	Karakter Kenar Tespiti	Cartoon Veri Seti (Tom and Jerry)	ROC, Accuracy
Takayama, Johan ve Nishita (2012)	Nesne Tespiti	Yüz Tespiti	Lucky Paraside, TV TOKYO, MBS	Precision, Recall, F-measure
Ho, Rigaud, Burie ve Ogier (2013)	Adjacency Graphs	Karakter Tespiti	Cosmozone	Precision
Ho, Rigaud, Burie ve Ogier (2014)	Adjacency Graphs	Karakter Tespiti	Cosmozone	Precision

Sun, Bruie, Ogier ve Kise (2013)	SIFT	Karakter Tespiti	eBDtheque Veri Seti	Precision, Recall
Iwata, Ito ve Kise (2014)	HOG	Karakter Tespiti	Manga Veri Seti	Precision, Recall
Riguad, Burie, Ogier ve Karatzas (2014)	Renk Tabanlı Nesne Tespiti	Karakter Tespiti	eBDtheque Veri Seti	Precision, Recall
Le, Luqman, Burie ve Ogier (2015)	RAGs, FSM	Karakter Tespiti	Cosmozone	Precision, Recall
Matsui ve diğerleri (2017)	ANN	Görüntü Tanımlama	Manga109	Recall
Qin ve diğerleri (2017)	R-CNN	Yüz Tespiti	JC2463, AEC912	Precision, Recall, F-measure
Nguyen, Rigaud, ve Burie (2017)	YOLOv2	Karakter Tespiti	Sequencity612, Fahad18, Ho42, Sun60	Precision, Recall
Yanagisawa, Yamashita ve Watanabe (2018)	Fast R-CNN, Faster R-CNN	Panel Düzeni, Konuşma Balonu, Karakter Yüzü ve Metin Tespiti	Manga109	mAP, IoU
Nguyen, Rigaud ve Burie (2018)	SSD, Faster R-CNN, YOLOv2	Paneller, Balonlar, Metin Karakter Yüzü ve Karakter Tespiti	DCM772, eBDtheque, Fahad18	Precision, mAP

Nguyen, Rigaud, ve Burie (2019)	CNN, Mask R-CNN ComicMTL	Paneller, Balonlar ve Karakterlerin Tespiti	DCM772, eBDtheque	Precision, Recall, F1
Chu ve Li (2019)	CNN	Yüz Tespiti	Manga FaceNet	Precision, Recall, F-measure
Dutta ve Bisvas (2019)	YOLO, CNN	Panellerin ve Karakterlerin Tespiti	BCBId	Precision, Recall, F-measure, IoU
Aizawa ve diğerleri (2020)	GAN, EOH, SSD	Paneller, Metinler, Karakter Yüzleri ve Karakter Bedenleri Tespiti	Manga109, COMICS	Precision, IoU
Mashood Nasir ve diğerleri (2020)	CNN	Karakter Tespiti	Publicly Veri Seti	Accuracy, Loss
Kim ve diğerleri (2020)	AA-R-CNN	Karakter Tespiti	Images ©Disney Enterprises	Precision, mAP
Kim ve diğerleri (2021)	Pose Mapping Method	Yüz Tespiti	AnimeCeleb	Self-identity, Cross-domain
Kotlicki (2021)	HaarCNN	Karakter Tespiti	Dilbert Comics	Precision, Recall, F1, Inference Time
Nir, Rapoport ve Shamir (2022)	CNN, CAST	Karakter Tespiti	Manga	Precision, Recall, F1, Accuracy, mAP

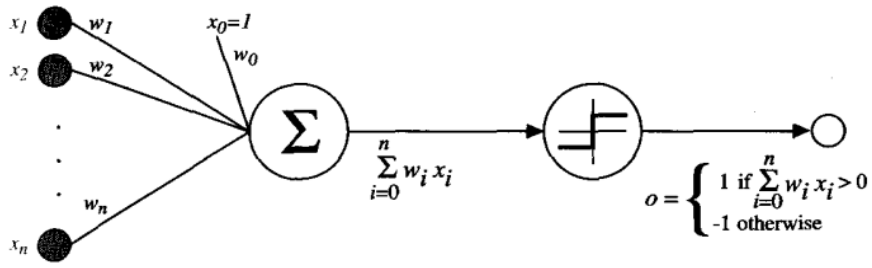
3. MATERYAL VE YÖNTEM

Bu bölümde, tezin içeriğinde kullanılan tanımlar ve modeller gibi konular ele alınmıştır. Ayrıca, uygulamada kullanılan yöntemlere ve çalışma planına yer verilmiştir. Çalışmada kullanılan KralSakir veri setinin hazırlanması aşamaları ve önerilen modeller hakkında ayrıntılı bir açıklama yapılmıştır.

3.1 Yapay Sinir Ağları

YZ'nin temelleri Turing makineleriyle atılmış ve 1970 yılından sonra bu konuda çalışmalar hız kazanmıştır. YZ, genellikle insanlar tarafından gerçekleştirilen zihinsel faaliyetleri otonom hale getirme amacını taşımaktadır. Çok kapsamlı araştırma konusu olan YZ; makine öğrenmesi ve derin öğrenmeyi içermekte olup aynı zamanda öğrenmeyi kapsamayan birçok yaklaşımı da barındırmaktadır (Chollet, 2021). YZ'da Shannon ve Newell (1996) tarafından geliştirilen her bir kuralı insanlar tarafından tek tek kodlanan satranç uygulaması bir makine öğrenmesi olarak kabul edilmemiştir (Samuel, Shannon, ve Simon, 1996). YZ'nın satranç gibi iyi tanımlanmış problemlerde iyi çözümler sunduğu kabul edilse de görüntü sınıflandırma (image classification), ses tanıma ve diller arasında çeviri yapma gibi daha birçok karmaşık problemi çözmek için makine öğrenmesi yaklaşımı ortaya çıkmıştır (Chollet, 2021).

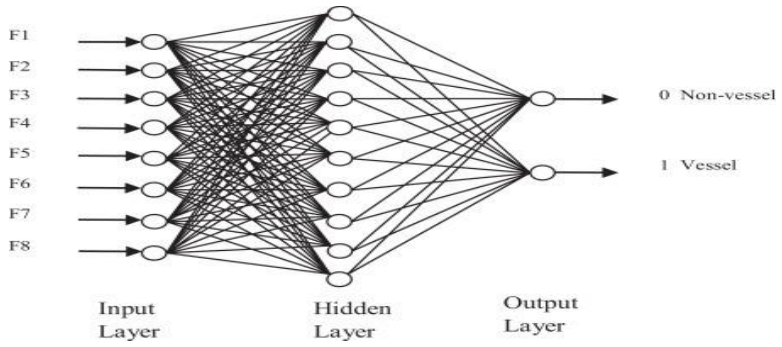
Makine öğrenmesi ve derin öğrenme temelleri, biyolojik nöronların basitleştirilmiş matematiksel modelini kuran McCulloch ve Pitts (1943) ile başlayan YSA araştırmalarına dayanmaktadır (McCulloch ve Pitts, 1943). Rosenblatt (1958) matematiksel modele öğrenme yeteneğini ekleyerek algılayıcı (perceptron) modelini ortaya atmıştır (Rosenblatt, 1958). Algılayıcının ortaya çıkışı, sinir ağı gelişiminin sıçramasına sebep olmuştur. Algılayıcı Şekil 1'de gösterilmiştir.



Şekil 1. Algılayıcı (Mitchell, 1997)

YZ'nin en çok araştırma yapılan alanı YSA'dır. Bilgisayar sistemleri olan YSA, insan beyninin temel özelliği olan öğrenme fonksiyonunu gerçekleştirmektedir. Bu ağlar, öğrenme sürecini örnekler yardımıyla yapmaktadırlar. YSA, birbirine bağlı işlem yapan yapay sinir hücreleri elemanlarından oluşmaktadır. Her bağlantıda ağırlık değerleri bulunmaktadır. YSA'nın sahip olduğu tüm bilgi, bağlantılardaki ağırlık değerlerinde saklı olmakla beraber ağa yayılmaktadır (Öztemel, 2003).

Tek katmanlı algılayıcılar YSA'nın ilk çalışmalarıdır. Tek katmanlı YSA'nın en önemli özelliği; model yalnızca doğrusal sınıflandırma problemlerini çözebilir, basit XOR ve XNOR problemlerini çözemez (Minsky ve Papert, 1969) Doğrusal olmayan ilişkileri çözebilmek içinse çok katmanlı algılayıcılar (ÇKA) geliştirilmiştir. Rumelhart, Hinton ve Williams (1986) tek katmanlı algılayıcının çözemediği bazı problemleri ele alan hata geri yayılım algoritması/geri yayılım ağı (BP, Back Propagation) ile eğitilen çok katmanlı ileri beslemeli bir ağ önermişlerdir. Doğrusal olmayan sınıflandırma ve öğrenme sorunlarını etkili bir şekilde çözebilen bir sinir ağı geliştirilmiştir (Rumelhart, Hinton, ve Williams, 1986).



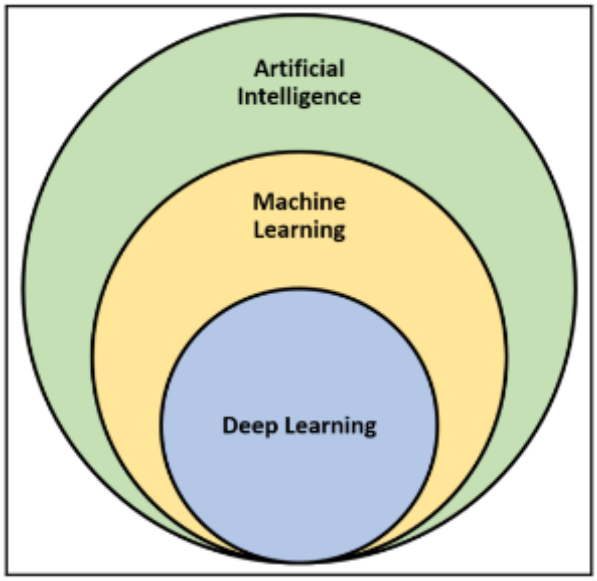
Şekil 2. Yapay Sinir Ağı (Barkana, Sarıçiçek, ve Yıldırım, 2017)

Şekil 2'de bir YSA yapısı gösterilmiştir. Günümüzde en yaygın kullanılan YSA modeli, ÇKA ağlarıdır. ÇKA'lar, %95 oranında mühendislik problemlerini çözebilecek niteliktedirler. Yapılan çalışmalar sonucunda XOR problemlerini çözebilmek için ÇKA ağları ileri sürülmüştür. ÇKA'lar üç katmandan oluşmaktadır; girdi katmanı, gizli katman ve çıktı katmanı. Girdi katmanı herhangi bir hesaplama içermediğinden, bu ağ ile çıktıların üretilmesi hem ara katmanları hem de çıktı katmanları için hesaplamaların gerçekleştirilmesini gerektirir; dolayısıyla, Şekil 2'deki bu ÇKA'da katman sayısı ikidir. Bu katmanların her ikisi de tam bağlı olduğundan dolayı her girdi, ara katmandaki her nöronu etkilemektedir ve bunların her biri de çıktı katmanındaki her nöronu etkilemektedir (Zhang, Lipton, Li, ve Smola, 2023). Girdi katmanından gelen bilgileri ara katmanlar belirli işlemlerden geçirmektedir. Tek bir ara katman, birçok problemle başa çıkabilir; ancak istenilen problemin girdi/çıktı ilişkisi karmaşıklaştıkça ve doğrusal olmadıkça, birden fazla ara katman kullanmak gerekebilir. Çıktı katmanı; ara katmandan gelen bilgileri belirli işlemlerden geçirerek ağın girdi katmanına sunulan veri için üretilen çıktıyı belirlemektedir. Nihai olarak çıktı katmanından gelen veri dış dünyaya iletilmektedir (Öztemel, 2003).

Waibel ve diğerleri (1989) konuşma tanıma için bir boyutlu ESA olarak görülebilecek bir zaman gecikmeli sinir ağı önermişlerdir (Waibel, Hanazawa, Hinton, Shikano, ve Lang, 1989). Ardından, Zhang ve diğerleri (1988) ilk iki boyutlu ESA kayma değişkenli YSA'yı önermişlerdir (Zhang, Tanida, Itoh, ve Ichioka, 1988). LeCun ve diğerleri (1989) BP algoritmasına dayanan el yazısı posta kodu tanıma için bir ESA oluşturmuş ve LeNet'in orijinal versiyonu olan konvolüsyon terimini ilk kez kullanmışlardır. LeCun ve diğerleri (1989) ESA modelini önererek modern bilgisayarla görme için derin öğrenmenin temelini oluşturmuşlardır (LeCun, ve diğerleri, 1989).

YZ, insan zekasını taklit ederek; verilen görevleri yerine getirebilmek için topladığı bilgilerle kendisinin en iyi versiyonu olan sistemleri veya makineleri ifade etmektedir. Makine öğrenmesi bilgisayarların algısal verileri veya veri tabanları gibi işlenmiş verilerden öğrenmelerini sağlayan algoritmaların tasarımı ve geliştirilmesini konu edinen bir bilim dalı olmakla beraber YZ'nin de bir alt dalıdır. Derin öğrenme, bir makine öğrenmesi yöntemidir. Bu yöntem, bir veri kümesi ile çıktıları tahmin etmek için YZ modelini eğitmek için kullanılmaktadır. Derin öğrenme, çok katmanlı ağlar gibi karmaşık

yapılar kullanarak büyük miktarda veriyi analiz edip öğrenme yeteneği ile bilinmektedir. Genellikle büyük ve karmaşık veri setlerinde daha etkili olan derin öğrenme aynı zamanda makine öğrenmesinin bir alt dalıdır (Özkan ve Kar, 2022). YZ, makine öğrenmesi ve derin öğrenmenin birbirleri ile ilişkileri Şekil 3'te verilmiştir.



Şekil 3.3. Yapay Zeka, Makine Öğrenmesi ve Derin Öğrenmenin İlişkisi (Sarkar, Bali, ve Ghosh, 2018)

Makine öğrenmesi ve derin öğrenmede, makinelere geçmiş verileri kullanarak tahmin yapma ve öğrenme yeteneği kazandırılmaktadır. Makine öğrenmesinde, genellikle istatistiksel metotlar yaygın kullanılırken derin öğrenmede ise ÇKA ile hesaplama yapılmaktadır. Makine öğrenmesinde, genellikle öznitelik çıkarımı manuel yapılırken derin öğrenmede ise öznitelikler otomatik olarak öğrenilmektedir.

3.2 Makine Öğrenmesi

Makine öğrenmesi, bilgisayar sistemlerinin belirli görevleri veri kullanarak belirli bir performans seviyesine getirebilmeleri için öğrenme yeteneği kazanmalarını sağlayan bir YZ alt dalıdır. Makine öğrenmesi sistemleri programları kurallarına göre tek tek kodlamak yerine eğitilmektedirler. Makine öğrenmesi algoritmaları, genellikle büyük miktarda veri setini inceleyerek veya öğrenerek belirli bir görevde doğru ve özel bir cevap üretmek üzere eğitilmektedirler (Özkesici ve Yılmaz, 2021). Makine öğrenmesi, verilen girdilere ve çıktılara bakarak istatistiksel örüntüler ortaya çıkarıp kuralları

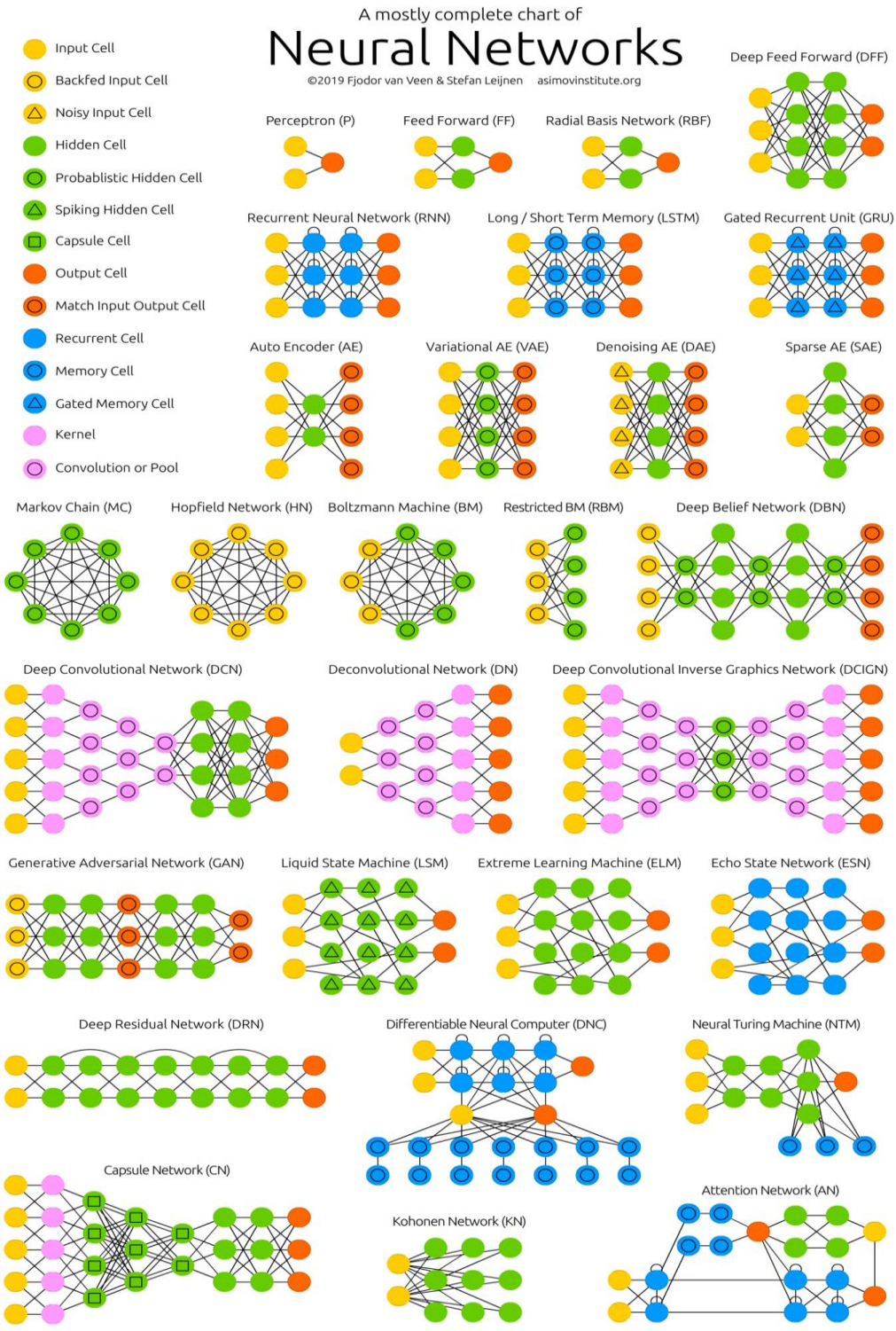
otomatikleştirerek kendi kendine öğrenen bir sistem geliştirilebilmektedir. Sistem, veri setlerindeki desenleri tanıma ve öğrenme yoluyla, belirli görevlerde (örneğin; sınıflandırma, regresyon, kümeleme) daha iyi performans sergilemektedir. Bu sistemler, veri analizi yaparak desenleri tanıyabilmekte, kararlar alabilmekte ve gelecekteki verilere dayanarak tahminlerde bulunabilmektedirler. Makine öğrenmesi, açıkça programlanmadan veri ve gözlemler aracılığıyla bilgisayarlara bilgi sağlamayı amaçlayan YZ araştırmalarının bir bölümünü ifade etmektedir. Bunu girdi (yani, algoritmanın beslendiği metin, görüntü veya video verileri) ve çıktı (yani, sınıflandırma) arasındaki uyumun iyiliğini optimize etmek için algoritma içindeki parametrelerin ayarlanmasıyla gerçekleştirmektedir. Edinilen bilgi, bilgisayarların yeni ortamlara doğru şekilde genelleme yapmasını sağlamaktadır (Tang, ve diğerleri, 2018). Makine öğrenmesi, genellikle deneme-yanılma yöntemiyle, istatistiksel modellerin oluşturulması, veri madenciliği ve algoritmaların geliştirilmesi gibi süreçleri içermektedir. Makine öğrenmesi; web araması, doğal dil işleme, görüntü tanıma, tıbbi teşhis, robotik, çeviri, sürücüsüz araçlar, sosyal medya analizi ve oyun stratejileri geliştirme gibi birçok alanda kullanılmaktadır.

3.3 Derin Öğrenme

Derin öğrenme, insan beyninde bulunan sinir ağlarının nöronunu taklit eden bir makine öğrenmesi türüdür. Geleneksel makine öğrenmesi yöntemlerinden ayrışan derin öğrenme; kodlanmış kurallar yerine büyük miktarda veri üzerinden öğrenme prensibine dayanmaktadır. Metin, ses, video ve resim gibi farklı veri çeşitlerini işleyerek karmaşık desenleri ve ilişkileri anlama yeteneği ile bilinmektedir. Derin öğrenme, çoğunlukla fazla miktardaki veri setleri üzerinde eğitilmekte ve öğrenme sürecinde örnekler üzerinden karmaşık hiyerarşik özellikleri öğrenmektedir (Yılmaz ve Kaya, 2021). Buda doğruluk oranlarını artırabilmekte ve daha karmaşık görevlerde yüksek performans elde etmeyi mümkün kılmaktadır. Derin öğrenme modelleri, bir görevi çözmek için belirli bir eğitim verisi görüntüsü üzerinde eğitilmektedir. Derin öğrenme modelleri, temel olarak bir insan gibi görmeyi ve görselleştirmeyi sağlayan bilgisayarların görme alanında kullanılmasını sağlamaktadır (Cengiz, 2020).

Çok katmanlı ağlar, yapılan araştırmalarda doğal verilerin bileşimsel yapısından faydalandıkları için başarılı bulunmaktadır. Bileşimsel hiyerarşide, hiyerarşinin bir

katmanındaki objelerin kombinasyonu, bir sonraki katmandaki nesneleri oluşturmaktadır. Birden çok katman kullanarak ve ağın doğru özelliklerinin kombinasyonunu öğrenmesini sağlayarak bu hiyerarşi taklit edilebilmekte ve derin öğrenme mimarileri elde edilmektedir. Sonuç olarak derin öğrenme ağları tabiatları gereği hiyerarşiktir. Derin ağ birçok katmandan oluşmakta ve bu katmanları artan karmaşıklıkla özellik hiyerarşisi oluşturmak için kullanmaktadır. Şekil 4'te verilen yapılara göre, YSA katman sayılarının artırılmasıyla oluşturulmuş farklı türlerde derin öğrenme mimarileri bulunmaktadır. En sık kullanılanlarından biri de ESA'lardır.



Şekil 4. Yapay Sinir Ağ Yapıları (Van Veen ve Leijnen Stefan, 2023)

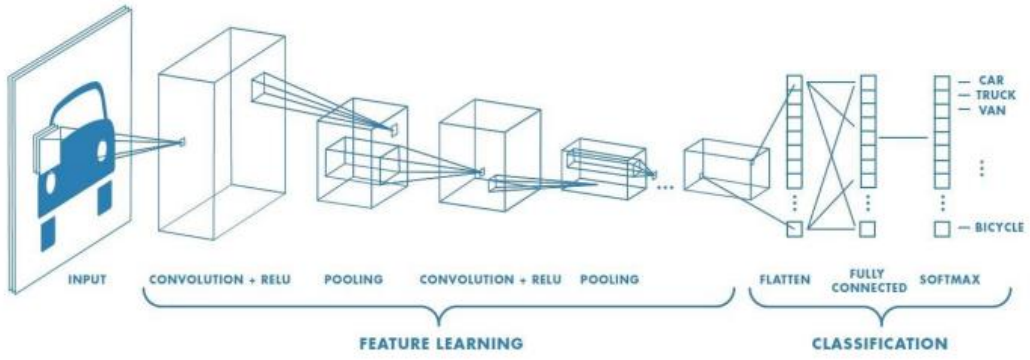
3.4 Evrişimli Sinir Ağları

Sinir ağları, nöral ağlardan etkilenerek oluşturulmuş bilgisayar sistemleridir. Bu tür sistemler, genellikle belirli görevler için özel kurallarla programlanmadan önce, örnekleri inceleyerek görevleri öğrenmektedirler. Bu öğrenme süreci, geçmiş örneklerden bilgi edinme, doğrusal olmayan verileri anlama, belirsiz bilgileri yürütme ve modelin bağlı olmayan verilere uygulanabilme yeteneğini içermektedir. Sinir ağları, yoğun bağlantılı bilgisayar işlemcilerinden oluşan bir ağı içermekte ve günümüzde özellikle görüntü analizi gibi alanlarda en yaygın olarak kullanılan algoritmalar arasında yer almaktadır (Tang, ve diğerleri, 2018).

Evrişimli sinir ağları, güçlü bir YZ tekniğidir. Bu ağlar, problemin uzamsal yapısını korumakla beraber el yazısı, rakam tanıma, karakter tanıma gibi nesne tespiti görevleri için geliştirilmiştir. ESA'lar zorlu bilgisayar görüşünde ve dil işleme görevlerinde başarılı sonuçlar verdiğinden dolayı oldukça popülerdirler. Bu yöntem, temelde bir görüntünün analizini kolaylaştıran bir derin öğrenme algoritmasıdır. Bu algoritma, görüntüdeki farklı objeleri veya nesneleri birbirinden ayırt etmeye yönelik bir işlevi yerine getirmektedir. ESA'lar, küçük giriş verisi kareleri kullanarak dahili özellik temsillerini öğrenmekte, pikseller arasındaki uzamsal ilişkiyi belirlemekte ve korumaktadırlar. Özellikler; tüm görüntüde öğrenilmekte ve kullanılmaktadır. Ayrıca görüntülerdeki nesneler sahnede kaydırılarak veya çevrilerek kullanılsa dahi özellikler yine de ağ tarafından algılanabilmektedir. ESA kullanmanın hem tamamen bağlı bir ağa göre öğrenmek için daha az parametre (ağırlık) kullanması hem de sahnedeki nesne konumuna ve bozulmaya karşı değişmez olacak şekilde tasarlanması gibi faydalarının yanı sıra girdi alanından özellikleri otomatik olarak öğrenip genelleştirme gibi faydaları da bulunmaktadır. Krizhevsky ve diğerleri (2012) çok katmanlı YSA'ların mükemmel özellik öğrenme yeteneklerine sahip olduklarını, katmanlı ön eğitimin, derin sinir ağlarını eğitmenin zorluklarının etkili bir şekilde üstesinden gelebildiğini, bir derin öğrenme çalışması ile ortaya çıkarmışlardır. Krizhevsky ve diğerleri (2012) derin ESA kullanarak o zamanki en iyi sınıflandırma sonucunu ImageNet Büyük Ölçekli Görsel Tanıma Yarışması'nda (LSVRC) elde etmişlerdir, bu da diğer araştırmacıların büyük ilgisini çekmiş ve modern ESA'nın gelişimini büyük ölçüde desteklemiştir (Krizhevsky, Sutskever, ve Hinton, 2012).

ESA'ların ilk katmanlarında, girdi üzerinde özellik çıkarımı yapmak amacıyla filtreler kullanılmaktadır. Bu filtreler, girdi verisi üzerinde belirli özellikleri vurgulamak veya tanımak için kullanılmaktadır. Bu süreç, diğer katmanlara öğrenilen özelliklerin özet bilgisini aktarırken aynı zamanda hesaplama maliyetini düşürmek için boyut düşürme fonksiyonlarıyla desteklenmektedir. Elde edilen özellikler daha sonra tek boyutlu bir vektöre dönüştürülmektedir. Dönüştürülmüş vektör, sınıflandırma veya tahmin gibi işlemler için tam bağlantılı katmanlara girdi olarak verilmektedir. Ağın performansı, öğrenilebilir parametrelerinde (filtreler, ağırlıklar) bir kayıp fonksiyonu kullanılarak hesaplanmaktadır. Bu parametreler, BP yöntemiyle kayıp değerine göre sürekli olarak güncellenmektedir (Nebauer, 1998). Birden fazla aşama ile bir ESA çok seviyeli özellik hiyerarşilerini öğrenebilmektedir.

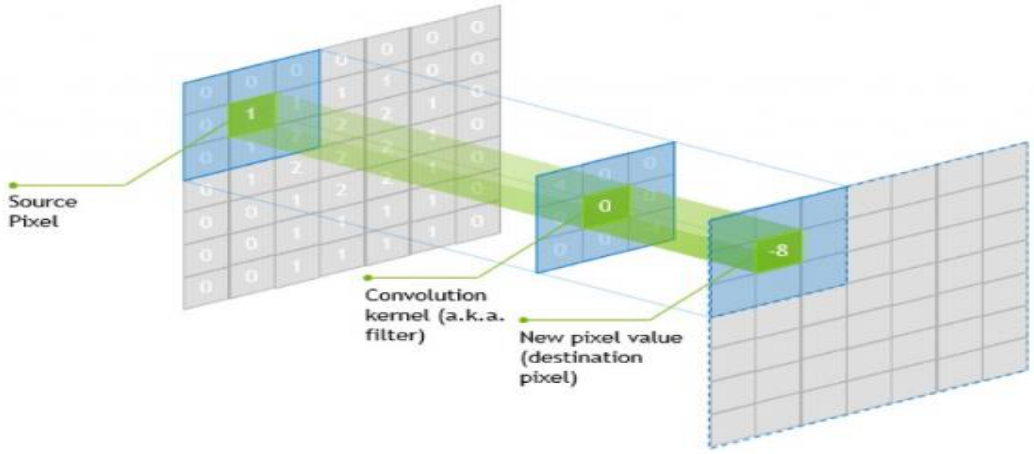
ESA'lar fotoğraflardan nesne tespiti, sayıları, yüzleri ve diğer öğeleri farklı perspektiflerde tanımak için son derece etkilidir. Bu ağları kullanmanın en önemli faydaları; tam bağlantılı ağlara göre daha az parametre/ağırlık kullanması, sahnedeki nesnenin konumuna ve bozulmaya karşı daha dirençli olacak şekilde tasarlanması ve girdi alanındaki özellikleri otomatik olarak öğrenerek genelleştirebilmesidir. Görüntü sınıflandırması için ESA'nın ana avantajı, tüm sistemin ham piksellerden nihai kategorilere kadar uçtan uca eğitilmesidir, bu da uygun bir özellik çıkarıcıyı manuel olarak tasarlama gereksinimini azaltmaktadır. Şekil 5'te görüldüğü üzere ESA'larda evrişim katmanları (convolution layers), havuzlama katmanları (pooling layers) ve tam bağlantılı katmanlar (fully-connected layers) olmak üzere üç katman yer almaktadır.



Şekil 5. ESA Mimarisi Tüm Katmanları (MathWorks, 2023)

3.4.1 Evrişimli katmanlar

Evrişimli katmanlar özellik çıkarıcılar olarak görev yapmaktadır ve böylece giriş görüntülerinin özellik temsillerini öğrenmektedirler. Evrişimli katmanlar, filtrelerden ve özellik haritalarından oluşmaktadırlar. Filtreler, katmanın nöronlarıdır. Değer verdikleri ağırlıklı girdileri almaktadırlar. Giriş boyutu kullanılan filtre boyutlarına, adım (stride) uzunluğuna ve dolgu (padding) ile ilgili parametrelere bağlıdır. Evrişim işlemi sırasında giriş verisinin boyutunun; filtre boyutlarına, adım uzunluğuna ve dolgu miktarına göre nasıl değişeceği belirlenmektedir. Her evrişimli katmanın giriş boyutu, o katmana verilen görüntü veya özellik haritasının boyutunu ifade etmektedir. Evrişimli katman bir önceki katmandaki özellik haritasından girdi almaktadır. Evrişimli katmanlardaki nöronlar özellik haritaları şeklinde düzenlenmiştir. Bir özellik haritasındaki her nöron, bir dizi eğitilebilir ağırlık aracılığıyla bir önceki katmandaki nöronların komşuluğuna dayanan bir alıcı alana sahiptir (LeCun, Bengio, ve Hinton, Deeplearning, 2015). Özellik haritası, ESA'ların önceki katmanına uygulanan bir filtrenin çıktısıdır. Filtreler, her seferinde bir piksel adımıyla önceki katmanın tamamında kaydırılarak uygulanmaktadır. Filtrelemenin her aşamasında, filtrenin önceki katmandaki girdi verisi üzerinde uygulanması, bir nöronun aktivasyonuyla sonuçlanmaktadır. Bu aktivasyonlar, özellik haritasında toplanmakta ve bu, filtrenin belirli bir özellik veya deseni tespit etmeye yönelik katkısını temsil etmektedir. Özellik haritasının her bir pikseli, filtrelerin giriş verisi üzerinde farklı konumlarını tarayarak öğrendiği özellikleri temsil etmektedir. Bu süreç, evrişimli katmanların girdi verisindeki yerel desenlerini algılamak ve daha üst seviyeli özellikleri belirlemek için kullanılmaktadır. ESA'lar, bu şekilde özellikleri hiyerarşik olarak öğrenerek karmaşık görsel hiyerarşiler oluşturabilmektedirler. Yeni bir özellik haritası hesaplamak için girdiler öğrenilen ağırlıklarla konvolide edilmekte ve konvolide edilen hesaplamalar doğrusal olmayan bir aktivasyon işlemi aracılığıyla gönderilmektedir. Alıcı alan, aktivasyondan aktivasyona bir piksel taşınırsa, alanın önceki aktivasyonla giriş değerleri kadar örtüşeceği görülmektedir. Şekil 6'da evrişim katmanının yaptığı işlem gösterilmektedir.



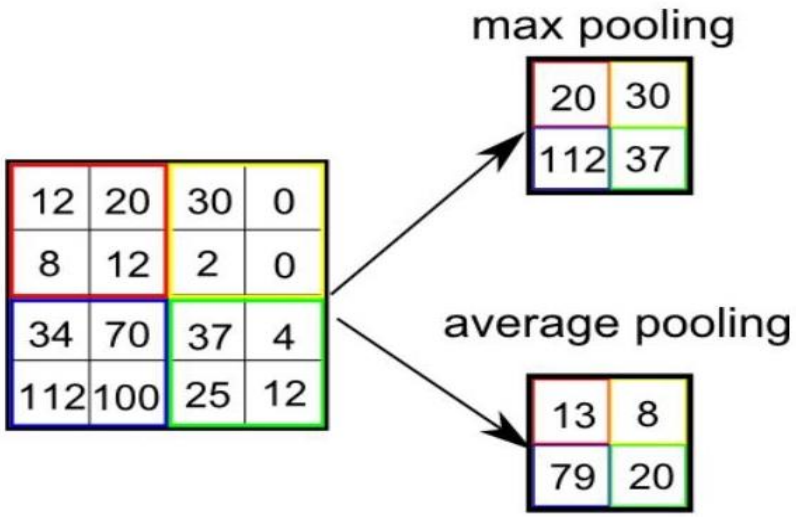
Şekil 6. Evrişim Katmanı (Martin, 2023)

Her bir özellik haritasındaki nöronlar eşit ağırlıklara sahipken, aynı evrişim katmanındaki farklı özellik haritaları farklı ağırlıkları içermektedir. Bu sayede öğrenilen bir konumdan birden çok özellik çıkarılabilmektedir. (Rawat ve Wang, 2017).

3.4.2 Havuzlama katmanları

Havuzlama katmanları, evrişimli katmanın çıktılarının bir veya daha fazlasını alarak önceki katmanın özellik haritasında öğrenilen özellikleri pekiştirmeyi amaçlamaktadır. Havuzlama katmanları özellikleri sıkıştırmak, genelleştirmek ve aşırı uyarlamayı azaltmak için kullanılmaktadırlar. Havuzlama, genellikle evrişimli katmanlardan gelen özellik haritalarını daha küçük boyutlara indirmektedir. Bu da modelin daha genel ve öğrenilmiş özelliklere odaklanmasını sağlamaktadır. Havuzlama, ayrıntıları filtrelemenin bir yoludur. Havuzlama katmanları genellikle daha küçük bir alıcı alana sahiptirler. Her aktivasyon için alıcı alanın taşındığı girdilerin adımı veya sayısı, çakışmaları önlemek ve özelliklerin daha genel olmasını sağlamak için genellikle alıcı alanın boyutuna eşitlenmektedir. Bu katmanlar genellikle iki tür havuzlama işlemi yapmaktadır: maksimum havuzlama ve ortalamalı havuzlama. Maksimum havuzlama, alıcı alandaki en büyük değeri alarak özelliklerin belirgin özelliklerini vurgularken, ortalamalı havuzlama ise alıcı alandaki değerlerin ortalamasını alarak daha genel bir temsil oluşturmaktadır. Şekil 7'de maksimum ve ortalamalı havuzlama katmanlarının nasıl hesaplandığı gösterilmiştir. Maksimum havuzlama toplama ya da ortalama katmanları, bir alıcı alan içindeki ortalama ya da maksimum değeri bir sonraki katmana yaymaktadır. Havuzlama

katmanları, önceki evrişimli katmanların özellik haritasını daha derin ve özgün temsilciler haline getirerek genellikle ağın performansını artırmaya yardımcı olmaktadırlar.



Şekil 7. Havuzlama Katmanı (Peixoto, 2023)

Özellik haritalarının belirgin çözünürlüğünü azaltmak ve böylece girdi bozulmalarına ve ötelemelerine karşı uzamsal değişmezlik elde etmek havuzlama katmanlarının amacıdır (Ranzato, Huang, Boureau, ve LeCun, 2007; LeCun, Bottou, Bengio, ve Haffner, 1998).

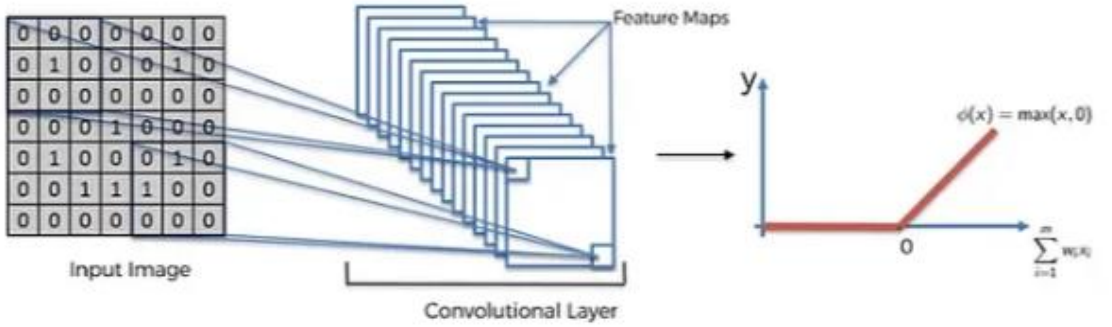
3.4.3 Tam bağlı katmanlar

Tam bağlı katmanlar, tipik bir düz ileri beslemeli sinir ağında bulunan katmanlardır. Bu katmanlar, sınıf tahminlerini olasılık değerlerine dönüştürmek için çoğunlukla softmax aktivasyon fonksiyonuna veya doğrusal olmayan herhangi bir aktivasyon fonksiyonuna sahip olmalıdır. Bu aktivasyon fonksiyonları, ağın çıkışını sınıflara atanabilir olasılıklar haline getirmektedirler ve genellikle sınıflandırma görevlerinde kullanılmaktadırlar. Softmax aktivasyon fonksiyonu, basitliği ve olasılıksal yorumu nedeniyle ESA'ların son tam bağlı katmanında yaygın olarak kullanılmaktadır. Bu aktivasyon fonksiyonu, bir ESA'nın son tam bağlı katmanında çapraz entropi kaybı ile birleştirildiğinde softmax kaybını oluşturmaktadır. (Rawat ve Wang, 2017). Ağın sonunda; evrişim ve havuzlama katmanları tarafından öznitelik çıkarımı ve konsolidasyon gerçekleştirildikten sonra tam bağlı katmanlar kullanılmaktadır. Bu katmanlar, nihai doğrusal olmayan özellik kombinasyonlarını oluşturarak ve ağ tarafından öğrenilen özellik temsillerini

yorumlayarak üst düzey muhakeme işlevini yerine getirmektedirler (Hinton, Krizhevsky, Sutskever, ve Salakhutdinov, 2012).

3.4.3.1 Aktivasyon katmanı

Çok katmanlı bir sinir ağında, üst düğümün çıktısı ile alt düğümün girdisi arasında işlevsel bir ilişki vardır (Hao, Yizhou, Yaqin, ve Zhili, 2020). Sinir ağındaki her nöron düğümü, bir önceki katmanın nöronunun çıkış değerini kendine giriş değeri olarak almaktadır. Girdi katmanı nöron düğümü, giriş öznitelik değerini doğrudan bir sonraki katmana aktarmaktadır. Şekil 8'de gösterilen aktivasyon katmanı, havuzlama katmanının çıkışına ReLU (Düzeltilmiş Lineer Birim - Rectified Linear Unit) gibi doğrusal olmayan bir aktivasyon fonksiyonu uygulamaktadır. Bu işlev, modele doğrusal olmamayı tanıtmaya yardımcı olarak giriş verilerinin daha karmaşık temsillerini öğrenmesine olanak tanımaktadır. Aktivasyon katmanı; ReLU, Binary Step Function, Parametrized ReLU, Tanh, Leaky ReLU, Linear, Exponential Linear Unit, Swish, Sigmoid ve Softmax gibi aktivasyon fonksiyonlarını içermektedir ve çıktı değerini aktivasyon fonksiyonuna göre belirlemektedir.

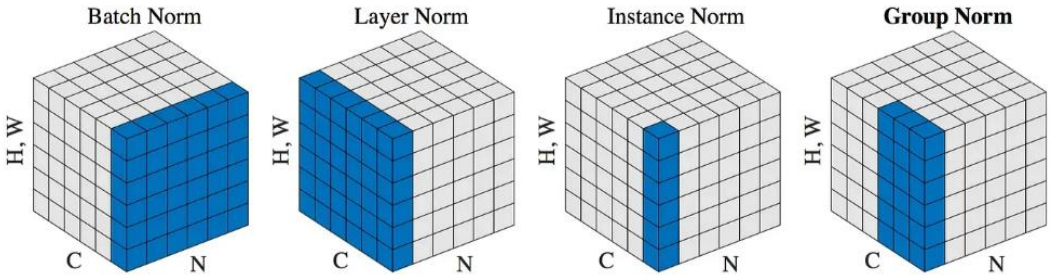


Şekil 8. Aktivasyon Katmanı (Prathammodi, 2023)

3.4.3.2 Normalleştirme katmanı

Sinir ağları her alanda görüntü analizine mükemmel bir şekilde uygulanamamaktadır. Bu sınırlamanın üstesinden gelmek için birçok girişimde bulunulmaktadır. Normalizasyon bu girişimlerden biridir. Normalizasyon, ağ performansını iyileştirmek için kullanılmaktadır ve birçok model için benimsenmiştir (Lee, Sung, Kim, ve Park, 2019). Normalleştirme katmanı, her katmanın aktivasyonlarının iyi koşullandırıldığından emin olmak ve aşırı uyumu engellemek için yığın normalleştirme (batch normalization) veya

katman normalleştirme gibi normalizasyon işlemlerini gerçekleştirmektedir. Yığın normalleştirme, ağların daha hızlı eğitilmesini ve yüksek bir doğruluk elde edilmesini sağlayan en popüler yöntemlerden biridir. Verilerdeki dahili ortak değişken kaymasını azaltarak derin öğrenme modellerinin eğitimi sırasında karşılaşılan sorunları hafifletmektedir. Yığın normalleştirme katmanı eğitim sırasında farklı çıkarım sırasında farklı işlev görmektedir. Eğitim sırasında, ortalama ve varyans her bir yığın için tek tek hesaplanmaktadır. Çıkarım sırasında, ağ tarafından görülen tüm görüntülerin hareketli ortalaması ve varyansı kullanılmaktadır. Yığın normalleştirmenin yalnızca daha iyi sonuçlar sağlamakla kalmayıp, ağın uygun şekilde eğitilmesi için de gerekli olduğu bilinmektedir. Yığın normalleştirme olmadan, çok derin ağlar eğitilememektedir (Thakkar ve Tewary, 2018). Şekil 9'da normalleştirme katman türleri gösterilmiştir.

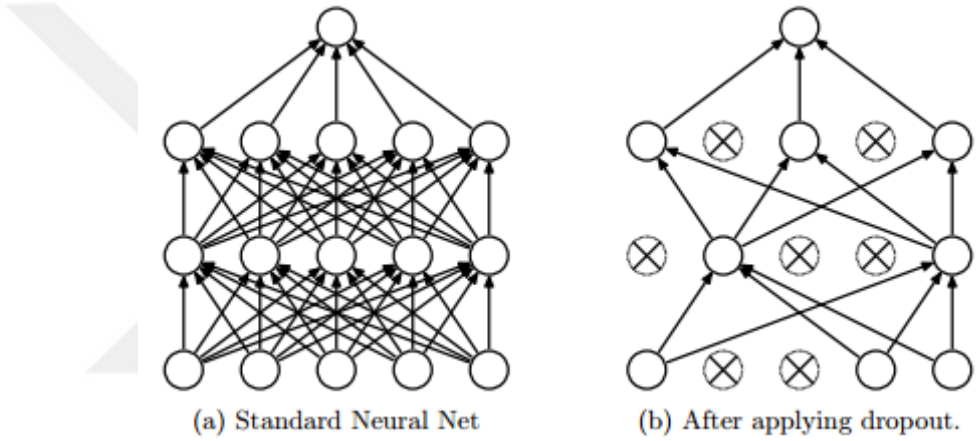


Şekil 9. Normalleştirme Katman Türleri (Prathammodi, 2023)

3.4.3.3 Düğüm seyreltme katmanı

Düğüm seyreltme (dropout) aşırı öğrenmeye (overfitting) karşı mücadele etmek için önerilen bir düzenleyicidir (Hinton, Krizhevsky, Sutskever, ve Salakhutdinov, 2012). Eğitim zamanında her bir eğitim durumu için gizli birimlerin aktivasyonlarını stokastik olarak sıfıra ayarlayan bir düzenleyici yöntemdir. Bu, özellik dedektörlerinin birlikte uyarlanmasını bozmaktadır çünkü bırakılan birimler diğer tutulan birimleri etkileyememektedir. Temel olarak Şekil 10'da gösterildiği gibi, gizli katman veya girdi katmanındaki belirli düğümlerin (eşik değeri kullanarak veya rastgele seçilerek) kaldırılması işlemidir. Düğüm seyreltme tekniği genellikle tam bağlantılı katmanlardan sonra uygulanmaktadır. Bu teknik, tam bağlantılı katmanlardaki bağları koparıp düğümleri birbirinden daha bağımsız hale getirmektedir. Sonuç olarak, düğümler

birbirleriyle ilgili fazla bilgiye sahip olmamakta ve bu da düğümlerin birbirlerinin ağırlık değişimlerinden daha az etkilenmesine yol açmaktadır. Bu, modelin eğitim verilerini ezberlemesini önlemekte, bunun yerine yeni, görünmeyen verilere genelleme yapmasını sağlamaya yardımcı olmaktadır (Haibing ve Gu, 2015). Sinir ağlarını düzenli hale getirmek, aşırı uyumu azaltmak için önemli bir görevdir. Düğüm seyreltme, sinir ağlarında genellikle kullanılan düzenlileştirme yöntemleri için etkilidir. ESA'larda, düğüm seyreltme genellikle tam bağlı katmanlara uygulanır. Evrişimsel katmanlarda bırakmanın genelleştirme etkisi, tam bağlantılı katmanlarda olduğu gibi model ortalamasından değil, evrişimsel katmanların girişlerine gürültü eklenerek sağlamlığın artırılmasından kaynaklanmaktadır (Park ve Kwak, 2017).

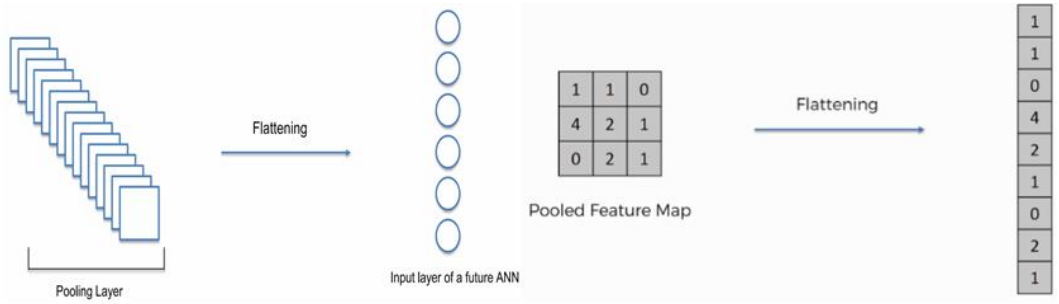


Şekil 10. Düğüm Seyreltme Katmanı (Peixoto, 2023)

3.4.3.4 Düzleştirme katmanı

Düzleştirme katmanı (flatten layer), ESA'larda genellikle evrişim ve havuzlama katmanlarından sonra kullanılmaktadır. Bu katman, evrişim ve havuzlama katmanlarının çıktılarını almakta ve bu çıktıları tek boyutlu bir vektöre dönüştürmektedir. Evrişim ve havuzlama katmanları, giriş görüntülerindeki özellikleri çıkarmak için çeşitli filtreleri uygulamakta ve bu özellikleri özel haritalarda saklamaktadır. Ancak, tam bağlı (fully connected) katmanlara geçiş yapmadan önce, 2D veya 3D özelliğindeki çıktıların tek boyutlu bir vektöre düzleştirilmesi gerekmektedir. Düzleştirme katmanı, önceki katmanlardan gelen 2D veya 3D tensörleri almakta ve bunları tek boyutlu bir vektöre dönüştürerek, bu vektörü tam bağlı katmanlara giriş verisi olarak sağlamaktadır. Şekil 11'de düzleştirme katmanı işlemi gösterilmiştir. Düzleştirme katmanı, evrişim ve

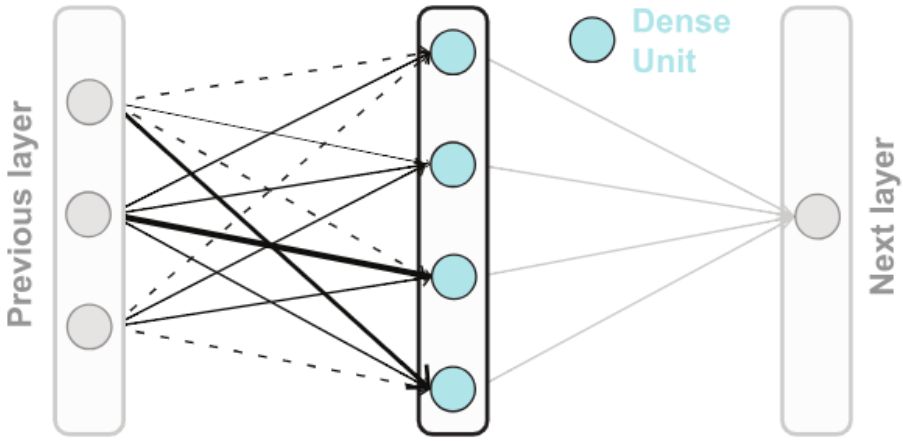
havuzlama katmanlarının çıktılarını uygun bir şekilde düzenleyerek, ağın evrişimli ve havuzlama katmanları tarafından öğrenilen özelliklerini daha üst düzeyde bir sınıflandırma katmanına yani tam bağlı katmanlara veya çıktı katmanına iletmek için kullanılmaktadır.



Şekil 11. Düzleştirme Katmanı (Team, 2023)

3.4.3.5 Yoğun katman

Evrişim ve havuzlama katmanları, gelen giriş görüntülerinden özellikler çıkarıp, bu özellikleri birleştirmek ve son bir tahmin yapmak için yoğun (dense) katmanını kullanmaktadırlar. Bir ESA'da genellikle son katman olan dense katmanı, çıktı tahminlerini üretmek için kullanılmaktadır. Bu katmanda, önceki katmanlardan gelen aktivasyonlar düzleştirilmektedir. Şekil 12'de yoğun katman gösterilmiştir. Sonrasında ağırlıklı toplam alınıp bir aktivasyon fonksiyonu uygulanarak nihai çıktı elde edilmektedir. Yoğun katman, derinlemesine bağlı bir sinir ağı katmanıdır, yani her nöron bu katmandaki tüm önceki nöronlardan girdi almaktadır. Bu nedenle tamamen bağlı bir yapıya sahiptir ve tam bağlı katman olarak bilinmektedir (Jha, ve diğerleri, 2020). Bu katman, genellikle vektör boyutlarını değiştirmek için kullanılmakta ve modellerde sıkça tercih edilmektedir. Ayrıca vektör üzerinde dönüşüm, ölçekleme ve kaydırma gibi işlemler için de kullanılmaktadır.

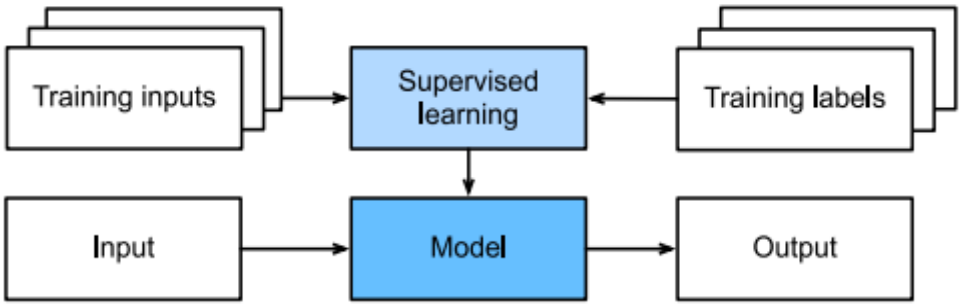


Şekil 12. Yoğun Katman (Canu, 2023)

3.5 Denetimli Öğrenme

Denetimli öğrenme (supervised learning), makine öğrenmesinde ciddi bir araştırma konusudur ve birçok denetimli öğrenme tekniği, özellikle multimedya içeriğinin işlenmesinde uygulanmaktadır. Denetimli makine öğrenmesi, dışarıdan sağlanan örneklerden akıl yürüterek genel hipotezler üreten ve bu hipotezlere dayanarak gelecekteki örneklerle ilgili tahminlerde bulunan bir algoritma araştırmasıdır (Kotsiantis, Zaharakis, ve Pintelas, 2007). Denetimli öğrenmenin temel özelliği, açıklamalı eğitim verilerinin bulunmasıdır. Denetimli algoritmalarda sınıflar önceden belirlenmiştir. Bu sınıflar; çalışma yapan araştırmacılar tarafından tanımlanan sonlu bir küme şeklinde oluşturulur, bu da pratikte belirli bir veri parçasının sınıflandırmalarla etiketleneceği anlamına gelmektedir. Makine öğrenmesi algoritmasının görevi örüntüleri bulmak ve matematiksel modeller oluşturmaktır. Bu modeller daha sonra verinin kendi içindeki varyans ölçüleriyle ilişkili olarak tahmin kapasitesine göre değerlendirilmektedir (Nasteski, 2017). Denetimli öğrenmenin amacı, girdi ve çıktı arasındaki eşlemeyi öğrenebilen ve yeni girdiler verildiğinde sistemin çıktısını tahmin edebilen yapay bir sistem oluşturmaktır (Liu ve Wu, 2011). Denetimli öğrenme; öğretmenle öğrenme (Haykin, 1998) olarak da adlandırılmaktadır. Denetimli öğrenme, özellikleri ve etiketleri içeren bir veri kümesinin verildiği, giriş özellikleriyle etiketleri tahmin eden bir modelin oluşturulduğu bir öğrenme türünü ifade etmektedir. Her özellik-etiket çiftine örnek denilmektedir. Denetimli öğrenme, denetleyicilerin modele etiketli örneklerden oluşan bir veri kümesi sağladığı süreci içermektedir. Genellikle, girdi özellikleri verilen bir

etiketin koşullu olasılığını tahmin etmeye odaklanılmaktadır. Denetimli öğrenme, birçok başarılı makine öğrenmesi uygulamasının temelini oluşturmaktadır (Zhang, Lipton, Li, ve Smola, 2023). Şekil 13'te denetimli öğrenme uygulama basamakları görülmektedir.



Şekil 13. Denetimli Öğrenme (Zhang, Lipton, Li, ve Smola, 2023)

3.6 Transfer Öğrenme

Transfer öğrenme (transfer learning), bir modelin bir görev için eğitildiği ve geliştirildiği ve daha sonra ikinci bir ilgili görevde yeniden kullanıldığı bir makine öğrenmesi tekniğidir. Genel olarak, transfer öğrenmenin amacı, ilgili görevlerden elde edilen eğitim verilerini, gelecekte ilgilenilen başka bir problem üzerinde öğrenmeye yardımcı olmak için kullanmaktır (Quattoni, Collins, ve Darrell, 2008). Kısacası bir ortamda öğrenilenlerin başka bir ortamda optimizasyonu iyileştirmek için kullanılması durumudur. Transfer öğrenme genellikle önceden eğitilmiş modeli eğitmek için kullanılan orijinal veri kümesinden daha küçük yeni bir veri kümesi olduğunda uygulanmaktadır (Larsen-Freeman, 2013).

Transfer öğrenme kullanılarak oluşturulmuş bir modeli güncellemek ve tekrar eğitmek, genellikle sıfırdan bir ağ inşa etmek ve eğitmekten daha hızlı ve kolay olabilmektedir. Tipik olarak, önceden eğitilmiş bir model, geniş veri setleri üzerinde eğitilmiş büyük ağ yapılarına sahiptir ve bu ağın öğrenilmiş ağırlıkları, yeni bir model oluşturulduğunda kullanılabilmektedir. Derin öğrenmede kullanılan transfer öğrenimini diğer yöntemlerden ayıran en önemli avantajlardan biri, öğrenilen özelliklerin taşınabilir olma yeteneğidir. Bu özellik, derin öğrenme modellerinde farklı görevlerde veya veri setlerinde de kullanılabilmesine olanak sağlamaktadır. Bu özelliği sayesinde transfer öğrenme, küçük veri problemlerinde etkin olmaktadır (Chollet, 2021). Transfer öğrenme modelleri özellik

çıkarımı için uygulanırken son katman çıkarılarak kullanılmaktadır. Oluşturulan yeni model, diğer katmanlarla birleştirildikten sonra, sabit özellikleri çıkarmak üzere tasarlanmış ve daha sonra diğer görevleri gerçekleştirmek için kullanılan bir özellik çıkarıcı olarak görev yapmaktadır (Ribani ve Marengoni, 2019).

3.7 Hiper-Parametreler

Hiper-parametreler, farklı veri modellerini tahmin etmek için gereken kısıtlamalar, öğrenme oranları veya ağırlıklar gibi farklı parametrelerdir. Bu parametreler, model doğruluğunu artırmak ve makine öğrenmesi problemlerini en iyi şekilde çözmek için ayarlanmaktadır. Hiper-parametrelerin ayarlanması, en aza indirilen bir kayıp fonksiyonu ile en iyi modeli bulmak için önceden tanımlanmış bağımsız veriler üzerinde gerçekleştirilmektedir. Bazı hiper-parametreler algoritmaların eğitim süresini ve maliyetini etkilerken, bazıları modelin genelleme performansını etkilemektedir (Ismail, Ahmad, Soh, Hassan, ve Harith, 2019). Hiper-parametre, makine öğrenmesi eğitimi sırasında güncellenemeyen parametreleri ifade etmektedir. Gizli katmanlar ve aktivasyon fonksiyonları modelin yapısının oluşturulmasında veya stokastik gradyan inişinin (SGD) öğrenme oranı (LR), yığın boyutu ve optimize edici gibi model eğitiminin verimliliğini ve doğruluğunu belirlemede yer alabilmektedirler. (Yu ve Zhu, 2020). Derin öğrenmede en sık kullanılan hiper-parametreler; katman boyutu, büyüklük (momentum, LR), düzenli hale getirme (dropout, drop connect, L1, L2), optimizasyon algoritması, mini-yığın (minibatch) boyutu, aktivasyonlar, eğitim tur sayısı, vs. sayılabilir. Hiper-parametreleri doğru ayarlamak genellikle algoritmaların en iyi sonuçları vermesi için kritik öneme sahiptir ve ayarlama işi zaman alan bir süreçtir (Han, Liu, ve Fan, 2018; Patterson ve Gibson, 2017).

3.7.1 Katman boyutu

Katman boyutu, bir katmandaki nöron sayısıyla tanımlanmaktadır. Girdi ve çıktı katmanlarını belirlemek genellikle basittir ancak gizli katmanlardaki nöron sayıları seçimi hiper-parametre ayarını zorlaştırmaktadır. Gizli katmanlar için ayrı ayrı belirlenen nöron sayısı seçimi, modelin karmaşıklığına doğrudan etki etmektedir. Bu seçimde herhangi bir katmanın nöron sayısının belirlenmesinde kesin bir kural bulunmamaktadır

ve problemin ne kadar karmaşık modellenebileceği, özellikle gizli katmanlardaki nöron sayısıyla doğru orantılıdır (Patterson ve Gibson, 2017).

3.7.2 Mini-yığın

Modele eğitim sürecinde tüm veri setini girdi olarak vermek yerine, veriyi küçük parçalar halinde bölmemizi sağlayan hiper-parametredir. Eğitim sırasında, her epokta eğitim setinin yalnızca bir kısmını kullanmaktadırlar. Derin öğrenme eğitimlerinde, mini-yığın yöntemleri çoğunlukla sinir ağlarının eğitimini hızlandırmalarından, rastgele seçilmelerinden ve beklenen gradyanın yansız bir tahminini hesaplayabilmelerinden dolayı tercih edilmektedir. Mini-yığın boyutu bir ile birkaç yüz arasında seçilebilmektedir. Mini-yığınlar 32, 64, 128, 256 vb. değerleri alabilmektedir. Örneğin B = 32 iyi bir varsayılan değerdir, 10'un üzerindeki değerler matris-vektör çarpımlarına göre matris-matris çarpımlarının hızlanmasından yararlanmaktadır. B'nin etkisi çoğunlukla hesaplama üzerinedir, yani daha büyük B daha hızlı hesaplama sağlamakta, ancak epok başına daha az güncelleme olduğundan aynı hataya ulaşmak için daha fazla örneğin ziyaret edilmesini gerektirmektedir. Teorik olarak, bu hiper-parametre eğitim süresini etkilemeli fakat test performansını çok fazla etkilememelidir. Bu nedenle diğer hiper-parametreler (öğrenme oranı hariç) seçildikten sonra eğitim eğrileri (eğitim ve doğrulama hatası ile eğitim süresi miktarı) karşılaştırılarak ayrı olarak optimize edilebilmektedir (Bengio, 2012).

3.7.3 Optimizasyon algoritmaları

Her ağırlık kümesi, girdilerin ne anlama geldiği yani kişinin etiketlerinde yer alan anlamlarla nasıl ilişkili oldukları hakkında belirli bir hipotezi temsil etmektedir. Ağırlıklar, ağların girdileri ile tahmin etmeye çalıştıkları hedef etiketler arasındaki korelasyonlar hakkındaki varsayımları temsil etmektedir. Bu problemdeki tüm olası ağırlıkları ve bunların kombinasyonları, hipotez uzayını oluşturmaktadır. En iyi hipotezi formüle etme işlemi, hipotez uzayında arama yapma problemidir ve bu optimizasyon algoritmaları kullanarak yapılmaktadır. Elimizde ne kadar çok girdi parametresi varsa, problemin arama uzayı da o kadar büyük olmaktadır. Öğrenme işinin büyük bir kısmı hangi parametrelerin göz ardı edileceğine ve hangilerinin dinleneceğine karar vermektir (Patterson ve Gibson, 2017). Keras kütüphanesi; SGD, RMSProp, AdaGrad, AdataDelta,

Adam, AdamW, Adamax ve Nadam gibi bir dizi iyileştirici uygulanmasına olanak sağlamaktadır.

Adam (Adaptive Moment Estimation), klasik stokastik gradyan inişinden farklıdır. Gradyan iniş bir optimizasyon algoritmasıdır. Açıkça hedef amaç fonksiyonunun birinci dereceden türevini kullandığı için teknik olarak birinci dereceden bir optimizasyon algoritması olarak anılmaktadır. Birinci dereceden yöntemler, aramayı minimuma yönlendirmeye yardımcı olmak için gradyan bilgilerine dayanmaktadır (Kochenderfer ve Wheeler, 2019). SGD, tüm ağırlık güncellemeleri için tek bir öğrenme oranını (alfa olarak adlandırılır) korumakta ve öğrenme oranı, eğitim sırasında değişmemektedir. Her ağ ağırlığı/parametresi için bir öğrenme oranı korumakta ve öğrenme geliştikçe ayrı ayrı uyarlanmaktadır. Bu yöntem, gradyanların birinci ve ikinci moment tahminlerine dayanarak farklı parametreler için özelleştirilmiş öğrenme oranlarını hesaplamaktadır (Ruder, 2016). Kingma ve Ba (2014), Adam'ın SGD'nin diğer iki uzantısının avantajlarını birleştirdiğini ifade etmişlerdir. Aynı zamanda Adam'ın, hem gradyanların birinci momenti olan hareketli ortalamayı hem de gradyanların ikinci momenti olan karesel ortalamanın karekök yayılımını (RMSProp) birleştirdiğinden bahsetmişlerdir. Adam, hem AdaGrad (uyarlamalı gradyan algoritması) hem de RMSProp'un birleştirilmiş olan özelliklerini de taşımaktadır (Kingma ve Ba, 2014). Adam algoritması, genellikle derin öğrenme modellerinin eğitiminde tercih edilen optimizasyon algoritmalarından biridir. AdamW optimizasyonu ise, Loshchilov ve Hutter (2019) çalışmalarındaki tekniklere göre ağırlıkları güncelleme işlemi sırasında ağırlıkların kendisiyle doğrudan ilişkili olan bir düzenleme teriminden bahsetmişlerdir. Bu düzenleme terimi, ağırlıkların büyüklüğünü kontrol etmek ve aşırı uydurmayı (overfitting) azaltmak için eklenmektedir (Loshchilov ve Hutter, 2019). AdamW, özellikle büyük ölçekli derin öğrenme modelleri ve transfer öğrenme problemleri için yaygın olarak kullanılmaktadır.

3.7.4 Öğrenme oranı

Öğrenme oranı (LR, Learning rate), her epok veya her veri noktası (mini-yığın) üzerinde modelin güncellenmesi sırasında kullanılan adım uzunluğunu belirleyen pozitif bir ölçüdür (Goodfellow, Bengio, ve Courville, 2015). Çoğu durumda, LR model eğitimi sırasında genellikle deneme yanılma yöntemiyle belirlenmekte ve probleme dayalı modelin karmaşıklığına bağlı olarak gelişmiş doğruluk için gerekmektedir (Bengio,

2012). Sabit LR yerine alternatif olarak eğitim süreci boyunca değişen bir LR kullanılabilmektedir. Bu yöntem LR programı veya LR bozunumu olarak adlandırılmaktadır (Goodfellow, Bengio, ve Courville, 2015). Uyarlanabilir LR, modelin performansına veya yapısına göre öğrenme oranını sabit bir değere ayarlamak yerine bir değer bandı içinde döngüsel olarak değişmesine izin verilerek ayarlanabilmekte ve bir öğrenme algoritması tarafından desteklenmektedir (Smith, 2017). En basit program olan sabit LR, genellikle derin öğrenme çerçeveleri (örneğin Keras) tarafından varsayılan olarak ayarlanmaktadır. LR için iyi bir değer belirlemek önemli ve zor bir görevdir. Uygun bir sabit LR ile ağ, özellikle son birkaç adımda, başlangıç değeri her zaman aşırı büyük belirlendiğinden, kötü olmayan ancak tatmin edici bir doğrulukla eğitilebilir. Sabit değere dayalı küçük bir iyileştirme için, başlangıç oranını örneğin 0,1 olarak ayarlamak, doğruluk doygunluğa ulaştığında 0,01'e hatta gerekirse 0,001'e ayarlamak mümkün olmaktadır.

3.7.5 L1-L2 düzenlileştirmesi

Matematiksel gösterimde, lambda katsayısı ile temsil edilen düzenli hale getirmenin, özellikler üzerindeki üsler arttıkça iyi bir uyum bulma ve belirli özellik ağırlıklarının değerini düşük tutma arasındaki dengeyi kontrol ettiği bilinmektedir. Düzenli hale getirme katsayıları L1 ve L2, aşırı uyumun etkilerini azaltabilen iki yaygın düzenlileştirme tekniğidir ve belirli ağırlıkları küçülterek aşırı uyumla mücadeleye yardımcı olmaktadır (Patterson ve Gibson, 2017). L1 ve L2 düzenlileştirmeleri, bir amaç fonksiyonu ile ya da BP algoritmasının bir kısmı olarak çalışabilmektedir. Bir amaç fonksiyonu ile ya da BP algoritmasının bir kısmı olarak çalıştığında, düzenli hale getirme algoritmaları, ek bir amaç eklenerek eğitim algoritmasına bağlanmaktadır. Bu algoritmaların her ikisi de sinir ağı eğitimine bir ağırlık cezası ekleyerek çalışmaktadır. Bu ceza, sinir ağını ağırlıkları küçük değerlerde tutmaya teşvik etmektedir. Hem L1 hem de L2 bu cezayı farklı şekilde hesaplamaktadır aynı zamanda hem L1 hem de L2 bir ağırlığın boyutunu cezalandırma biçiminde de farklı çalışmaktadırlar. L1 düzenlemesi, ağırlıkların mutlak değerlerine bağlı olarak bir düzenleme terimi eklemektedir. L2 düzenlemesi ise ağırlıkların karelerine bağlı olarak bir düzenleme terimi eklemektedir (Ng, 2004).

Sinir ağında seyreklik yaratmak için L1 düzenlemesi kullanılabilmektedir. Başka bir deyişle, L1 algoritması birçok ağırlık bağlantısını sıfıra yaklaştırmaktadır. Bir ağırlık 0'a yaklaştığında, program onu ağdan düşürmektedir. Ağırlıklı bağlantıların düşürülmesi seyrek bir sinir ağı oluşturmaktadır. Özellik seçimi, seyrek sinir ağlarının faydalı bir yan ürünüdür. Özellikler, eğitim setinin giriş nöronlarına sağladığı değerlerdir. Bir giriş nöronunun tüm ağırlıkları 0'a ulaştığında, sinir ağı eğitimi özelliğin gereksiz olduğunu belirlemektedir. Veri setinde ihtiyaç duyulmayan çok sayıda giriş özelliği varsa, L1 düzenlemesi sinir ağının gereksiz özellikleri tespit etmesine ve göz ardı etmesine yardımcı olabilmektedir.

Bir uzay ağı oluşturma konusunda daha az endişe duyulduğunda ve düşük ağırlık değerleri konusunda daha fazla endişe duyulduğunda L2 düzenlileştirmesi kullanılabilmektedir. Düşük ağırlık değerleri tipik olarak daha az aşırı uyuma yol açmaktadır (Heaton, 2015).

3.7.6 Eğitim tur sayısı

Eğitim tur sayıları epok terimi ile ifade edilmektedir. Bir epok (epoch), modelin tüm eğitim veri setini bir kez gördüğü bir döngüdür yani tüm veri kümesinin ağdan bir kez geçtiği, tüm iterasyonların veri gruplarına göre tamamlandığı zamandır (Sarkar, Bali, ve Ghosh, 2018).

Derin öğrenme modelinin eğitimi, tüm verilerin aynı anda değil, belli sayıda parçalar halinde eğitime katılmasıyla gerçekleşmektedir. Her bir eğitim adımında, önceki adımda öğrenilen ağırlıklar kullanılarak model eğitilmekte ve başarısı test edilmektedir. BP yöntemiyle ağırlıklar güncellenmekte ve bu işlem, belirlenen epok sayısına kadar tekrarlanmaktadır. Epok sayısı arttıkça başarım genellikle artmakta, belirli bir epoktan sonra öğrenme hızı azalabilmektedir. Model eğitimi için genellikle uzun süreler harcanmakta ve bu nedenle diğer hiper-parametrelerle birlikte eğitim süreci mümkün olduğunca kısaltılmaya çalışılmaktadır.

Bu hiper-parametre, erken durdurma ilkesi kullanılarak neredeyse maliyetsiz optimize edilebilmesi açısından özeldir: eğitim ilerledikçe örneklem dışı hatayı (örneğin bir doğrulama kümesinde tahmin edildiği gibi) takip ederek, diğer tüm hiper-parametrelerin herhangi bir ayarı için ne kadar süre eğitileceğine karar verebilmektedir. Erken durdurma,

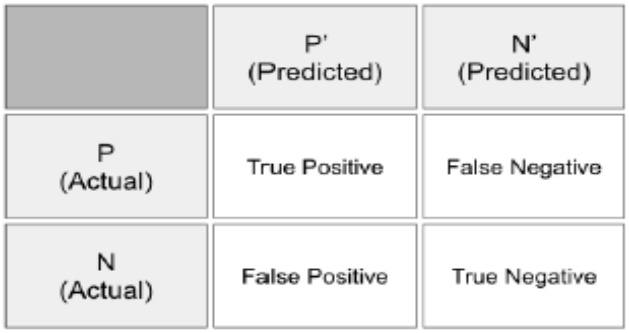
güçlü aşırı uyumdan kaçınmak için iyi bir yoldur, her ne kadar diğer hiper-parametreler aşırı uyuma neden olsa da erken durdurma ortaya çıkacak aşırı uyum hasarını önemli ölçüde azaltmaktadır.

3.8 Değerlendirme Metrikleri

Metrikler bir modelin performansını değerlendirmekte ve yapılan sınıflandırmanın ne kadar başarılı olduğunu göstermektedir. Fakat her bir metrik bunu farklı bir şekilde değerlendirmektedir.

3.8.1 Karmaşıklık matrisi

Karmaşıklık matrisi (Confusion Matrix), model tahminlerine karşı temel gerçek etiketlerinin tablo halinde görselleştirilmesidir. Karmaşıklık matrisinin her bir satırı tahmin edilen sınıftaki örnekleri, her bir sütunu ise gerçek sınıftaki örnekleri temsil etmektedir. Şekil 14'te karmaşıklık matrisi elemanları gösterilmiştir.



Şekil 14. Karmaşıklık Matrisi (Patterson ve Gibson, 2017)

Karmaşıklık matrisleri tahmin edilen ve gerçek değerlerden gelen sayıları temsil eder. Doğru Pozitif (TP-True Positive), modelin kaç tane pozitif sınıf örneğini doğru tahmin ettiğini göstermektedir. Doğru Negatif (TN-True Negative), modelin kaç tane negatif sınıf örneğini doğru tahmin ettiğini göstermektedir. Yanlış Pozitif (FP-False Positive), modelin kaç tane negatif sınıf örneğini yanlış tahmin ettiğini göstermektedir. Yanlış Negatif (FN-False Negative), modelin kaç tane pozitif sınıf örneğini yanlış tahmin ettiğini göstermektedir. Python'da karmaşıklık matrisi sklearn kütüphanesinin bir parçası olan confusion_matrix() fonksiyonu kullanılarak elde edilmektedir. Karmaşıklık matrisinde

sınıflandırma yapılırken en sık kullanılan metriklerden biri doğruluktur (Kulkarni, Batarseh, ve Chong, 2020).

3.8.2 Doğruluk

Sınıflandırma doğruluğu (accuracy-acc) kullanımı ve uygulaması en basit metriklerden biridir. Doğru tahminlerin sayısının toplam tahmin sayısına bölünüp 100 ile çarpılması sonucunda elde edilmektedir. Bir döngü içinde temel gerçek ve tahmin edilen değerleri karşılaştırarak veya scikit-learn modülünü kullanarak uygulanmaktadır. Denklem 3.1'de formülü gösterilmiştir.

$$\mathrm{ACC} = (\mathrm{TP} + \mathrm{TN})/(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}) \tag{3.1}$$

3.8.3 Kesinlik

Kesinlik (precision), pozitif olarak tahmin edilen örneklerin kaç tanesinin doğru tahmin edildiğini göstermektedir. Tahmin edilen olumlu bir sonucun doğruluğunu ölçmektedir. Python'da precision_score() fonksiyonu kullanılarak hesaplanmaktadır. Denklem 3.2'de formülü gösterilmiştir.

$$\mathrm{P} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FP}) \tag{3.2}$$

3.8.4 Duyarlılık, geri çağırma

Pozitif olarak tahmin edilmesi gereken örneklerin oransal olarak kaç tanesinin doğru tahmin edildiğini göstermektedir. Duyarlılık (recall), bir modelin pozitif sonuçları tahmin etme performansını ölçmek için kullanılmakta ve aynı zamanda bir modelin duyarlılığı

olarak da bilinmektedir. Python'da recall_score() fonksiyonu kullanılarak hesaplanmaktadır. Denklem 3.3'te formülü gösterilmiştir.

$$R = TP/(TP + FN) \tag{3.3}$$

3.8.5 F1 skoru

F1 skoru, bir sınıflandırma modelinin performansını ölçen bir metriktir. Doğruluk, kesinlik, ve duyarlılık değerlerinin bir kombinasyonunu temsil etmektedir. F1 skoru, kesinliği ve duyarlılığı dengeleyerek, modelin sınıflandırma yapma yeteneğini değerlendirmektedir. Pozitif örneklerin sınıflandırılması için, doğruluk ve kapsam arasındaki dengenin anlaşılmasına yardımcı olmaktadır. Dengesiz sınıflar için kullanılan yararlı bir hibrit metriktir. Denklem 3.4'te formülü gösterilmiştir.

$$F1 = 2\ 2TP/(2TP + FP + FN) \tag{3.4}$$

3.8.6 Eğri altında kalan alan

Eğri Altındaki Alan (Area Under Curve – AUC) skorları, ikili sınıflandırma modellerinin performansını değerlendirmek için yaygın olarak kullanılan bir metriktir. Geleneksel AUC skoru, tüm olası eşik değerleri için Doğru Pozitif Oranı (TPR) ile Yanlış Pozitif Oranını (FPR) gösteren ROC eğrisinin altındaki alanı hesaplamaktadır. Puan, rastgele bir model için 0,5 ile mükemmel bir model için 1 arasında değişir ve bire yakın değerler daha iyi bir performansa işaret etmektedir (Ssawla, 2023).

ROC eğrisi, sınıflandırıcı performansının iki boyutlu bir tasviridir. Sınıflandırıcıları karşılaştırmak için ROC performansı beklenen performansı temsil eden tek bir skaler

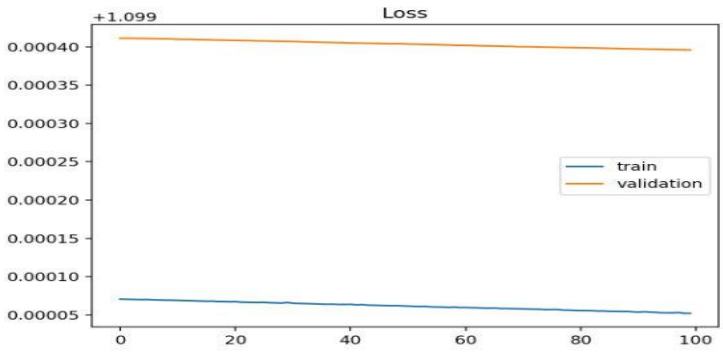
değere indirgenebilir. Yaygın bir yöntem olan AUC, kısaltılan ROC eğrisi altındaki alanı hesaplamaktır (Bradley, 1997; Hanley ve McNeil, 1982). AUC, birim karenin alanının bir kısmını belirtmekte ve değeri her zaman 0 ile 1 arasında olmaktadır. Bununla birlikte, rastgele tahmin 0,0 ile 1,1 arasında 0,5'lik bir alana sahip olan diyagonal çizgiyi ürettiğinden, hiçbir gerçekçi sınıflandırıcı 0,5'ten daha düşük bir AUC değeri almamaktadır (Fawcett, 2006).

3.8.7 mAP değeri

Ortalama kesinlik (mAP, Mean Average Precision), bir nesne tespiti modelinin performansını değerlendirmek için kullanılan bir ölçüdür. YOLO gibi çalışan nesne tespit algoritmaları, modellerini değerlendirmek için mAP kullanmaktadır. Farklı tespit eşikleri (thresholds) kullanılarak kesinlik hesaplanmakta ve bu değerlerin ortalaması alınmaktadır. Bu ortalama sıfır ile bir arasındaki geri çağırma değerleri üzerinden hesaplanmaktadır. mAP metriğinin formülü; karmaşıklık matrisi, birlik üzerinde kesişim (Intersection over Union, IoU), geri çağırma ve kesinlik gibi alt metriklere dayanmaktadır. Nesne tespit modelleri, bir görüntüdeki nesnelerin sınırlayıcı kutusunu ve kategorisini tahmin etmektedirler. IoU, sınırlayıcı kutunun doğru şekilde tahmin edilip edilmediğini belirlemek için kullanılmaktadır. IoU, sınırlayıcı kutunun ne kadar örtüştüğünü göstermektedir. İki sınırlayıcı kutunun bölgeleri arasındaki bu örtüşme oranı, tam eşleşme durumunda 1,0 örtüşme olmaması durumunda ise 0,0 olmaktadır.

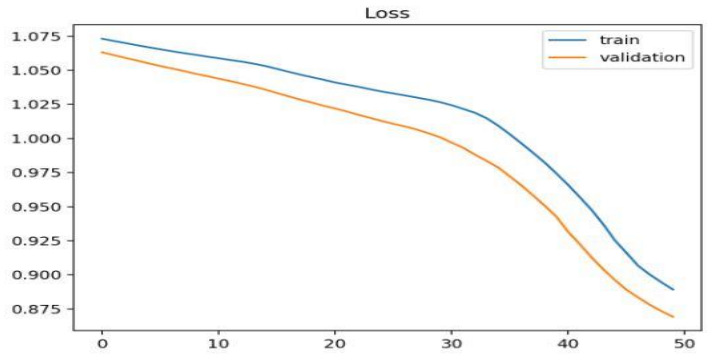
3.9 Yetersiz Uyum, Aşırı Uydurma ve İyi Form

Yetersiz uyum (underfitting), bir modelin eğitim veri setini etkili bir şekilde öğrenemediği durumu ifade etmektedir. Yetersiz uyum, modelin eğitim setinde beklenen performans düzeyine ulaşamadığını veya yeterince öğrenemediğini göstermektedir. Bu durum genellikle modelin karmaşıklığının eğitim veri setine uygun olmadığı veya modelin yeterince öğrenme esnekliğine sahip olmadığı durumlarda ortaya çıkmaktadır (Goodfellow, Bengio, ve Courville, 2015). Yetersiz bir model, yalnızca eğitim kaybının öğrenme eğrisinden belirlenebilmektedir. Grafikte düz bir çizgi gibi gözükmekte, bu da modelin eğitim veri setini asla öğrenemediğini göstermektedir. Yetersiz uyum gösteren öğrenme eğrisi Şekil 15'te gösterilmiştir.



Şekil 15. Yetersiz Uyum Gösteren Öğrenme Eğrisi

Uygun olmayan bir model, eğitim kaybının, çizimin sonunda düştüğü durumlarda belirlenmektedir. Modelin daha fazla öğrenme potansiyeline sahip olduğunu ve eğitim sürecinin erken bir aşamada durdurulduğunu göstermektedir. Bu durumun grafiği Şekil 16'da gösterilmiştir.



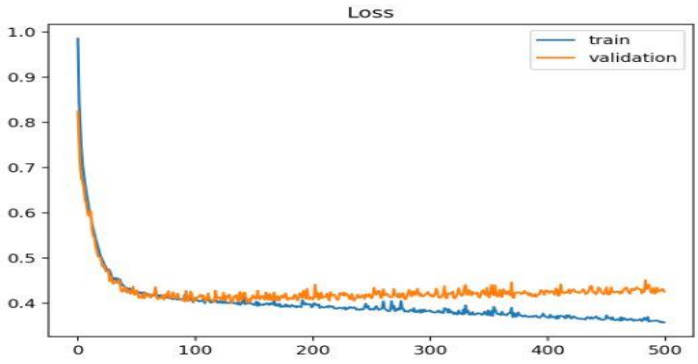
Şekil 16. Daha Fazla Eğitim Gerektiğini Gösteren Öğrenme Eğrisi

Aşırı uydurma, modelin eğitim verisindeki özellikleri aşırı derecede öğrendiği durumlarda ortaya çıkmaktadır. Daha esnek bir model uydurmak, daha fazla sayıda parametrenin tahmin edilmesini gerektirmektedir. Bu, daha karmaşık modellerde verilerin aşırı uydurulması olarak bilinen bir fenomene yol açmaktadır; buda esasen hataları veya gürültüyü çok yakından takip ettikleri anlamına gelmektedir (James, Witten, Hastie, ve Tibshirani, 2013). Aşırı uydurma probleminin temel sorunu, modelin eğitim verisine çok odaklandığıyla veya özelleştiğiyle ilgilidir, bu durum nedeniyle yeni ve farklı veri setlerine genelleme yapma yeteneği azalmaktadır. Yani, model eğitim verisindeki örüntülere aşırı adapte olduğunda veya onlara odaklandığında, farklı veri setlerine uyum

sağlama yeteneği azalmaktadır. Bu durum genellikle genelleme hatasının artmasına ve modelin yeni veri setlerinde daha kötü performans sergilemesine neden olmaktadır. Genelleme hatasındaki bu artış, modelin doğrulama veri kümesindeki performansı ile ölçülebilmektedir. Bu, verilerin aşırı uydurulmasının bir örneğidir ve olmaması gereken bir durumdur. Elde edilen uyum, orijinal eğitim veri setinin parçası olmayan yeni gözlemlere verilen yanıtın doğru tahminlerini vermemektedir. Aşırı uydurma genellikle, modelin problemin çözümü için fazladan kapasiteye sahip olması ve bundan dolayı çok fazla esneklik göstermesi durumunda ortaya çıkmaktadır. Model, eğitim sürecinde veri setine aşırı uyarlandığında veya çok uzun süre eğitildiğinde aşırı uydurma ortaya çıkmaktadır. Bu durum, modelin eğitim verilerindeki gürültüyü veya rastgele örüntüleri de öğrenmesine neden olabilmekte, bu da genelleme yeteneğini azaltabilmektedir. Aşırı uydurma, modelin aşırı karmaşık olması veya sınırlı eğitim verisine sahip olması gibi durumlarda ortaya çıkabilmektedir. Aşağıdaki durumlarda öğrenme eğrileri aşırı uydurmayı gösterir:

- Eğitim kaybı grafiği, deneyimle birlikte azalmaya devam ediyor.
- Doğrulama kaybı grafiği bir noktaya düşer ve tekrar artmaya başlar.

Doğrulama kaybındaki bükülme noktası, eğitimin durdurulabileceği nokta olabilir, çünkü bu noktadan sonraki deneyim aşırı uyumun dinamiklerini gösterir. Şekil 17'de grafiği gösterilmiştir.



Şekil 17. Aşırı Uydurma Gösteren Öğrenme Eğrisi

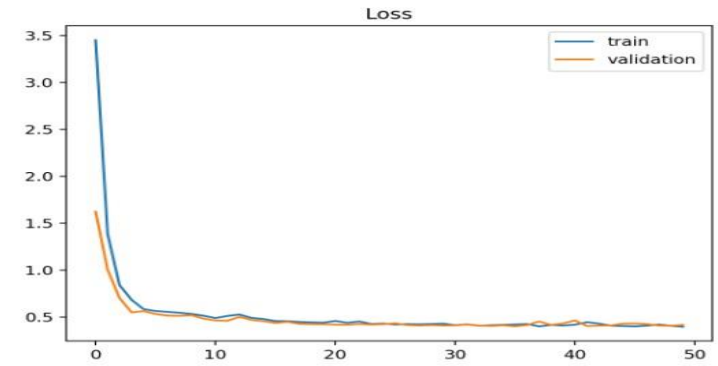
İyi bir uyum, öğrenme algoritmasındaki önemli amaçlardan biridir ve aşırı uydurma ile yetersiz uyum modelleri arasındaki bir durumdur. İyi bir uyum, eğitim ve doğrulama

kaybı arasındaki farkın minimum olduğu bir kararlılık noktasına ulaşmayı içermektedir. Şekil 18'de grafiği gösterilmiştir. Modelin kaybı, genellikle eğitim veri setinde doğrulama veri setine kıyasla daha düşük olmaktadır. Eğitim ve doğrulama kaybı öğrenme eğrileri arasındaki boşluğa genelleme boşluğu denilmektedir ve modelin öğrendiği özelliklerin daha genel ve taşınabilir olma ihtimalini yansıtarak aşırı uydurmadan kaçınılmasına yardımcı olmaktadır.

Öğrenme eğrileri grafiği aşağıdaki durumlarda iyi bir uyum göstermektedir:

- Antrenman kaybı grafiği bir denge noktasına düşer.
- Doğrulama kaybı grafiği, bir kararlılık noktasına düşer ve eğitim kaybıyla küçük bir boşluğa sahiptir.

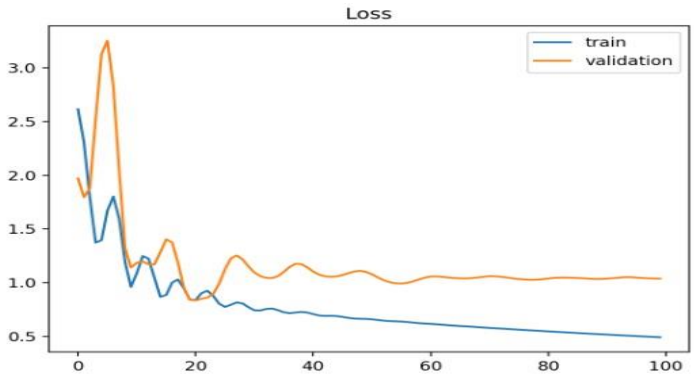
İyi bir uyum için sürekli eğitim, muhtemelen bir aşırı uyum ile sonuçlanacaktır.



Şekil 18. İyi Uyum Gösteren Öğrenme Eğrisi

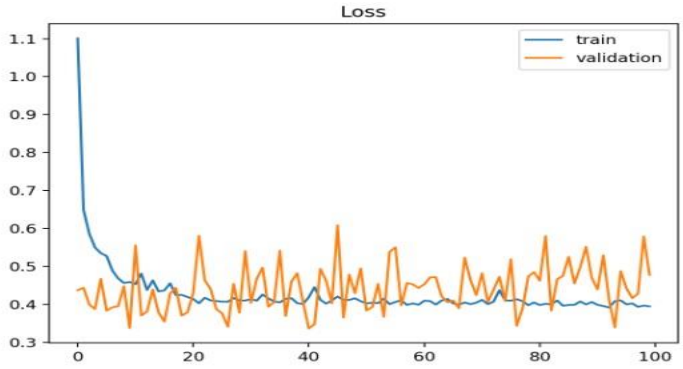
Temsili olmayan bir eğitim veri seti, modelin sorunu çözmek için yeterli bilgi sağlayamadığı anlamına gelmektedir. Bu durum, eğitim veri setinin, değerlendirme için kullanılan doğrulama veri setine kıyasla yetersiz olması veya az bir örneğe sahip olması durumlarında gözlemlenebilmektedir. Model için gelişme gösteren bir eğitim kaybı eğrisi ve aynı şekilde gelişme gösteren bir doğrulama kaybı eğrisi gözlemlenebilir. Fakat eğitim kaybı hızla düşerken, doğrulama kaybı yeterince düşmez ve iki eğri arasında belirgin bir açıklık ortaya çıkar. Şekil 19'daki grafikte açıklık gözükmektedir. Bu boşluk, modelin eğitim veri setinde iyi performans sergilemesine rağmen, doğrulama veri setinde beklenen performansı gösterememesinden kaynaklanmaktadır. Bu durum genellikle

modelin öğrenme sürecinde genelleme yapma yeteneğinin zayıf olduğunu ve yeni veri setlerinde iyi sonuçlar elde edemeyeceğini göstermektedir.



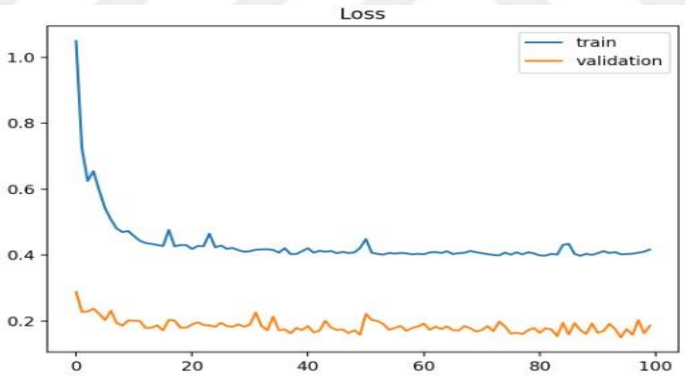
Şekil 19. Doğrulama Veri Kümesine Göre Çok Küçük Olabilecek Bir Eğitim Veri Kümesini Gösteren Öğrenme Eğrisi

Temsili olmayan bir doğrulama veri seti, modelin genelleme performansı için yeterli bilgi sağlanmadığı durumlarda gözlemlenebilmektedir. Doğrulama veri kümesi, eğitim veri kümesiyle kıyaslandığında ve çok az örneğe sahip olduğunda ortaya çıkabilmektedir. Eğitim veri kümesinde iyi bir uyum gösteren bir öğrenme eğrisi gözlemlenirken, doğrulama veri kümesinin öğrenme eğrisinde ise eğitim kaybı etrafında çok dalgalı hareketler gösteren bir doğrulama kaybı eğrisi görülebilmektedir. Şekil 20'deki grafikte gösterilmiştir. Model eğitim veri setinde doğru gibi görünse de doğrulama veri setinde beklenen performansı gösterememektedir. Bu durum genellikle modelin genelleme yapma yeteneğinin zayıf olduğunu ve yeni veri setlerinde iyi sonuçlar elde edemeyeceğini göstermektedir. Bu durum, modelin doğrulama veri seti için yeterince öğrenemediğini ve bu setteki örüntüleri tam olarak kavrayamadığını ifade etmektedir.



Şekil 20. Eğitim Veri Kümesine Göre Çok Küçük Olabilecek Bir Doğrulama Veri Kümesini Gösteren Öğrenme Eğrisi

Eğitim veri kümesinden daha kolay tahmin edilen doğrulama veri kümesini gösteren eğitme ve doğrulama öğrenme eğrilerinde ise eğitim kaybından daha düşük bir doğrulama kaybı da tanımlanabilmektedir. Şekil 21'de görüldüğü üzere; bu durum, modelin bu öğrenilen örüntüleri eğitim veri kümesinden daha kolay tahmin edebilecek bir yapıda olabileceğini göstermektedir.



Şekil 21. Eğitim Veri Kümesinden Daha Kolay Tahmin Edilen Doğrulama Veri Kümesini Gösteren Öğrenme Eğrisi

3.10 Data Augmentation

Algoritmanın performansını artırmanın yanı sıra derin öğrenme modellerinde büyük miktarda veri gereksinimlerini karşılamak için veri artırma, uygulanması gereken önemli bir araçtır (Murugan ve Durairaj, 2017). Eğitimde kullanılan görsellere rastgele dönüşümler yapılarak görüntülere yapay bir çeşitlilik kazandırılıp verilerin sayısı artırılabilmektedir. Veri artırma, mevcut veri setindeki örnekleri çeşitlendirmek ve yeni

örnekler oluşturmak amacıyla çeşitli dönüşümler ve manipülasyonlar uygulamayı içermektedir. Bu, modelin genelleme yeteneğini artırmakta ve aşırı uyumu azaltmaktadır. Veri artırma, basit bir ifadeyle, mevcut veri noktalarının varyasyonlarını oluşturarak kendisini büyütmek için seçilmiş bir veri kümesini kullanma sürecidir (Sarkar, Bali, ve Ghosh, 2018). Veri artırma, özellikle sınırlı sayıda örnek içeren veri setlerinde, modelin daha genel ve çeşitli özellikleri öğrenmesine yardımcı olmaktadır. Örneğin, görüntü sınıflandırma problemlerinde, bir görüntüyü yatay olarak çevirme, döndürme, kesme, renk düzenlemeleri gibi işlemlerle yeni eğitim örnekleri oluşturmak yaygın olarak kullanılan veri artırma yöntemlerindendir.

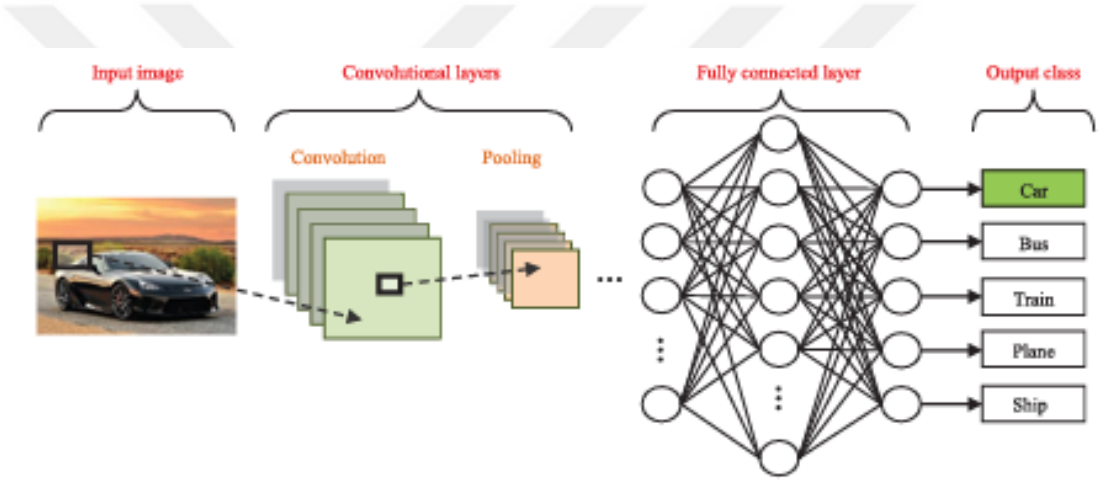
3.11 Uçtan – Uca Mimari

Uçtan uca mimari (End-to-End Learning), böl ve yönet mantığıyla tüm görüntü işleme hattını birkaç modüle veya alt adıma ayırarak, karmaşık ağ sorunlarını çözmenin temel bir fikridir. Farklı bir açıdan bakıldığında, derin öğrenme orijinal girdi ile istenen çıktı arasında doğrudan eşleme yapılmasını sağlamakta ve tüm süreç uçtan uca bir şekilde bir bütün olarak eğitilebilmektedir (Zuo, ve diğerleri, 2022).

3.12 Çok Sınıflı Görüntü Sınıflandırma

Görüntü sınıflandırma, görüntülerden bilgi çıkarma ve görüntüleri etiketleme veya kategorize etme işlemidir. Sınıflandırmalar ikili sınıflandırma ve çok sınıflı sınıflandırma olarak ikiye ayrılmaktadır. İkili sınıflandırmada çıktı sıfır veya bir ikili değerindedir, bir kedi görüntüsü verildiğinde görüntünün kedi olup olmadığı tespit edilmektedir. Çok sınıflı sınıflandırma da ise çıktılar çok sınıflı olmaktadır, bu nedenle bir görüntü verildiğinde ve 37 sınıf arasından köpek cinsinin tespit edilmesinin istendiği durumda kullanılmaktadır. Çok sınıflı sınıflandırma, ikiden fazla sınıf içeren bir sınıflandırma problemidir. Bu tür problemlerde, öğrenme algoritması, girdi verilerini inceleyerek her bir sınıf için olasılık veya güven skorları üretmektedir. Ardından, bu skorlara dayanarak girdinin hangi sınıfa ait olduğunu belirlemektedir. Önemli bir nokta, çok sınıflı sınıflandırmada, bir veri örneği yalnızca bir sınıfa ait olabilmekte ve genellikle en yüksek tahmin oranına sahip olan sınıf seçilmektedir. Yani, bir girdi, birden fazla sınıfa dahil olmaz; maksimum olasılık oranına sahip olan sınıf seçilmekte ve bu sınıf, girdinin tahmini sınıfı olarak belirlenmektedir. Etiketli örnek fotoğraflar kullanılarak, hedef

sınıfları (görüntülerde tanımlanacak nesneler) tespit etmek için bir model eğitilmektedir. Denetimli öğrenmenin en önemli alanlarından biri görüntü sınıflandırmasıdır. ESA, görüntü tanıma ve işlemede kullanılan ve verileri (pikselleri) işlemek için özel olarak tasarlanmış bir YSA türüdür. Şekil 22'de ESA görüntü sınıflandırma mimarisi gözükmektedir. Literatürde sınıflandırma problemlerinde birçok algoritma kullanılmaktadır. Bunlardan bazıları; k-en yakın komşu, karar ağaçları, naive bayes, destek vektör makineleri, gradyan artırma, YSA, adaBoost, lojistik regresyon sınıflandırıcı ve rasgele ormandır. Bu çalışmada sınıflandırma problemlerinde en çok tercih edilen yöntemlerden biri olan YSA kullanılmıştır.



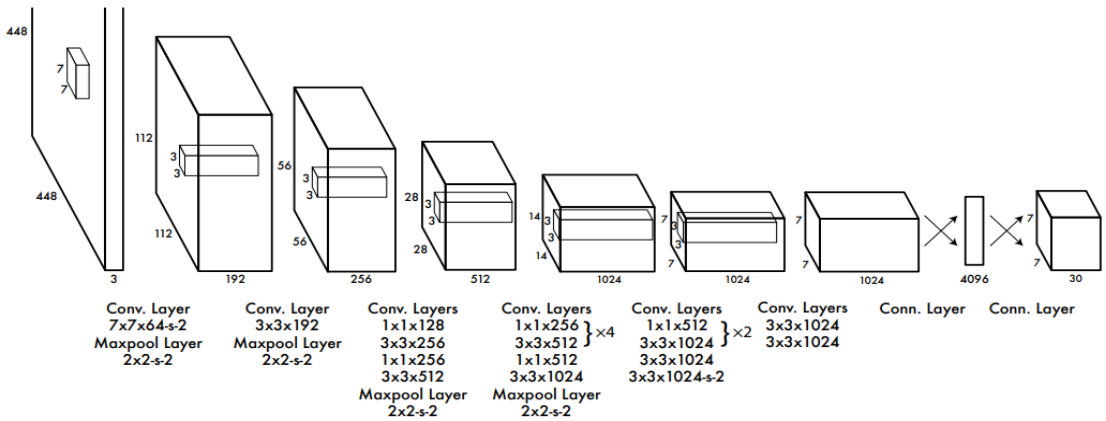
Şekil 22. ESA Görüntü Sınıflandırma Mimarisi (Rawat ve Wang, 2017)

3.13 Nesne Tespiti

Temel bilgisayarla görme problemlerinden biri olan nesne tespiti, görüntülerin ve videoların anlamsal olarak anlaşılması için değerli bilgiler sağlayabilmekte ve görüntü sınıflandırma, insan davranış analizi, yüz tanıma ve otonom algılama gibi birçok uygulama ile ilgilenmektedir (Zhao, Zheng, Xu, ve Wu, 2019). Nesne tespitinin amacı, bilgisayarla görme uygulamalarının ihtiyaç duyduğu en temel bilgi parçalarından birini sağlayan hesaplama modelleri ve teknikleri geliştirmektir. Nesne tespiti için en önemli iki ölçüt doğruluk (sınıflandırma doğruluğu ve lokalizasyon doğruluğu dahil) ve hızdır (Zou, Chen, Shi, Guo, ve Ye, 2023).

Nesne tespiti; nesne yerelleştirme (object localization) ve görüntü sınıflandırmanın bir birleşimidir. Giriş olarak görüntüler alınmakta ve görüntülerde bulunan nesnelerin konumları belirlenerek sınırlayıcı kutular (bounding box) oluşturulmaktadır. Tespit edilen nesneler özelliklerine (feature) göre farklı kategorilerde sınıflandırılmaktadır. Bu şekilde, görüntülerdeki nesnelerin konumlarıyla birlikte temsil ettikleri de belirlenmektedir. Nesne tanıma ile tespit edilen nesneyi sınırlandıran sınırlayıcı kutular her zaman dikdörtgen şeklindedir.

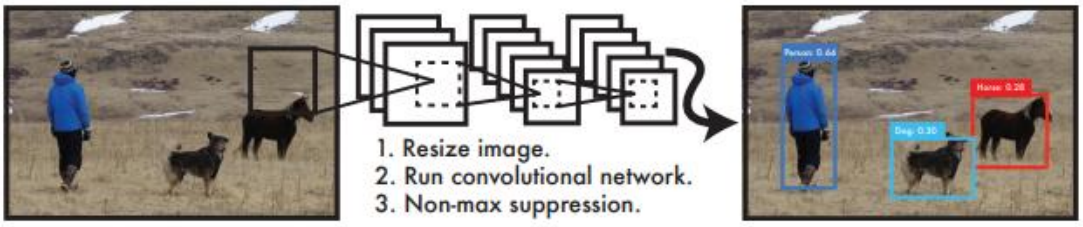
YOLO-Sadece Bir Kez Bakarsınız: Birleşik, Gerçek Zamanlı Nesne Algılama makalesi ile Joseph ve arkadaşları (2015) tarafından önerilmiştir. Derin öğrenme dönemindeki ilk tek aşamalı dedektördür (Redmon, Divvala, Girshick, ve Farhadi, 2015). YOLO; nesne tespiti için kullanılan bir derin öğrenme algoritmasıdır. YOLO, bir görüntü veya video içindeki nesneleri bulma yeteneğine sahip olmakla birlikte tespit ettiği nesnelerin hangi sınıflara ait olduğunu da tahmin edebilmektedir. Son derece hızlıdır ve iki aşamalı dedektörlerden tamamen farklı olarak tüm görüntüye tek bir sinir ağı uygulamaktadır. Bu ağ görüntüyü sınıflandırmakta ve her sınıf için sınırlayıcı kutuları ve olasılıkları aynı anda tahmin etmektedir (Zou, Chen, Shi, Guo, ve Ye, 2023). Görseldeki nesnelerin hangi sınıfa ait olduğunu tahmin etmekte böylelikle çoklu nesnelerin ve sınıfların aynı anda tespit edilebilmesini sağlamaktadır. Şekil 23'te YOLO mimarisi gösterilmiştir.



Şekil 23. YOLO Mimarisi (Redmon, Divvala, Girshick, ve Farhadi, 2015)

Çalışmada YOLO sürümlerinden yararlanan YOLOv8 modeli kullanılmıştır. YOLOv8, derin öğrenmede, özellikle ESA'lara dayalı açık kaynaklı bir nesne algılama

algoritmasıdır. Nesneleri gerçek zamanlı olarak algılamadaki hızları ve doğrulukları ile bilinen YOLOv8, nesne algılama algoritmaları ailesinin en son tanıtılan sürümüdür.



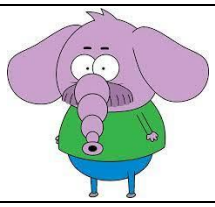
Şekil 24. YOLO Tespit Sistemi (Redmon, Divvala, Girshick, ve Farhadi, 2015)

YOLOv8, nesne algılama, görüntü sınıflandırma ve örnek bölümlendirme görevleri için kullanılabilecek en yeni, son teknoloji ürünü YOLO modelidir. YOLOv8, aynı zamanda etkili ve endüstriyi belirleyen YOLOv5 modelini de yaratan Ultralytics tarafından geliştirilmiştir. YOLOv8, temelde bir tek ağı geçen ve bu ağın çıktısını kullanarak nesne tespiti yapan bir mimariye sahiptir. Önceki versiyonlara kıyasla daha hafif ve daha hızlı çalışabilen bir yapıya sahiptir. YOLO, bir görüntüyü küçük bir örneklem altında bölerek, her bir bölge üzerinde nesne tespiti ve sınıflandırması yapmaktadır.

3.14 KralSakir Veri Setinin Özellikleri

Öncelikle üzerinde çalışılacak karakterler belirlenmiştir. Yasal olarak karakterlerin yaratıcısı Varol Yaşaroğlu ile iletişime geçilerek gerekli izinler alınmıştır (Yaşaroğlu, 2023). Kral Şakir çizgi filminin dört karakteri ile veri setimiz oluşturulmuştur. Google ve youtube vasıtasıyla Kral Şakir, Remzi, Necati ve Canan karakterlerinin manuel fotoğrafları gerekli düzenlemeler yapılarak toplanmıştır (Grafi2000, 2023). Video yoluyla toplanan görüntüler VLC Player (Video LAN Client) programı ile fotoğraf olarak klasörlere alınmış ve kaydedilen görseller tek tek karakterlere göre ayıklanmıştır. Farklı türde (jpeg, png, jfif) toplanan görüntüler dönüştürülerek JPG (Joint photographic experts group) dosyası olarak kaydedilmiştir. KralSakir veri seti sınıf başına farklı sayıda olup toplamda 16725 adet görüntü ile dört sınıflı ve boyutları değişen piksellerde (1280 x 720, 640 x 368, 256 x 144) olan renkli görüntüden oluşmaktadır. KralSakir veri setindeki sınıflar Tablo 2'de gösterilmiştir. Sınıflara ait görsellerin sayısı Tablo 3'te, grafiği ise Şekil 25'te verilmiştir.

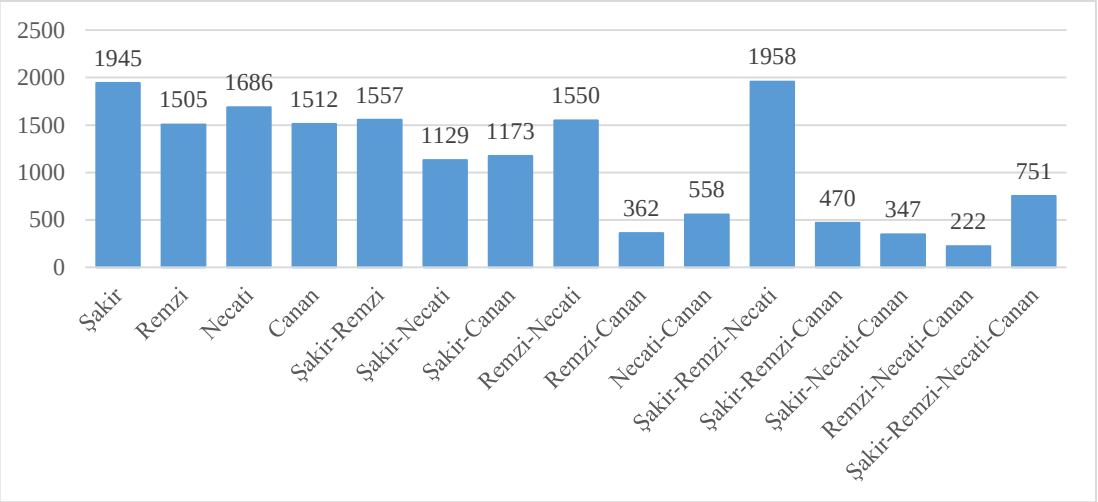
Tablo 2. KralSakir Veri Seti Sınıfları

Sınıflar	**Sınıf 1**	**Sınıf 2**	**Sınıf 3**	**Sınıf 4**
	Şakir	Remzi	Necati	Canan
Görseller				

Tablo 3. KralSakir Veri Seti Çizgi Karakter Sayıları

Çizgi Karakterlerin Sayısı	
Şakir	1945
Remzi	1505
Necati	1686
Canan	1512
Şakir-Remzi	1557
Şakir-Necati	1129
Şakir-Canan	1173
Remzi-Necati	1550
Remzi-Canan	362
Necati-Canan	558
Şakir-Remzi-Necati	1958
Şakir-Remzi-Canan	470
Şakir-Necati-Canan	347
Remzi-Necati-Canan	222
Şakir-Remzi-Necati-Canan	751
Toplam	**16725**

Toplam 16725 görüntüden, çok etiketli görüntü sınıflandırma modeli oluşturulurken %70'i eğitim verisi (11707 adet) ve %30'u test verisi (5017 adet) olacak şekilde bölünmüştür. 16725 adet görüntüden etiketlenen sınıfların sayı grafiği Şekil 25'te gösterilmiştir.



Şekil 25. Çizgi Karakterlerin Etiketlenen Görüntü Sayı Dağılım Grafiği

Tahmin için kullanılan 1200 görüntü, eğitim setinden farklı görüntülerden oluşmaktadır ve tüm eğitim kümesini yansıtacak şekilde seçilmiştir.

YOLOv8 ile oluşturulan nesne tespiti modelinde etiketleme yapılırken bazı görüntüler null olarak döndürülmüş bu yüzden veri setine dahil edilmemiştir. Toplam 14080 görüntüden %80'i (11215) eğitim, %10 'u (1407) test, %10'u da doğrulama (1401) olarak ayrılmıştır. Tahmin için ise veri tabanından ayrı 57 görsel kullanılmıştır. Şekil 26'da veri setinden rastgele görüntülere yer verilmiştir.



Şekil 26. Veri Setinden Rastgele Seçilen Görüntüler (Grafi2000, 2023)

Çizgi karakterler, süper kahramanlar, karikatür karakterleri yaratıcılarının hayal dünyalarına ya da hikayenin gidişatına göre çok farklı resmedilebilmektedirler. Farklı giyimler, renkler, aksesuarların yanı sıra yaşlılık halleri, bebeklik halleri vs. gibi karakterlerin çeşitli görüntüleri ve tarzları çizgi filmlerde kullanılmaktadır. İnsan beyni bu karışıklıkta resmedilen karakterin kim olduğunu belirli ipuçları ile anlayabilmektedir. Oluşturulan modellerin de bunu en yüksek doğrulukla öğrenebilmesi amaçlanmıştır. Şekil 27'de Kral Şakir, Şekil 28'de Remzi, Şekil 29'da Necati ve Şekil 30'da Canan karakterlerinin birkaç farklı versiyonu gösterilmiştir.



Şekil 27. Kral Şakir Karakteri Görselleri



Şekil 28. Remzi Karakteri Görselleri



Şekil 29. Necati Karakteri Görselleri



Şekil 30. Canan Karakteri Görselleri

Görsellerde çok geniş bir çerçeve sunulmuştur. Veri setinde düşük çözünürlüğe ve yüksek çözünürlüğe sahip görüntülerin yanı sıra bulanık ve net görüntülerde mevcuttur. Ayrıca karakterlerin yakın ya da uzakta oldukları görseller bulunmaktadır.

3.15 Çizgi Karakter Tanıma Yöntemi Oluşturulması

Çalışmada çizgi karakter tespiti için iki yöntem önerilmiştir. Modellerin oluşturulması aşamasında Intel Core i7 – 3750HF işlemcili, 16 GB RAM ve Windows 10 işletim sistemine sahip bir dizüstü bilgisayar kullanılmıştır. Bu çalışmanın başlangıç modelleri Pyhton 3.x ve Jupyter Notebook 6.x kullanılarak yapılmıştır. Modeller oluşturulduktan sonra Google Colab Pro'da çalışılmaya devam edilmiştir Colab'ı kullanarak, birçok alanda çeşitli projeler geliştirilmektedir. GPU ve TPU desteği, Python programlama dili desteği ve çeşitli kütüphanelerin kullanımı büyük bir avantaj sağlamaktadır. Colab'ın sunduğu Tesla K80, GPU RAM'i 15.0 GB, sistem RAM'i 51.0 GB ve 166.8 GB disk alanıyla eğitim aşamalarında denemeler yapılmıştır. Yöntemler için kullanılan kütüphaneler; TensorFlow, Keras, Numpy, Matplotlib, Pandas, tqdm, Sklearn, Seaborn, IPython, os, cv2, Glob, Ultralytics ve YOLO'dur.

3.15.1 Çok etiketli görüntü sınıflandırma yöntemi

Çalışmada çizgi karakter tespiti için görüntü sınıflandırma teknikleriyle oluşturulan bir görüntü sınıflandırma yöntemi önerilmiştir. Görüntü sınıflandırma, bir veri kümesindeki görüntüleri ilgili kategorilere veya etiketlere ayırmayı amaçlayan denetimli makine öğrenimi problemlerinden biridir. Görüntü sınıflandırmaları için yeterli sayıda eğitim örneği ve bunların temsil gücü kritik öneme sahiptir (Lu ve Weng, 2007). İkili sınıflandırmaların, çıktısı evet ya da hayır şeklindedir. Eğer cevabın, ikiden fazla seçenek arasından seçilerek verilmesi gerekiyorsa, bu duruma çok sınıflı sınıflandırma (multi-class classification) problemi adı verilmektedir. Eğer hedeflediğimiz sınıf, birden fazla sınıfa ait bir kümeden gelirse ve bu sınıfların birden fazlasını aynı anda belirleme ihtiyacı doğarsa, buna çok etiketli sınıflandırma (multi-label classification) denilmektedir. Tanımlayıcıya verilen bir görselde veri setimiz üzerinden örnek verilirse hem Kral Şakir hem Remzi hem de Canan karakteri aynı görsel üzerinde görülebilir, bu nedenle sadece bir etiketi seçmek diğerlerinin olmadığı anlamına gelmez. Bu tür problemlerde, çok etiketli sınıflandırıcılar en doğru çıktıyı sağlamaktadır. Günlük hayatta karşılaştığımız sınıflandırma problemlerinin çoğu, birden fazla etiketle sınıflandırma ihtiyacını içermektedir. Çok etiketli sınıflandırmanın hedefi, tek bir örneğe birden fazla ilgili etiketi atamaktır (Read ve Perez-Cruz, 2014). Ancak, mevcut bilinen algoritmaların çoğu

genellikle çok sınıflı sınıflandırma problemleri için tasarlanmıştır ve çok etiketli problemleri ele almakta sınırlıdır.

Bu çalışmada oluşturulan modelde uçtan-uca bir mimari hazırlanmış ve makine öğrenmesinin gücü ile bir derin öğrenme uygulaması önerilmiştir. Bir derin öğrenme uygulaması yaparken izlememiz gereken bazı temel adımlar bulunmaktadır. Bunlar mimariyi kodlamak, verileri hazırlamak, eğitim, performans sonuçlarını edinmek, sonuçları görselleştirmek, sonuçlara göre aksiyon almak şeklinde sıralanabilir.

3.15.1.1 Mimarinin kodlanması

Oluşturulan mimariye cartoonNet adı verilmiştir. Öncelikle programda kullanılacak kütüphanelerden TensorFlow, Keras, Numpy, Matplotlib, Pandas, tqdm, Sklearn, Seaborn, IPython, Os ve cv2 kodlara dahil edilmiştir. Mimarinin temelleri Keras kütüphanesiyle oluşturulmuştur. TensorFlow kütüphanesi ile de optimizasyon algoritması çağırılmıştır. Mimari ilk etapta konvolisyon katmanı, normalizasyon katmanı ve havuzlama katmanlarından oluşturulmuştur. Bunlar ESA mimarisi örnek alınarak uygulanmıştır. Kodlamada görüntü sınıflandırması yapıldığından Keras kütüphanesinin bize önermiş olduğu ve kullanım kolaylığı sağlayan ESA katmanları kullanılmıştır. Geliştirilen mimari bir Python fonksiyonu olarak, ESA modeli oluşturan bir Keras modelini tanımlamaktadır. Filtre boyutları değişen adetlerde tercih edilerek sonrasında veriler flatten() metodu ile düzleştirilmiş ve yoğun katmana sokulmuştur. Bu kısımda tam bağlantılı bir yapı bulunmaktadır. Bu görüntülerde temelde dört karakter tespit edilmek istenmiştir. Yani oluşturulan mimari çıktı olarak dört karakteri ve bunların olasılıklarını bulmaktadır. Derin öğrenme uygulamalarında öğrenme işlemi temelde bir optimizasyon problemidir. Doğrusal olmayan problemlerin çözümünde çeşitli optimizasyon yöntemleri kullanılarak optimum değer bulunmaya çalışılır. Derin öğrenme uygulamalarında yaygın olarak SGD, Adagrad, Adadelta, Adam, Adamax gibi optimizasyon algoritmaları kullanılmaktadır. Bu algoritmalar arasında başarım ve hız bakımından farklılıklar bulunmaktadır. Bu çalışmada edinilen tecrübelerden ve daha önce yapılmış çalışmalar incelendiğinde alınan sonuçlardan en iyi optimizasyon algoritmasının Adam olduğu kanaatine varılmıştır ve çalışmada kullanımına karar verilmiştir. Mimari bir metot şeklinde yazılmıştır. Sonrasında modele vereceğimiz resimlerin boyutları input-shape() ile belirtilmiştir. Modelimizi eğitmeden önce ilk olarak onu yapılandırmamız

gerekmektedir ve bu yapılandırma işlemi kayıp fonksiyonunu, optimize edicileri ve tahmin ölçümlerini tanımlayan model.compile() tarafından yapılmıştır. Oluşturulan yöntemin kodları ekte sunulmuştur.

3.15.1.2 Veri setinin modele tanıtılması

Dört sınıfa ait KralSakir veri setimizdeki görseller 1'den başlayarak isimlendirilip file_name adlı tek bir dosyada veri seti olarak toplanmıştır. Aynı dosyada toplanan görüntüler excel dosyasına manuel bir şekilde Şekil 31'deki gibi kaydedilmiştir. Bu dosya modelde kullanılmak üzere CSV dosyasına dönüştürülmüştür.

CSV, daha iyi okunulabilmesi, daha kolay anlaşılması ve daha hızlı işlenmesi için tablo halindeki verileri saklayan bir metin dosyası türüdür. CSV dosyaları bir JSON (JavaScript object notation) dosyasından dönüştürülebilir, Python veya Java kullanılarak oluşturulabilir. CSV virgülle ayrılmış değerler anlamına gelmektedir. Bu dosyanın içeriğinde aralarına virgül konularak birbirinden ayrılmış metin veya rakamlar yer almaktadır. CSV uzantılı dosyalar not defteri ile açılabilmesine rağmen bu dosyaları doğru anlaşılır şekilde görüntülemek için Excel ya da Access ofis programlarında veya türevlerinde açmak gerekmektedir. Pandas'ta Dataframe olarak küçük bir script yazılarak veri setimiz kaydedilmiştir. Dataframe, iki boyutlu veriler ve karşılık gelen etiketleri içeren bir yapıdır. Veri bilimi, makine öğrenmesi, bilimsel hesaplar ve diğer birçok verinin yoğun olduğu alanlarda kullanılmaktadır. Dataframe, SQL tabloları veya Excel gibi tablolara benzetilebilir. Fakat Numpy Kütüphanesi de kullanıldığında hesaplama yapmak için Excel'den çok daha hızlıdır, güçlüdür ve kullanımı da daha kolaydır. Pandas Dataframe; pandasın çok boyutlu veri yapılarıdır. Dataframe yapıları veri analizi işlemlerinde oldukça faydalı olmaktadırlar. CSV dosyasının ilk satırı başlıktır. Dosyadaki sütun adları olarak üstte belirlenen başlıklar alanların/özelliklerin adlarını içermektedir. Başlıktan sonra dosyanın her satırı bir gözlem/kayıttır. Bir kaydın değerleri virgüllerle ayrılır. Veri setimizi kodlamamıza dahil etmek için Pandas Kütüphanesi kullanılmıştır. Pandas kütüphanesini kullanılarak belirtilen CSV dosyası okunur ve bir Dataframe'e yüklenir. Görsellerin adresi belirtilmiştir. Pandas kütüphanesinden CSV okuma metodu ile veri seti kontrol satırlarını içeren bir kodla Şekil 32'deki CSV dosyası çağırılmıştır.

filename	name	kral_sakir	remzi	fil_necati	canan
1	['kral_sakir']	1	0	0	0
2	['kral_sakir']	1	0	0	0
3	['kral_sakir']	1	0	0	0
4	['kral_sakir']	1	0	0	0
5	['kral_sakir']	1	0	0	0
6	['kral_sakir']	1	0	0	0
7	['kral_sakir']	1	0	0	0
8	['kral_sakir']	1	0	0	0
9	['kral_sakir']	1	0	0	0
10	['kral_sakir']	1	0	0	0
11	['kral_sakir']	1	0	0	0
12	['kral_sakir']	1	0	0	0
13	['kral_sakir']	1	0	0	0
14	['kral_sakir']	1	0	0	0
15	['kral_sakir']	1	0	0	0
16	['kral_sakir']	1	0	0	0
17	['kral_sakir']	1	0	0	0
18	['kral_sakir']	1	0	0	0
19	['kral_sakir']	1	0	0	0
20	['kral_sakir']	1	0	0	0
21	['kral_sakir']	1	0	0	0
22	['kral_sakir']	1	0	0	0
23	['kral_sakir']	1	0	0	0
24	['kral_sakir']	1	0	0	0
25	['kral_sakir']	1	0	0	0

Şekil 31. Oluşturulan CSV Dosyası

CSV dosyalarından etiket ve değerler okunmaktadır ve o değerler resimlerle eşleştirilerek modelde veri seti olarak kullanılmaktadır. Kontrol amaçlı sütunlar ile ilk 10 satır ve son 10 satır programda yazdırılmıştır. Şekil 32'de çıktısı verilmiştir.

```
Index(['filename', 'name', 'kral_sakir', 'remzi', 'fil_necati', 'canan'], dtype='object')
   filename            name  kral_sakir  remzi  fil_necati  canan
0         1  ['kral_sakir']           1      0           0      0
1         2  ['kral_sakir']           1      0           0      0
2         3  ['kral_sakir']           1      0           0      0
3         4  ['kral_sakir']           1      0           0      0
4         5  ['kral_sakir']           1      0           0      0
5         6  ['kral_sakir']           1      0           0      0
6         7  ['kral_sakir']           1      0           0      0
7         8  ['kral_sakir']           1      0           0      0
8         9  ['kral_sakir']           1      0           0      0
9        10  ['kral_sakir']           1      0           0      0
       filename                                          name  kral_sakir  \
16715     16716  ['kral_sakir','remzi','fil_necati','canan']           1
16716     16717  ['kral_sakir','remzi','fil_necati','canan']           1
16717     16718  ['kral_sakir','remzi','fil_necati','canan']           1
16718     16719  ['kral_sakir','remzi','fil_necati','canan']           1
16719     16720  ['kral_sakir','remzi','fil_necati','canan']           1
16720     16721  ['kral_sakir','remzi','fil_necati','canan']           1
16721     16722  ['kral_sakir','remzi','fil_necati','canan']           1
16722     16723  ['kral_sakir','remzi','fil_necati','canan']           1
16723     16724  ['kral_sakir','remzi','fil_necati','canan']           1
16724     16725  ['kral_sakir','remzi','fil_necati','canan']           1

       remzi  fil_necati  canan
16715      1           1      1
16716      1           1      1
16717      1           1      1
16718      1           1      1
16719      1           1      1
16720      1           1      1
16721      1           1      1
16722      1           1      1
16723      1           1      1
16724      1           1      1
```

Şekil 32. Veri Seti Kontrol Satırları

Keras.prosecessing kütüphanesi ise görselleri okuma, yükleme gibi fonksiyonlar sağlanmaktadır. Makine öğrenmesi modellerinin sistematiği genelde şöyledir: Veri eğitim ve test setlerine ayrılmaktadır. Makine öğrenmesi modelleri verinin bir kısmıyla eğitilmekte ve modelin hiç görmediği test verisi üzerinde isabetli tahminler yapması beklenmektedir. Diğer bir deyişle, modelin genelleme yeteneğinin yüksek olması demek sadece oluşturulan veri seti üzerinde iyi sonuçlar vermesini beklemek değil aynı zamanda hiç görmediği başka veri setleri üzerinde de iyi sonuçlar vermesini beklemek anlamına gelmektedir.

Sklearn kütüphanesi kullanılarak eğitim ve test verilerinin düzenlendiği train_test_split() metodu programa tanıtılmıştır. Veriler normalize edildikten sonra bir x matrisinde depolanmıştır ve oluşturulan mimarinin eğitiminde kullanılmak üzere %70'i train ve %30'u test olacak şekilde ikiye ayrılmıştır.

3.15.1.3 Eğitim

Sklearn veya scikit-learn, veri bilimi topluluğundaki en etkili makine öğrenimi paketidir. Python programlama dilinde yazılan sklearn, makine öğrenmesi modellerini uygulamaya yönelik veri işleme için oldukça etkili ve kullanımı kolay araçlar sağlamaktadır. Eğitim için kullanılacak veriler modelin içerisine yerleştirilerek veri seti, eğitim ve test alt kümelerine ayrılmıştır. Eğitim ve test için %70'e %30 oranında bir bölümleme yapılmıştır. Çeşitli eğitim epok değerleri denenmiştir. Alınan sonuçlar ve grafikler bulgular kısmında değerlendirilmiştir.

Eğitim süreci boyunca, model çeşitli epok sayılarında eğitilmekte ve belirli aralıklarla doğrulama verisi üzerinde performansı değerlendirilmektedir. Epok sayısı, eğitim sürecinin uzunluğunu belirlemekte ve modelin daha fazla veriyi görmesi ve öğrenmesi için bir fırsat sunmaktadır. Daha yüksek epok sayıları, modelin daha fazla eğitim yapmasına ve veriye daha fazla adapte olmasına olanak sağlayabilmektedir. Ancak, epok sayısını çok yüksek tutmak, aşırı öğrenmeye yol açabilmekte, yani modelin eğitim veri setine çok fazla uyum sağlayarak genelleme yeteneğini kaybetmesine neden olabilmektedir. Genellikle, epok sayısı eğitim veri setinin boyutuna ve karmaşıklığına bağlı olarak belirlenir. Daha büyük ve karmaşık veri setleri genellikle daha fazla epok sayısı gerektirirken, daha küçük veri setleri için daha az epok sayısı yeterli olabilmektedir. Epok sayısını belirlerken, eğitim süresi ve hesaplama kaynakları gibi faktörlerin de göz önünde bulundurulması önemli olmaktadır. Farklı eğitim epok sayıları denenmiş ve grafiklerden takip edilmiştir. Ezberleme olması yönündeki eğilimin engellemesi için modelde düzenlileştirme yapılmasına karar verilmiştir. Modelde L1 düzenlileştirilmesi uygulanarak 200 epokta eğitim yapılmıştır.

3.15.1.4 Sonuçların görselleştirilmesi

Oluşturulan modelde history() nesnesini kullanarak, eğitim süreci boyunca elde edilen metrik değerlerini bir Pandas Dataframe olarak dönüştürüp görüntüleyen Python kodu yazılmıştır. Görselleştirilen sonuçlar için plot() yöntemi kullanılarak bulgular kısmında ele alınmıştır.

3.15.1.5 Sınıfların tahmin edilmesi

Sınıflandırma problemlerinde, predict() işlevi bir veya daha fazla veri örneğinin bir listesini veya dizisini almakta ve eğitilmiş bir modelde yeni veriler üzerinde tahminler yapmak için kullanılmaktadır. Bu fonksiyon, modelin çıktıları olan tahminleri döndürmektedir. Genellikle çoklu sınıflandırma problemlerinde bir olasılık dağılımı döndürmektedir. Yani, her sınıf için bir olasılık değeri elde edilmektedir. Bu çıktıyı yorumlamak, en yüksek olasılığa sahip sınıfı belirlemek veya bir güven eşiği (threshold) uygulamak gibi işlemleri içermektedir. Sonuçların yüzdesel olasılıkları çıktı ekranına sınıfları ile birlikte yazdırılmış ve bulgular kısmında bahsedilmiştir.

3.15.2 YOLOv8 ile nesne tespiti yöntemi

YOLOv8 önceden yayınlanmış tüm YOLO nesne algılama modellerinin en son sürümüdür ve daha yüksek doğruluğa ve performansa sahiptir. Bu nedenle oluşturulan KralSakir veri setinin nesne tespiti çalışmasında tercih edilmiştir. YOLOv8'in mimarisi, YOLO algoritmalarının önceki sürümlerine dayanmaktadır. Gerçek zamanlı nesne algılama ve görüntü bölümleme modeli Ultralytics'in YOLOv8'in en yeni versiyonunu sunmaktadır. Bu sürüm, derin öğrenme ve bilgisayarlı görme alanındaki en son gelişmeleri kullanarak olağanüstü hız ve doğruluk sağlamaktadır. Verimli tasarımı çok çeşitli uygulamalara hitap etmekte ve kullanımı kolay Ultralytics Python paketindeki uygulaması sayesinde uç cihazlardan bulut uygulama programlarına kadar çeşitli donanım platformlarına sorunsuz bir şekilde ayarlanabilmektedir (Chaurasia ve Jocher, 2023). Nesne tespitine yönelik temel değerlendirme ölçütleri doğruluk, sınıflandırma, yerelleştirme kesinliği ve hızlılıktır. Nesne lokalizasyonu ile yapılan sınıflandırmalarda sınırlayıcı kutu çizilerek veya görüntünün görüntü içindeki konumu belirlenerek hem sınıfını hem de nesnenin görüntünün neresinde olduğunu bulmaktadır.

3.15.2.1 Mimarinin kodlanması

YOLOv8 GitHub'taki kaynağından veya pip komutu yoluyla iki şekilde kurulabilmektedir. Mimari oluşturulurken YOLOv8 pip'ten modele dahil edilmiştir. COCO (Common Objects in Context) nesne algılama modellerinin değerlendirilmesine yönelik endüstri standardı bir kıyaslamadır. COCO'daki modelleri karşılaştırırken

çıkarım hızı için mAP değerine ve FPS (frame per second) ölçümüne bakılmaktadır. Modeller benzer çıkarım hızlarında karşılaştırılarak seçilmektedir.

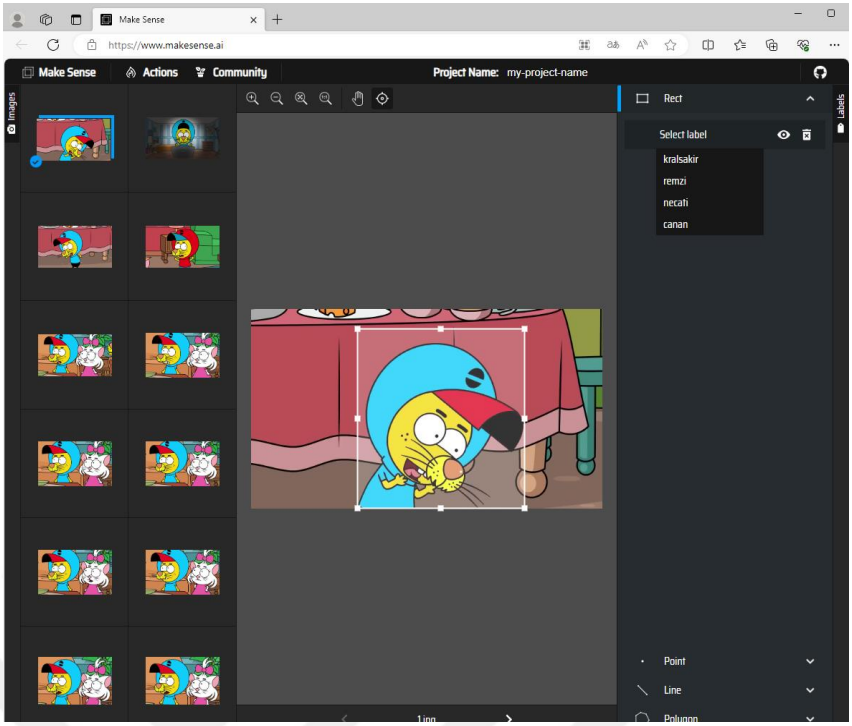
Model	Filenames	Task	Inference	Validation	Training	Export
YOLOv8	`yolov8n.pt` `yolov8s.pt` `yolov8m.pt` `yolov8l.pt` `yolov8x.pt`	Detection	✅	✅	✅	✅
YOLOv8-seg	`yolov8n-seg.pt` `yolov8s-seg.pt` `yolov8m-seg.pt` `yolov8l-seg.pt` `yolov8x-seg.pt`	Instance Segmentation	✅	✅	✅	✅
YOLOv8-pose	`yolov8n-pose.pt` `yolov8s-pose.pt` `yolov8m-pose.pt` `yolov8l-pose.pt` `yolov8x-pose.pt` `yolov8x-pose-p6.pt`	Pose/Keypoints	✅	✅	✅	✅
YOLOv8-cls	`yolov8n-cls.pt` `yolov8s-cls.pt` `yolov8m-cls.pt` `yolov8l-cls.pt` `yolov8x-cls.pt`	Classification	✅	✅	✅	✅

Şekil 33. YOLOv8 Görev Dosyaları

Şekil 33'te görüldüğü üzere önceden eğitilmiş dosyalardan çalışmaya uygun olan yolov8n.pt ve yolov8l.pt dosyaları programa eklenerek transfer öğrenme yapılmıştır. Daha önceden eğitilmiş dosyalar programımıza dahil edilerek oluşturulan model eğitilmiştir. YOLOv8'nin mimarisi, önceden eğitilmiş bir ESA içermektedir. Bu ağ, görüntülerden özellikler çıkarmakta ve ardından bu özellikleri kullanarak nesne tespiti yapmaktadır. YOLOv8, daha önceki versiyonlara kıyasla daha fazla evrişimli katman içermektedir ve özellikle "l" eki ile belirtilen büyük modelde daha fazla parametre ve katman bulunmaktadır. Önceden eğitilmiş modelleri kullanmanın en önemli avantajı kendi modelimizi daha güçlü yapmaktır. Mimariyi kodlamaya başlarken öncelikle kullanılacak kütüphaneler programa dahil edilmiştir. Ultralytics kütüphanesi indirilmiştir ve Os Kütüphanesi programa dahil edilmiştir. Böylece kullanılacak YOLOv8 mimarisi programa tanıtılmıştır.

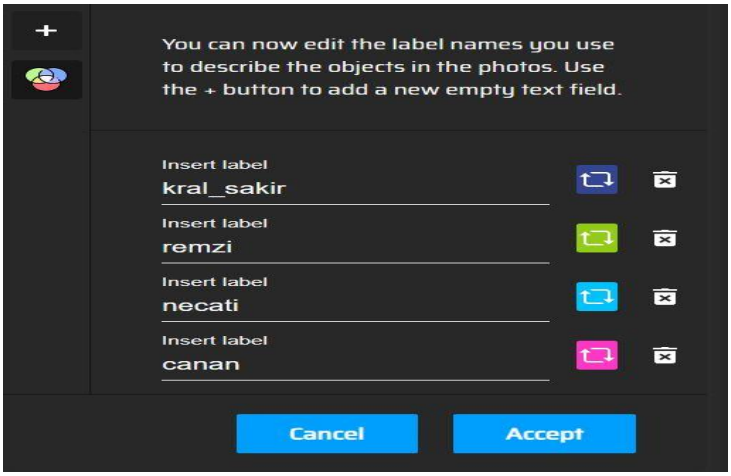
3.15.2.2 Veri setinin modele tanıtılması

YOLOv8 teknolojisini kullanırken, modele hassas veri etiketlemesi sağlamak için bir etiketleyicinin yardımı gereklidir. Oluşturulan KralSakir veri setindeki karakterlerin tespiti için veri setindeki tüm görseller makesense.ai sitesi yardımıyla tek tek etiketlenmiştir. Tüm görsellerin manuel olarak etiketlenmesi zor ve zaman alan bir süreçtir.



Şekil 34. Makesense.ai Sitesi Proje Ekranı

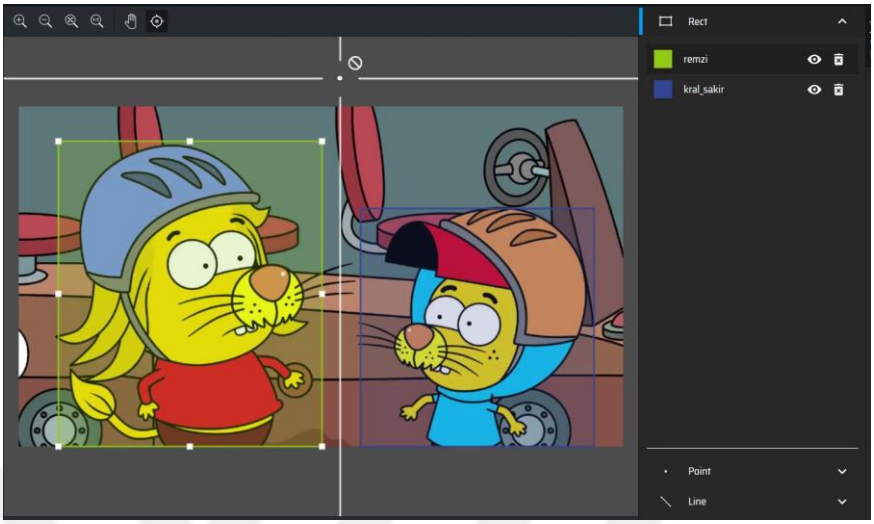
Etiketleme işleminde birkaç aşama mevcuttur. Şekil 34'teki gibi makesense.ai sitesinden yapılacak sınıflama türü seçilerek veri seti siteye yüklenmiştir.



Şekil 35. Makesense.ai Sitesi Etiket Oluşturma İşlemi

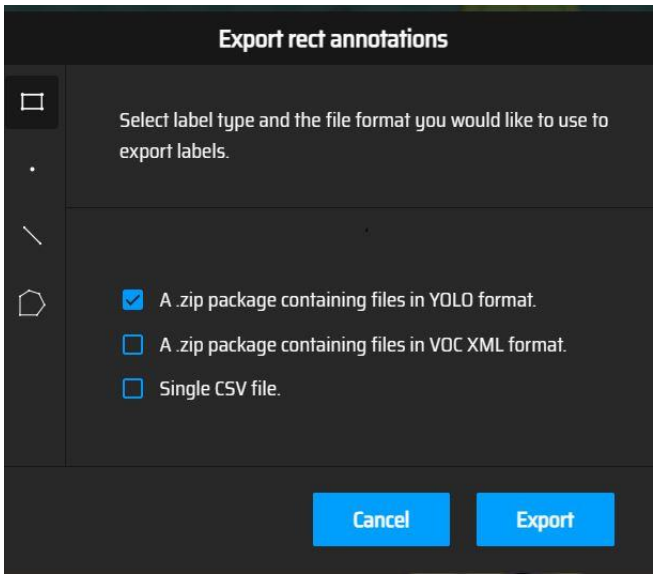
Yükleme işi tamamlandıktan sonra Şekil 35'teki gibi sınıflar belirlenmiş ve Şekil 36'daki gibi çizgi karakterler kutucuklara alınarak yerleri belirtilmiştir. Bu etiketleme işlemi 16725 adet görsel için tek tek yapılmıştır. Bazı görsellerin yüklenirken hata vermesinden

ve etiket değerini null döndürmesinden dolayı toplam 14080 adet görsel için etiketleme bilgisi alınabilmiştir. Bu nedenden dolayı YOLOv8 modelinde kullanılan veri setinde bir miktar düşüş yaşanmıştır.

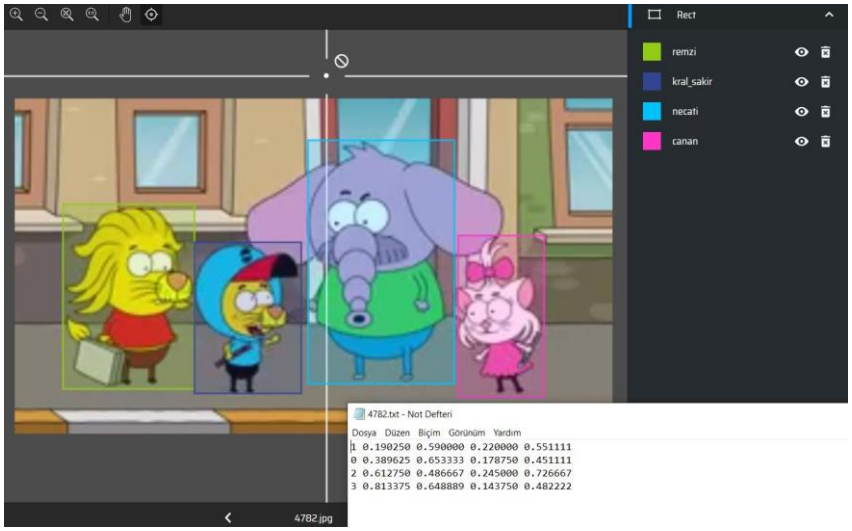


Şekil 36. Makesense.ai Sitesi Etiketleme İşlemi

Görsellerin tamamı etiketlendikten sonra etiketleme bilgilerini içeren dosya Şekil 37'deki seçimle birlikte YOLO formatında dışarı aktarılmıştır. Dışarı aktarılan dosyada her bir görsel için tek tek belirlenen konum (koordinat) bilgilerinin bulunduğu bir .txt belgesi elde edilmiştir. Şekil 38'de görseli gösterilmiştir.

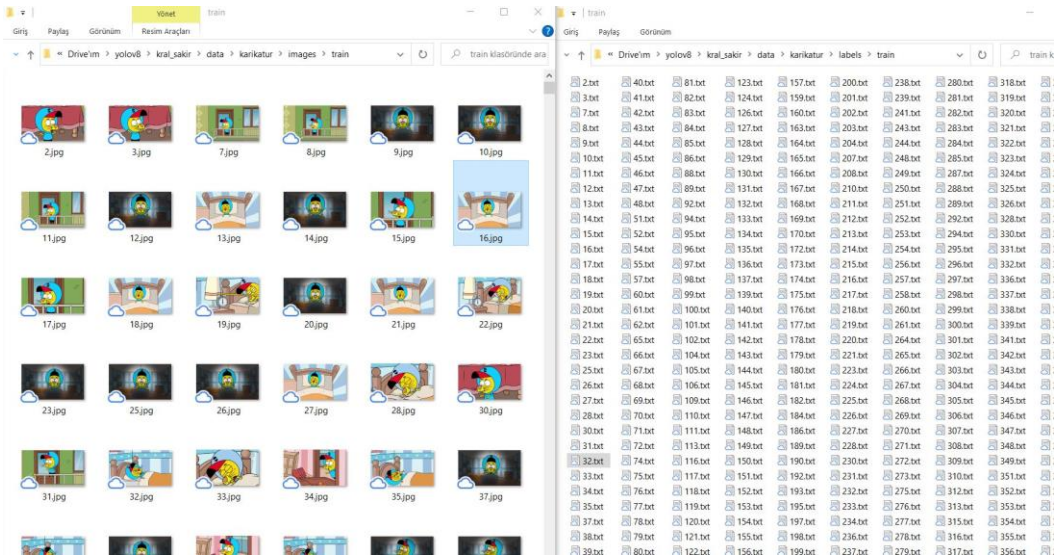


Şekil 37. Etiketlemesi Yapılan Görsellerin Dışarı Aktarımı



Şekil 38. Etiketleme Sonrası Sınıflarıyla Birlikte Oluşturulan .txt Belgesi

Görseller; images klasörüne train, val ve test klasörleri oluşturularak %80 eğitim, %10 test ve %10 doğrulama verisi olarak eklenmiştir. Şekil 38'te görüldüğü gibi etiketlemeden de elde edilen .txt dosyaları image klasöründeki görsellere uygun bir şekilde label klasörünün altında oluşturulan train, val ve test dosyalarına eklenmiştir. Şekil 39'da görseller ve .txt dosyaları gösterilmiştir. Eğitim için 11215, doğrulama için 1401, test için 1407 ve inference olarakta 57 adet görsel veri setimizden ayrılmıştır. Toplam 14.080 adet veri üzerinden model eğitilmiştir. Inference olarak oluşturulan dosyada model eğitildikten sonra çıktı olarak tahmin edeceği dosyalar bulunmaktadır.



Şekil 39. Dosyalara Uygun Şekilde Etiketlerin Eklenmesi

Veri setinin adreslerinin bulunduğu bir YAML (YAML Ain't Markup Language) uzantılı dosya düzenlenmiştir. İçinde train (train/images ve label yolu), val (val/images ve label yolu), test (test/images ve label yolu), sınıfların sayısı ve sınıfların adları bulunmaktadır. Bu dosya ile tüm veri setine ait adresler tanımlanmıştır. YAML dosyasının içeriği Şekil 40'da gösterilmiştir.

karakter.yaml ×

```yaml
# Dataset Path
path: /content/drive/MyDrive/yolov8/kral_sakir/data/karikatur

train: images/train # relative to path
val: images/val # relative to path
test: images/test # relative to path

# Class Names
names: [kral_sakir, remzi, necati, canan]
```

Şekil 40. YAML Dosyası

3.15.2.3 Eğitim

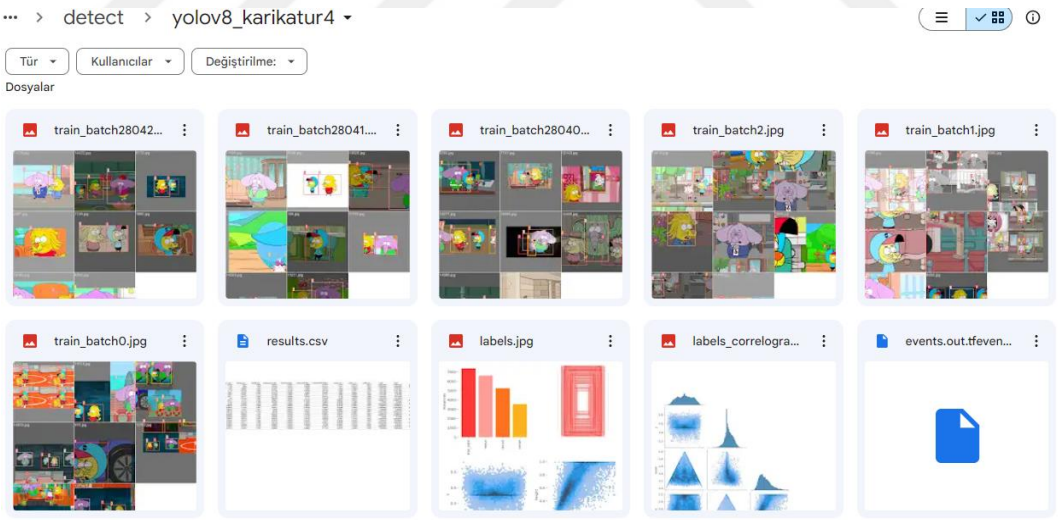
Bir derin öğrenme modelini eğitmek, doğru tahminler yapabilmesi için veri beslemeyi ve parametrelerini ayarlamayı içermektedir. Ultralytics YOLOv8'deki eğitim modu, modern donanım özelliklerinden tam olarak yararlanarak nesne algılama modellerinin etkili ve verimli eğitimi için tasarlanmıştır (Jocher, 2023).

YOLOv8 modeli, tahmin edilen sınırlayıcı kutular ile temel gerçek sınırlayıcı kutular arasındaki farkı en aza indirecek şekilde model parametrelerini optimize eden etiketli veri kümeleri üzerinde eğitilmektedir. Sınırlayıcı kutu koordinatları ve sınıf olasılığı yardımıyla yalnızca tespit edilen nesnenin hangi sınıfa ait olduğu değil, aynı zamanda nesne lokalizasyonunu da görüntülemektedir. Oluşturulacak sınıflama modeli için kullanılmasına karar verilen önceden eğitilmiş (pretrained) modellerden yolov8l.pt ve yolov8n.pt modeli Google Drive'a klonlanmıştır. YOLOv8 model dosyalarında "l" büyük (large) ve "n" ise küçük (numel) anlamlarına gelmektedir. Bu, modelin büyüklüğünü veya parametre sayısını ifade etmektedir. İki modelde deneme yapılmış ve veri setinin büyüklüğü ve parametre sayısının çokluğundan dolayı yolov8l.pt dosyasından transfer öğrenme yapılarak eğitim yapılmasına karar verilmiştir. İlk eğitimde epok sayısı 100

olarak belirlenmiş fakat 54 epokta YOLOv8 eğitimi durdurmuştur. İyileşmelerin olmadığı durumlarda otomatik bir durma söz konusudur. Alınan en iyi ağırlıklar ve son ağırlıklar kullanılarak eğitim tekrar edilmiştir. Birkaç denemeden sonra en iyi sonuçların 20 epokta alındığı gözlemlenmiş ve söz konusu epok sayısı kadar eğitim yapılmasına karar verilmiştir. Sonuçları bulgular kısmında sunulmuştur.

3.15.2.4 Sonuçların görselleştirilmesi

Tahminleri görselleştirmek için plot() yöntemi kullanılmaktadır. Nesnede bulunan tüm tahmin türleri (kutular, maskeler, anahtar noktalar, olasılıklar vb.) daha sonra gösterilebilecek veya kaydedilebilecek bir numpy dizisine result olarak çizilmektedir (Jocher, 2023). Tıpkı train, val ve test klasörlerinde olduğu gibi yapı gereği sonuçların kaydedilebileceği bir run klasörü oluşturulmuştur. YOLOv8 ile nesne tespiti yapılırken eğitim sonucunda elde edilen ağırlıklar, doğruluk, kayıp değerleri vb. gibi sonuçlar oluşturulan run klasörünün içinde modelin kendi oluşturduğu detect alt klasörüne kaydedilmiştir. Şekil 41'de detect klasörünün içeriği gösterilmiştir. Modelden görsel olarak alınan tüm sonuçlar bulgular kısmında sunulmuştur.



Şekil 41. Detect Klasöründe Sonuçların Görselleştirilmesi

3.15.2.5 Sınıfların tahmin edilmesi

Ultralytics YOLOv8, çok çeşitli veri kaynakları üzerinde yüksek performanslı, gerçek zamanlı çıkarım için tasarlanmış, tahmin modu olarak bilinen güçlü bir özellik sunmaktadır. Ultralytics YOLO modelleri, çıkarım sırasında modele iletildiğinde ya bir

Python results() nesne listesi ya da bellek açısından verimli bir Python nesne oluşturucusu döndürmektedir. Tüm Ultralytics predict() çağrıları nesnelerin bir listesini döndürmektedir (Jocher, 2023). YOLOv8 tahmin için kullandığı predict() kodu ile inference içinde olan tüm görsellerin tahmin çıktılarını yüzdeleriyle birlikte vermektedir. Tahmin edilen görsellerin sonuçları bulgular kısmında sunulmuştur.

4. BULGULAR VE TARTIŞMA

Bu bölümde çalışmada oluşturulan modellerin mimarisi, eğitim kısımları ve değerlendirme sonuçlarının yanı sıra çalışmanın görsel çıktılarının performansları ile yorumlama kısımlarına yer verilmiştir.

4.1 Çizgi Karakter Tanıma İçin Önerilen Çok Sınıflı Görüntü Sınıflandırma Yöntemi

Önerilen ilk yöntem olan çok sınıflı görüntü sınıflandırma yönteminin mimarisinin oluşturulması, kodlanması ve metriklerin sonuçlarıyla ilgili bilgiler aşağıda başlıklar halinde sunulmuştur.

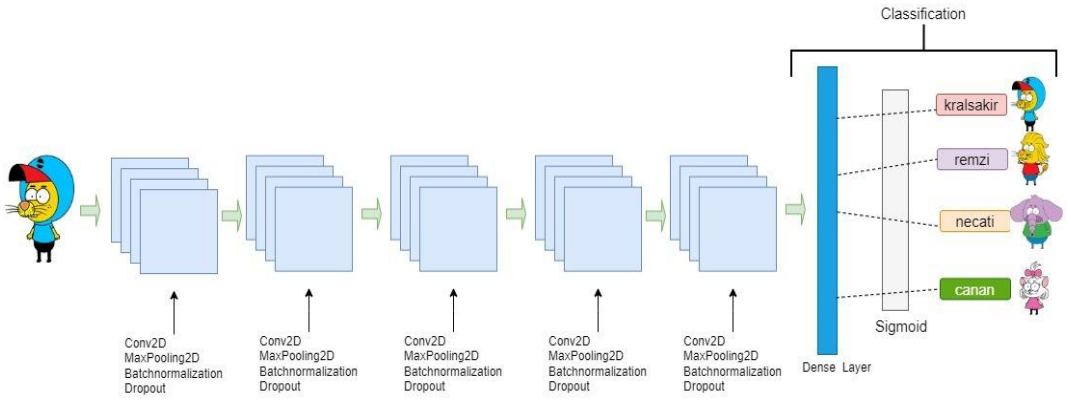
4.1.1 Yöntemin mimarisi

Fonksiyon şeklinde oluşturulan modelin katmanları ve özellikleri şöyledir:

- Giriş Katmanı (Conv2D); Filtre sayısı: 16, Filtre boyutu: (5, 5), Aktivasyon fonksiyonu: ReLU, Giriş şekli: input_shape(224, 224, 3) (Görüntülerin kaça kaç boyutta modele alınacağı belirlenmiştir).
- Batch Normalization Katmanı: Girişi normalize etmek için eklenmiştir.
- Max Pooling Katmanı: Havuzlama boyutu: (2, 2)
- Dropout Katmanı: Düğüm seyreltme oranı: 0.2, aşırı uydurmayı azaltmak için eğitim sırasında bağlantıları rastgele kapatılmıştır.
- Conv2D, BatchNormalization, MaxPooling2D ve Dropout katmanları sırasıyla tekrarlanır. Filtre sayıları artarak daha karmaşık özelliklerin öğrenilmesi amaçlanmıştır.
- Flatten Katmanı: Önceki katmanlardan gelen özellik haritaları düzleştirilmiştir.
- Fully Connected Katmanlar: Bu katmanlar, düzleştirilmiş özellik haritasından çıkan özellikleri kullanarak daha karmaşık özellikler öğrenmektedir. Aktivasyon fonksiyonu olarak ReLU kullanılmıştır.
- Dense (256, activation='relu', activity_regularizer=regularizers. L1(1’e-4)).
- Çıkış Katmanı (Dense): Sınıflandırma için kullanılan çıkış katmanıdır.

- Dense (4, activation='sigmoid', activity_regularizer=regularizers. L1(1’e-4)), çoklu etiketli sınıflandırma problemi çalışıldığından dolayı aktivasyon fonksiyonu olarak sigmoid kullanılmıştır.
- Metrikler (Metrics): Binary Accuracy, Precision, Recall, AUC ve F1 skoru dahil olmak üzere çeşitli metrikler kullanılmıştır. Metrikler için oluşturulan fonksiyon, verilen giriş şekline sahip bir ESA modelini oluşturmuş ve belirtilen metriklerle derlemiştir.
- Model Derleme: Adam optimizasyonu tercih edilmiştir.
- Kayıp fonksiyonu olarak binary_crossentropy kullanılmıştır.

Oluşturulan fonksiyonda toplamda 24 katman kullanılmıştır. Bu sayı, her bir farklı katman türünün toplam sayısını ifade etmektedir. Ayrıca, ağın karmaşıklığını artırmak ve daha iyi genelleme yapabilmek amacıyla kullanılan düğüm seyreltme katmanları, normalizasyon katmanları ve aktivasyon fonksiyonları da bu sayıya dahildir. Şekil 42’de mimari gösterilmiştir.



Şekil 42. Oluşturulan Modelin Mimarisi

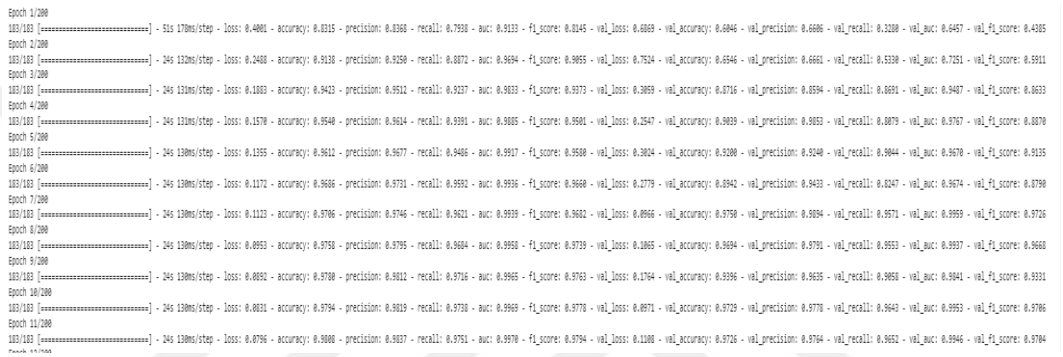
Son olarak modele ait bir özet alınmış ve kaç tane parametre olduğu görülmüştür. Özet alınıp her şeyin beklenildiği gibi olduğunu doğrulamak için model kontrol edilmiştir. Model özeti Şekil 43’te gösterilmiştir.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 220, 220, 16)      1216

 batch_normalization (Batch  (None, 220, 220, 16)      64
 Normalization)

 max_pooling2d (MaxPooling2  (None, 110, 110, 16)      0
 D)

 dropout (Dropout)           (None, 110, 110, 16)      0

 conv2d_1 (Conv2D)           (None, 106, 106, 32)      12832

 max_pooling2d_1 (MaxPoolin  (None, 53, 53, 32)        0
 g2D)

 batch_normalization_1 (Bat  (None, 53, 53, 32)        128
 chNormalization)

 dropout_1 (Dropout)         (None, 53, 53, 32)        0

 conv2d_2 (Conv2D)           (None, 49, 49, 64)        51264

 max_pooling2d_2 (MaxPoolin  (None, 24, 24, 64)        0
 g2D)

 batch_normalization_2 (Bat  (None, 24, 24, 64)        256
 chNormalization)

 dropout_2 (Dropout)         (None, 24, 24, 64)        0

 conv2d_3 (Conv2D)           (None, 20, 20, 128)       204928


 max_pooling2d_3 (MaxPoolin  (None, 10, 10, 128)       0
 g2D)

 batch_normalization_3 (Bat  (None, 10, 10, 128)       512
 chNormalization)

 dropout_3 (Dropout)         (None, 10, 10, 128)       0

 conv2d_4 (Conv2D)           (None, 6, 6, 128)         409728

 max_pooling2d_4 (MaxPoolin  (None, 3, 3, 128)         0
 g2D)

 batch_normalization_4 (Bat  (None, 3, 3, 128)         512
 chNormalization)

 dropout_4 (Dropout)         (None, 3, 3, 128)         0

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 256)               295168

 dropout_5 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_6 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dropout_7 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 4)                 260

=================================================================
Total params: 1018020 (3.88 MB)
Trainable params: 1017284 (3.88 MB)
Non-trainable params: 736 (2.88 KB)
_________________________________________________________________
```

Şekil 43. Model Özeti

4.1.2 Yöntemin eğitimi

Eğitim tamamlandığında, eğitim sürecinin geçmişini (history()) içeren bir nesne elde edilmiştir. Bu history() nesnesi, eğitim ve doğrulama setleri üzerindeki kayıp ve metrik değerlerini içerir. Eğitim sürecinin sonunda history(), modelin gelişimini değerlendirmek için genellikle eğitim ve doğrulama setleri üzerindeki kayıp ve metrik değerlerini görselleştirmek için kullanılmıştır. Bu, tüm kayıtlarda ortalama olarak düşük kayba sahip ağırlıklar ve sapmalar için bir dizi değer bulduğu anlamına gelmektedir. Model daha sonra kullanılmak üzere kaydedilmiştir.



Şekil 44. Eğitim Sırasında Ekrana Yansıyan Değerler

Şekil 44'teki ekran görüntüsünde görüldüğü üzere eğitim yapılırken baştaki epoklar sırasında değerler düşükken, sonraki değerlerde bir yükseliş gözlemlenmiştir.

4.1.3 Yöntemin değerlendirme sonuçları

Yapılan eğitim sonucu elde edilen metrik sonuçları Tablo 4'te gösterilmiştir. Tablo 4'te görüldüğü üzere kayıp değeri düşük, doğruluk değeri yüksektir. Genel olarak değerler eğitimin başarılı olduğunu göstermektedir.

Tablo 4. Model Performans Tablosu

Metrik Adı	Eğitim	Doğrulama
Kayıp	0.010	0.066
Doğruluk	0.999	0.988
Kesinlik	1.000	0.993
Duyarlılık	0.997	0.981
AUC	1.000	0.994
F1 Skor	0.999	0.987

Oluşturulan modelin eğitim ve test setleri üzerindeki performansının değerlendirme sonuçları evaluate() fonksiyonu ile belirlenmiştir. Giriş verileri (x_train ve x_test) ve gerçek etiketler (y_train ve y_test) arasındaki kaybı gösteren metriklere ait ortalama sonuçlar elde edilmiştir. Şekil 45'te gösterilen 366/366 sayısı eğitim setinin, 157/157 sayısı test setinin kaç mini-yığına bölündüğünü göstermektedir.

```
366/366 [==============================] - 15s 17ms/step - loss: 0.0097 - accuracy: 0.9987 - precision: 0.9998 - recall: 0.9973 - auc: 0.9997 - f1_score: 0.9985
157/157 [==============================] - 3s 19ms/step - loss: 0.0659 - accuracy: 0.9876 - precision: 0.9925 - recall: 0.9811 - auc: 0.9942 - f1_score: 0.9866
```

Şekil 45. Ortalama Metrik Sonuçları

Şekil 46'da 200 epok boyunca eğitim ve doğrulama setleri üzerinde model performansını belirten metrikler gösterilmiştir. İdeal olarak, kaybın sıfıra ve doğruluğun bire (örn. %100) gitmesi beklenmektedir. Modelde her zaman bir miktar hata olması beklenir. Amaç, belirli bir veri kümesi için mümkün olan en düşük kaybı ve en yüksek doğruluğu sağlayan bir model konfigürasyonu ve eğitim konfigürasyonu elde etmektir.

Epoch	loss	accuracy	precision	recall	auc	f1_score
0	0.393768	0.837384	0.842733	0.801135	0.917250	0.821490
1	0.242769	0.916674	0.928098	0.890480	0.971528	0.908956
2	0.186500	0.941616	0.950997	0.922458	0.983995	0.936500
3	0.152186	0.956778	0.964890	0.941672	0.989183	0.953073
4	0.134555	0.962629	0.968065	0.951324	0.991647	0.959581
...	...	...	...	...	...	...
195	0.008425	0.999189	0.999222	0.999039	0.999885	0.999136
196	0.009237	0.998847	0.998993	0.998536	0.999860	0.998770
197	0.010101	0.998633	0.998536	0.998536	0.999763	0.998541
198	0.009976	0.998612	0.998582	0.998445	0.999808	0.998509
199	0.009225	0.998804	0.998765	0.998673	0.999856	0.998702

[200 rows x 6 columns]

Şekil 46. Metriklerin Sonuçlarının Eğitim Boyunca Değerleri

Eğitim seti sonuçlarına göre; modelin eğitim seti üzerinde yaptığı tahminlerin gerçek değerlere olan uzaklığını ölçen ortalama kayıp değeri 0.0097 olarak hesaplanmıştır. Düşük kayıp, modelin ne kadar iyi öğrendiğini göstermektedir. Doğruluk değeri 0.9987 (99.87%)'dir. Eğitim setindeki doğru sınıflandırma oranındaki yüksek doğruluk, modelin eğitim setinde iyi performans sergilediğini göstermektedir. Kesinlik değeri 0.9998 (99.98%)'dur. Eğitim setindeki kesinlik değeri modelin pozitif tahminlerinin %99'unun gerçekten pozitif olduğunu göstermektedir. Duyarlılık değeri 0.9973 (99.73%)'dur. Eğitim setindeki duyarlılık, gerçek pozitif örneklerin % 99'unun doğru bir şekilde tahmin

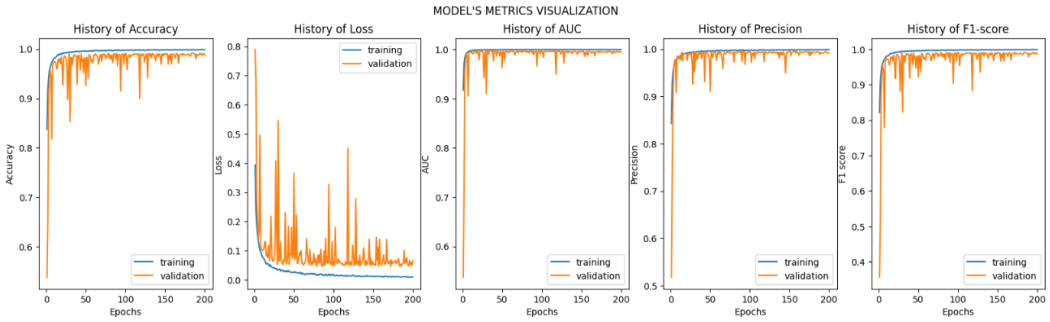
edildiğini göstermektedir. Eğitim setindeki AUC değeri 0.9997 (99.97%)'dur. ROC eğrisinin altındaki alanı ölçmekte ve sınıflandırma performansını değerlendirmektedir. Eğitim sonucunda alınan değer oldukça yüksektir. Eğitim setindeki F1 skor değeri 0.9985 (99.85%)'dur. F1 skor, kesinlik ve duyarlılığın harmonik ortalamasıdır ve modelin sınıflandırma performansını %99 ölçmüştür.

Doğrulama (Validation) seti sonuçlarına göre; Modelin doğrulama seti üzerinde yaptığı tahminlerin gerçek değerlere olan uzaklığını ölçen ortalama kayıp değeri 0.0659'dur. Eğitim seti ile kıyaslandığında daha yüksek bir kayıp değeri hesaplanmıştır. Doğruluk değeri 0.9876 (98.76%)'dır. Doğrulama setindeki doğru sınıflandırma oranı, modelin yeni ve görünmeyen veriler üzerinde ne kadar iyi performans gösterdiğini göstermektedir. Kesinlik değeri 0.9925 (99.25%)'dur. Doğrulama setindeki yüksek kesinliğin, modelin pozitif tahminlerinin doğruluğunu göstermektedir. Duyarlılık değeri 0.9811 (98.11%)'dir. Doğrulama setindeki duyarlılık değeri, gerçek pozitif örneklerin %98 doğru bir şekilde tahmin edildiğini göstermektedir. Doğrulama setindeki AUC değeri 0.9942 (99.42%)'dur ve eğitim setine benzer bir performans göstermektedir. Doğrulama setindeki F1 skor değeri 0.9866 (98.66%)'dır ve yüksek bir F1 skor, modelin hem kesinliği hem de duyarlılığı dengeli bir şekilde başardığını göstermektedir.

Sonuç olarak, model eğitim setinde yüksek performans göstermektedir ve doğrulama setinde de iyi bir genellemeye sahip olduğu görünmektedir.

Eğitim ve test setleri üzerindeki model performansını görselleştirmek için bir Dataframe oluşturulmuş ve HTML formatında bir tabloya dönüştürülerek görüntülenmiştir. Bu, tablo formatındaki metrik değerlerini daha okunabilir hale getirmek ve değerlendirmeyi daha anlaşılır bir şekilde sunmak için kullanışlı bir yöntemdir.

Sonuçları görselleştirmek için programa Matplotlib Kütüphanesi dahil edilmiştir. Eğitim geçmişi (history.history) verileri kullanılarak modelin eğitim ve doğrulama setleri üzerindeki metriklerini grafikleştirmek için Matplotlib kütüphanesi kullanılmıştır. Şekil 47'de tüm metriklerin grafiksel gösterimi bulunmaktadır.

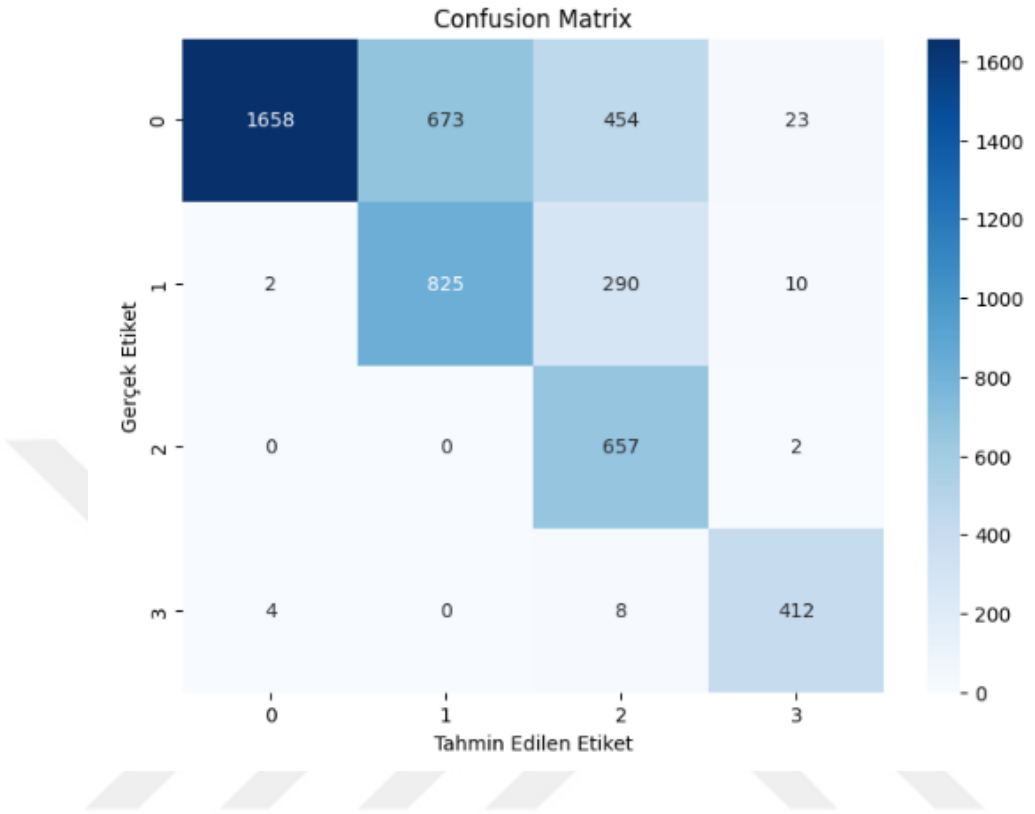


Şekil 47. Tüm Metriklerin Grafiksel Gösterimi

Accuracy (Doğruluk) grafiğindeki mavi çizgi, eğitim seti üzerindeki doğruluk değerini temsil etmektedir. Turuncu çizgi ise doğrulama seti üzerindeki doğruluk değerini temsil etmektedir. Eğitim ve doğrulama doğruluk çizgilerinin birbirine yakın olması doğruluk grafiğinin başarılı bir modele ait olduğunu göstermektedir. Kayıp grafiğindeki mavi çizgi, eğitim seti üzerindeki kayıp değerini temsil etmektedir. Turuncu çizgi ise doğrulama seti üzerindeki kayıp değerini temsil etmektedir. AUC grafiğindeki mavi çizgi, eğitim seti üzerindeki AUC değerini temsil etmektedir. Turuncu çizgi ise doğrulama seti üzerindeki AUC değerini temsil etmektedir. AUC, modelin sınıflandırma performansını ölçen bir metriktir. Bire ne kadar yakın ise, modelin performansı o kadar iyi olmaktadır. Grafikte görüldüğü üzere bire eşitlenmiştir. Kesinlik ve F1 skor grafiklerindeki mavi çizgiler, eğitim seti üzerindeki kesinlik ve F1 skor değerlerini temsil etmektedir. Turuncu çizgiler ise doğrulama seti üzerindeki kesinlik ve F1skor değerlerini temsil etmektedir. Bu metrikler, sınıflandırma modelinin performansını farklı açılardan değerlendirmeye yaramaktadır. Dikkat edilmesi gereken, bu metriklerin eğitim ve doğrulama setleri arasındaki dengeyi göstermesidir. Genel olarak, grafiğin düzgün bir şekilde eğitim ve doğrulama setlerine uyması beklenmektedir. Eğer eğitim setindeki performans çok yüksek ancak doğrulama setinde performans düşükse model aşırı uydurma yapabilmektedir. Grafiklerden izlendiği üzere model uyumlu çalışmakta ve ezberleme görülmemektedir.

Modelin sınıflandırma performansını daha detaylı bir şekilde değerlendirmek için classification_report() ve confusion_matrix() fonksiyonları kullanılmıştır. Sonuçları yazdırırken, classification_report() fonksiyonu tarafından üretilen metrikler ve confusion_matrix() tarafından üretilen karmaşıklık matrisi ekrana yazdırılmıştır. Ayrıca

seaborn kütüphanesi kullanılarak confusion matrix() Şekil 48'deki gibi görselleştirilmiştir.



Şekil 48. Karmaşıklık Matrisi

Karmaşıklık matrisi her bir sınıfın gerçek ve tahmin edilen örnek sayılarını içermektedir. Matrisin köşegeninde (diagonal) yer alan sayılar, doğru tahmin edilen örnekleri temsil ederken, matrisin dışındaki diğer sayılar ise yanlış tahmin edilen örnekleri temsil etmektedir. Karmaşıklık matrisinde görüldüğü üzere, gerçek etiket değeri 0 (kral_sakir) olan çizgi karaktere ait 2808 görüntüden 1658 tanesi doğru tahmin edilmiştir. 673 tanesi 1 nolu etiket değerine, 454 tanesi 2 nolu etiket değerine, 23 tanesi de 3 nolu etiket değerine sahip görüntü olarak tahmin edilmiştir. Gerçek etiket değeri 1 (remzi) olan çizgi karaktere ait 1127 görüntüden 825 tanesi doğru tahmin edilmiştir. İki tanesi 0 nolu etiket değerine, 290 tanesi 2 nolu etiket değerine, 10 tanesi de 3 nolu etiket değerine sahip görüntü olarak tahmin edilmiştir. Gerçek etiket değeri 2 (necati) olan çizgi karaktere ait 659 görüntüden 657 tanesi doğru tahmin edilmiştir. Sadece iki tanesi 3 nolu etiket değerine tahmin edilmiş diğer etiketlerde yanlış bir tahmin gözlemlenmemiştir. Gerçek etiket değeri 3 (canan) olan çizgi karaktere ait 424 görüntüden 412 tanesi doğru tahmin

edilmiştir. Dört tanesi 0 nolu etiket değerine, sekiz tanesi 2 nolu etiket değerine tahmin edilmiş, 1 nolu etiket değerine sahip görüntü olarak herhangi bir yanlış tahminde bulunulmamıştır.

Tablo 5. Sınıflara Göre Test Seti Metrik Değerleri

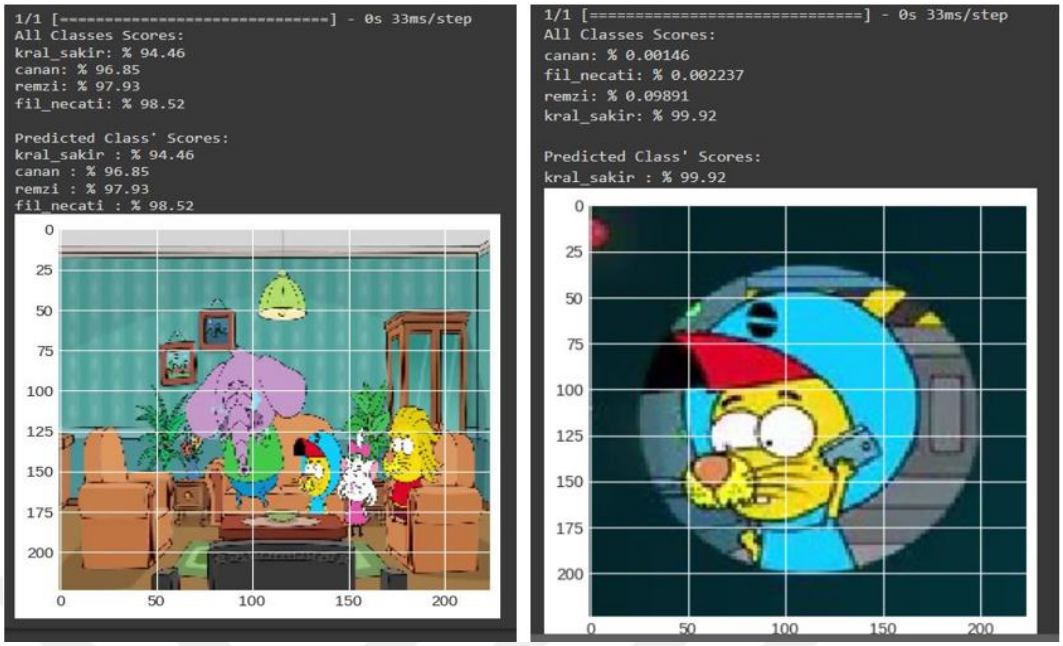
```
157/157 [==============================] - 2s 11ms/step
              precision    recall  f1-score   support

           0       1.00      0.59      0.74      2808
           1       0.55      0.73      0.63      1127
           2       0.47      1.00      0.64       659
           3       0.92      0.97      0.95       424

    accuracy                           0.71      5018
   macro avg       0.73      0.82      0.74      5018
weighted avg       0.82      0.71      0.72      5018
```

Tablo 5'teki sonuçlara göre Sınıf 2'nin duyarlılık değeri 1.00 (100%) olduğu için, bu sınıfın tüm gerçek pozitif örnekleri doğru bir şekilde sınıflandırılmış gibi görünmektedir. Sınıf 0'nın kesinliği yüksek (1.00), ancak duyarlılığı düşük (0.59) olduğu için, bu sınıfın bir kısmı doğru bir şekilde sınıflandırılmış, ancak bir kısmı da sınıflandırılamamış olabilir. Sınıf 3 için yüksek kesinlik (0.92) ve yüksek duyarlılık (0.97), bu sınıfın doğru bir şekilde sınıflandırıldığını göstermektedir. Sınıf 1 için kesinlik ve duyarlılık birbirine yakın, ancak F1 skor düşüktür. Bu durum, Sınıf 1'in kesin ve duyarlı olduğu, ancak modelin bir dengeye sahip olduğu söylenebilmektedir.

Sonuçlara göre model başarılı bir şekilde çalışmaktadır. Veri setimiz oluşturulan çoklu etiketli bir sınıflandırma modelinde eğitilmiş ve değerlendirilmiştir. Sonuçların yüksek bir başarı ile tahmin edildiği görülmüştür.

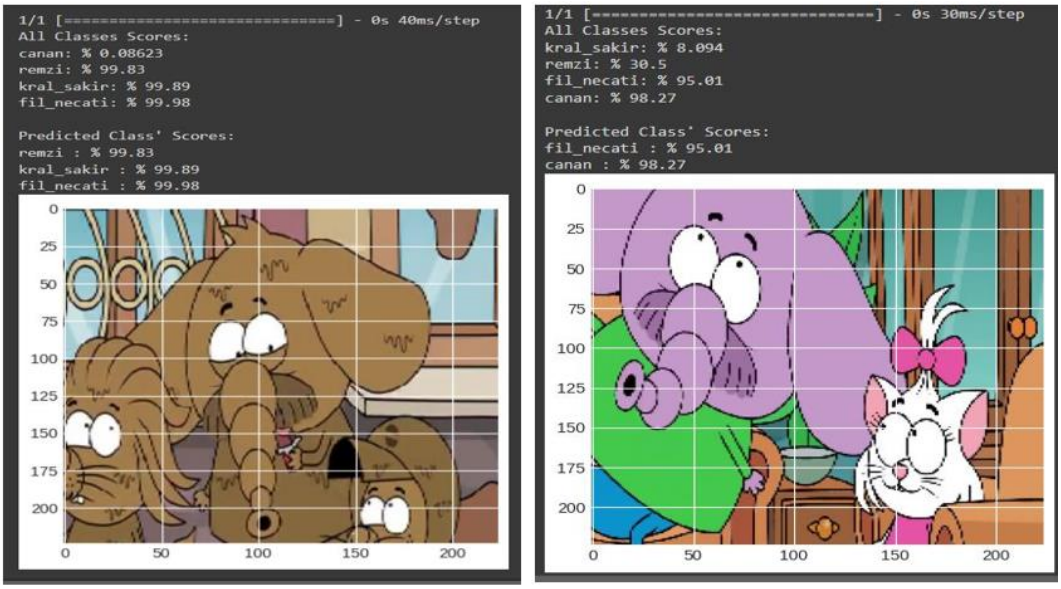
4.1.4 Yöntemin tahmin sonuçları

KralSakir veri setine ait görsellerden modelin yüksek tahmin oran sonuçları Şekil 49 ve 50'de verilmiştir.



Şekil 49. Görüntü Sınıflandırma Tahmin Sonuçları 1

Şekil 49'daki ilk görselde kral_sakir %94.46, canan %96.85, remzi %97.93 ve necati %98.52 oranlarında tahmin edilmiştir. İkinci görselde ise kral_sakir %99.92 oranında model tarafından tahmin edilmiştir.



Şekil 50. Görüntü Sınıflandırma Tahmin Sonuçları 2

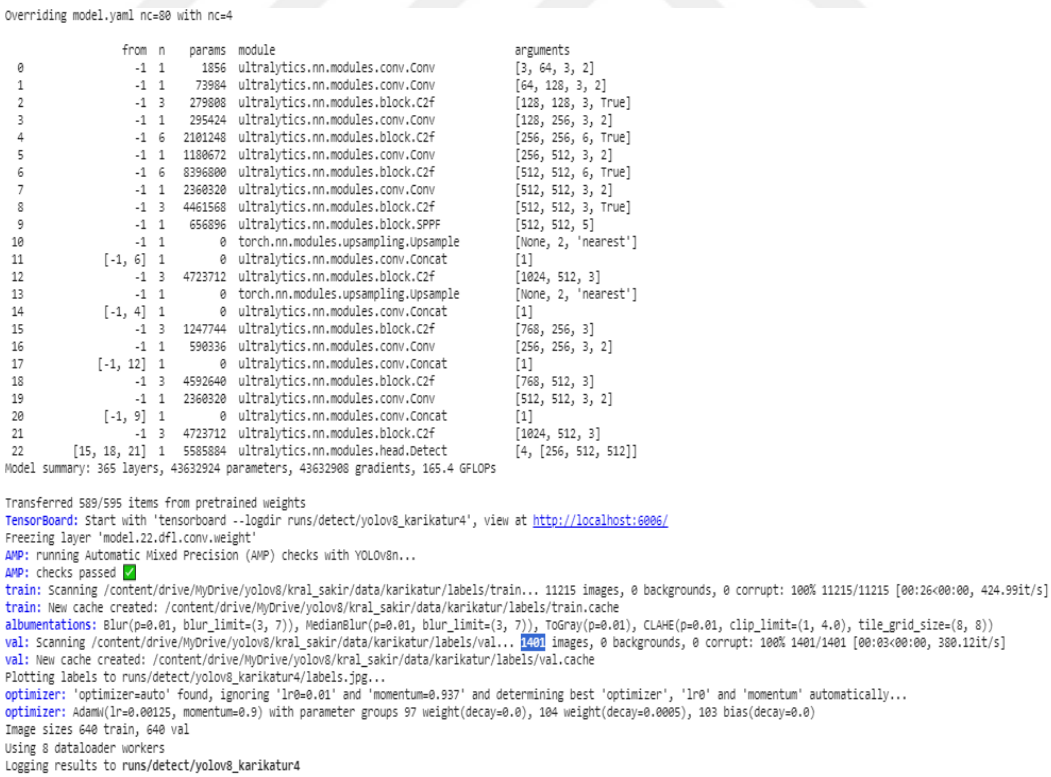
Şekil 50’de ise, ilk görselde remzi %99.83 kral_sakir %99.89, ve necati %99.98 oranlarında tahmin edilmiştir. İkinci görselde ise necati %95.01 ve canan %98.27 oranında model tarafından tahmin edilmiştir.

4.2 YOLOv8 ile Nesne Tespiti Yöntemi

Çalışılan ikinci yöntem olan nesne tanıma yönteminin mimarisinin oluşturulması, kodlanması ve metriklerin sonuçlarıyla ilgili bilgileri aşağıda başlıklar halinde sunulmuştur.

4.2.1 Yöntemin mimarisi

Modelde 365 katman, 43632925 parametre, 43632908 gradyan ve 165.4 GFLOPs (Gigaflops) bulunmaktadır. Modelin özeti Şekil 50’de verilmiştir. Optimizasyon algoritması Yolov8 mimarisinde otomatik olarak seçilmektedir. AdamW optimizasyon algoritması, lr=0.00125 ve momentum=0.9 olarak belirlenmiştir. Şekil 51’de modelin özeti gösterilmiştir.



```
Overriding model.yaml nc=80 with nc=4

                   from  n    params  module                                       arguments
  0                  -1  1      1856  ultralytics.nn.modules.conv.Conv             [3, 64, 3, 2]
  1                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  2                  -1  3    279808  ultralytics.nn.modules.block.C2f             [128, 128, 3, True]
  3                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  4                  -1  6   2101248  ultralytics.nn.modules.block.C2f             [256, 256, 6, True]
  5                  -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  6                  -1  6   8396800  ultralytics.nn.modules.block.C2f             [512, 512, 6, True]
  7                  -1  1   2360320  ultralytics.nn.modules.conv.Conv             [512, 512, 3, 2]
  8                  -1  3   4461568  ultralytics.nn.modules.block.C2f             [512, 512, 3, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  3   4723712  ultralytics.nn.modules.block.C2f             [1024, 512, 3]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  3   1247744  ultralytics.nn.modules.block.C2f             [768, 256, 3]
 16                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  3   4592640  ultralytics.nn.modules.block.C2f             [768, 512, 3]
 19                  -1  1   2360320  ultralytics.nn.modules.conv.Conv             [512, 512, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  3   4723712  ultralytics.nn.modules.block.C2f             [1024, 512, 3]
 22       [15, 18, 21]  1   5585884  ultralytics.nn.modules.head.Detect           [4, [256, 512, 512]]
Model summary: 365 layers, 43632924 parameters, 43632908 gradients, 165.4 GFLOPs

Transferred 589/595 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/yolov8_karikatur4', view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
AMP: checks passed ✅
train: Scanning /content/drive/MyDrive/yolov8/kral_sakir/data/karikatur/labels/train... 11215 images, 0 backgrounds, 0 corrupt: 100% 11215/11215 [00:26<00:00, 424.99it/s]
train: New cache created: /content/drive/MyDrive/yolov8/kral_sakir/data/karikatur/labels/train.cache
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid_size=(8, 8))
val: Scanning /content/drive/MyDrive/yolov8/kral_sakir/data/karikatur/labels/val... 1401 images, 0 backgrounds, 0 corrupt: 100% 1401/1401 [00:03<00:00, 380.12it/s]
val: New cache created: /content/drive/MyDrive/yolov8/kral_sakir/data/karikatur/labels/val.cache
Plotting labels to runs/detect/yolov8_karikatur4/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.00125, momentum=0.9) with parameter groups 97 weight(decay=0.0), 104 weight(decay=0.0005), 103 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to runs/detect/yolov8_karikatur4
```

Şekil 51. Model Özeti

4.2.2 Yöntemin eğitimi

ESA mimarisinde geliştirilen YOLOv8l modeli eğitim için tercih edilmiştir. YOLOv8l ile oluşturulan modelde eğitim için gerekli minimum epok sayısına, kullanılan veri setine, modelin karmaşıklığına, başlangıç ağırlıklarına ve eğitim hiper-parametrelerine bağlı olarak değişebilmektedir. Her uygulama farklıdır. Yöntem ilk denemede 100 epokla eğitilmiştir. Eğitimde önemli olan bir nokta bulunmaktadır. Herhangi bir şekilde eğitim yarıda kesildiğinde eğitime devam edecek ve son ağırlıkların bulunduğu dosyayı kaydedecek bir kod yönteme eklenmelidir. 54 epoktan sonra bir düzelme gözlenmediği için eğitim sonlandırılmıştır. Epok sayısı 20 olarak ayarlanmış ve sonuçlar 20 epok üzerinden değerlendirmiştir. Eğitim parametreleri olarak öğrenme hızı 2,64 ms olarak gözlemlenmiştir. Şekil 52'de model eğitiminden bir kesit gösterilmiştir.

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 1/20      5.53G      1.228      1.234      1.563         28        640: 100% 1402/1402 [09:13<00:00,  2.53it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 88/88 [00:21<00:00,  4.17it/s]
             all       1401       2855      0.664      0.733      0.727      0.456

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 2/20      5.84G      1.197      1.063      1.537         43        640: 100% 1402/1402 [08:56<00:00,  2.61it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 88/88 [00:19<00:00,  4.47it/s]
             all       1401       2855      0.737      0.802      0.795      0.514

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 3/20      5.87G      1.154      0.981      1.502         31        640: 100% 1402/1402 [08:52<00:00,  2.63it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 88/88 [00:19<00:00,  4.49it/s]
             all       1401       2855       0.77      0.811       0.82      0.542

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 4/20      5.86G      1.112     0.9111      1.466         34        640: 100% 1402/1402 [08:51<00:00,  2.64it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 88/88 [00:19<00:00,  4.49it/s]
             all       1401       2855      0.784      0.816      0.845      0.581

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 5/20      5.88G      1.089     0.8738      1.441         29        640: 100% 1402/1402 [08:51<00:00,  2.64it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 88/88 [00:19<00:00,  4.48it/s]
             all       1401       2855      0.777      0.834      0.865      0.596

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 6/20      5.86G      1.065     0.8411      1.415         22        640: 100% 1402/1402 [08:51<00:00,  2.64it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 88/88 [00:19<00:00,  4.47it/s]
             all       1401       2855      0.821      0.848      0.891       0.62

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 7/20      5.86G      1.041     0.8013      1.409         34        640: 100% 1402/1402 [08:51<00:00,  2.64it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 88/88 [00:19<00:00,  4.47it/s]
             all       1401       2855      0.835      0.855      0.907      0.645
```

Şekil 52. Modelin Eğitimi

4.2.3 Yöntemin değerlendirme sonuçları

Yöntemde sonuçlar runs klasöründe results.cvs dosyasına kaydedilmektedir. Eğitim devam ettikçe kayıp değerinde düşme kesinlik değerinde ise bir yükselme mevcuttur. Şekil 53'te gözüktüğü üzere; 20 epokta kesinlik %90, duyarlılığı ise %92'dir. Kayıp değerlerinin ise eğitim süresince düştüğü gözlemlenmektedir. Eğitim kayıp değeri %44, doğrulama kayıp değeri ise %4 olarak gözükmektedir.

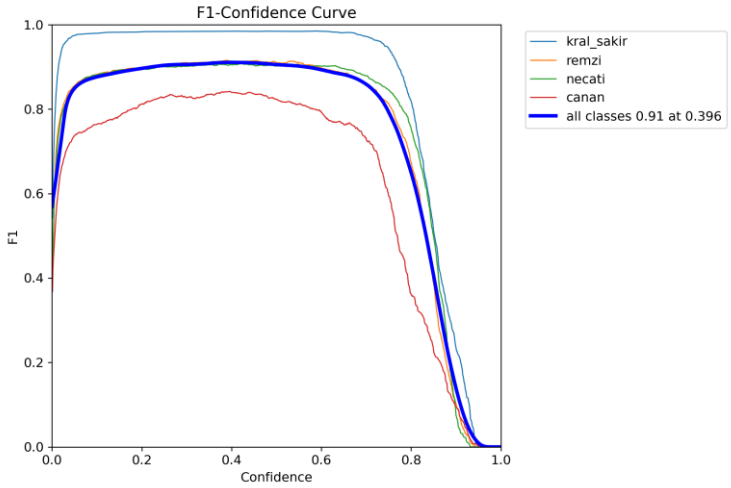
epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precisior	metrics/recall(B)	metrics/mAP50(I	metrics/mAP50-!	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
1	1.2276	1.2337	1.5626	66.412	73.304	72.686	45.615	1.2617	99.173	1.6221	41.637	41.637	41.637
2	1.1965	1.0632	1.5372	73.736	802	7.949	51.424	1.1262	81.149	1.4895	7.918	7.918	7.918
3	1.1543	98.099	1.5018	77.014	81.131	8.196	54.169	1.119	76.765	1.5561	1.126	1.126	1.126
4	1.1121	91.111	1.466	78.392	81.592	84.471	58.082	1.0472	70.804	1.4146	10.644	10.644	10.644
5	1.0891	87.385	1.4414	77.709	83.408	86.527	59.614	1.047	67.345	1.3953	10.644	10.644	10.644
6	1.0653	84.105	1.4148	82.111	84.826	89.066	62.048	01.03	6.186	1.4054	10.025	10.025	10.025
7	1.0406	80.132	1.4092	83.453	85.534	90.671	64.498	1.1958	59.044	1.3724	94.062	94.062	94.062
8	1.0329	78.288	1.3885	86.243	85.315	91.919	65.362	99.674	57.457	1.3738	87.875	87.875	87.875
9	1.016	76.302	1.3756	84.853	85.033	91.012	65.425	98.479	56.061	1.353	81.688	81.688	81.688
10	1.1956	74.364	1.3733	8.617	86.979	922	66.241	98.254	54.673	1.3675	755	755	755
11	1.1953	59.162	1.3858	86.382	87.791	92.342	6.666	97.856	5.402	1.3361	69.313	69.313	69.313
12	9.964	57.014	1.401	85.771	90.987	93.876	6.819	96.897	50.544	1.3555	63.125	63.125	63.125
13	97.745	54.749	1.3882	89.241	88.328	94.333	68.559	96.914	50.298	1.3391	56.938	56.938	56.938
14	96.583	52.528	1.36	87.375	91.367	94.176	69.349	95.632	48.688	1.326	5.075	5.075	5.075
15	96.065	51.482	1.3788	87.017	92.799	94.957	69.991	95.277	47.267	1.3485	44.563	44.563	44.563
16	94.565	5.004	1.3603	89.128	89.955	94.838	69.873	94.659	46.884	1.2772	38.375	38.375	38.375
17	93.698	48.467	1.3159	88.912	92.011	94.921	70.353	942	45.479	1.2797	32.188	32.188	32.188
18	92.435	46.854	1.3035	88.513	92.853	95.185	70.958	93.223	44.847	1.2977	26	26	26
19	91.488	45.669	1.3147	8.893	92.315	95.264	71.231	92.905	4.385	1.2771	19.813	19.813	19.813
20	90.181	44.319	1.2942	90.135	92.043	9.548	7.157	92.513	4.281	1.275	13.625	13.625	13.625

Şekil 53. Metrik Sonuçlarının CSV Dosyası

Result dosyasının içeriğindeki terimlerin anlamları şöyledir:

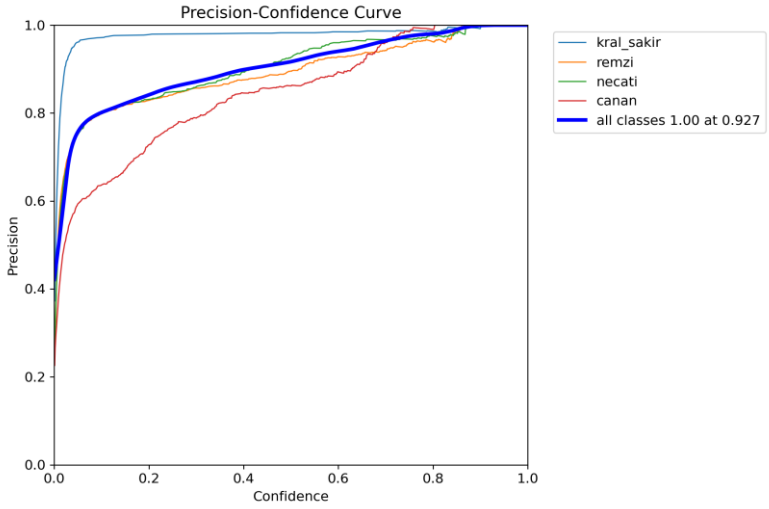
- train/box_loss, train/cls_loss, train/dfl_loss: Eğitim sırasında kutu kaybı, sınıflandırma kaybı ve deformasyon kaybı değerlerini temsil etmektedir. Bu, modelin eğitim verilerine karşı performansını ölçmektedir.
- metrics/precision(B), metrics/recall(B), metrics/mAP50(B), metrics/mAP50-95(B): Eğitim sırasında ölçülen metrik değerlerini temsil etmektedir. Bu, özellikle nesne algılama görevlerinde kullanılan doğruluk metrikleridir. mAP50, ortalama kesinliği ifade etmektedir.
- val/box_loss, val/cls_loss, val/dfl_loss: Doğrulama veri seti üzerinde kutu kaybı, sınıflandırma kaybı ve deformasyon kaybı değerlerini temsil etmektedir. Bu, modelin genelleme yeteneğini değerlendirmektedir.
- lr/pg0, lr/pg1, lr/pg2: Öğrenme oranı ve grup öğelerini temsil etmektedir. Birden fazla parametre grubu olduğunda, her bir grup için öğrenme oranlarını göstermektedir.

Şekil 54'teki F1 skor grafiğe göre, sınıf bazında F1 skor değerlerinin yüksek olduğu görülmektedir. Bütün sınıflar %91'in üzerinde bir F1 skor değerine sahiptir. Bu, modelin bu sınıfları doğru pozitifleri ve gerçek pozitifleri başarılı bir şekilde tahmin etme konusunda iyi bir performans sergilediğini göstermektedir. F1 skor tablosuna göre, modelin çoğu sınıf için yüksek F1 skor değerleri elde ettiği görülmektedir. Bu, modelin hem doğru pozitifleri hem de gerçek pozitifleri iyi bir şekilde tahmin etme yeteneğinin yüksek olduğunu göstermektedir.



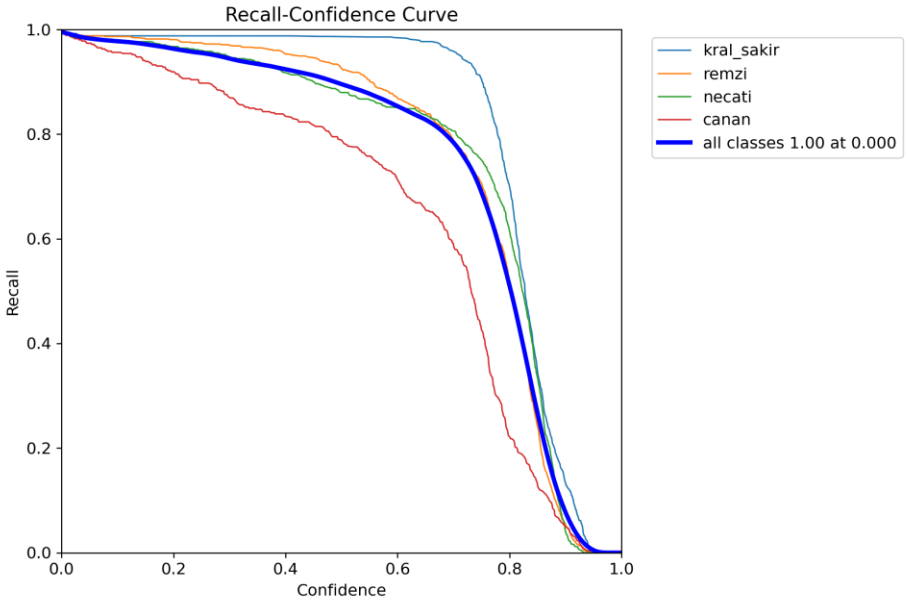
Şekil 54. F1 Skor Grafiği

Şekil 55'teki Kesinlik grafiğine göre, sınıf bazında kesinlik değerlerinin yüksek olduğu görülmektedir. Tüm sınıflarda %92'nin üstünde bir değer elde edilmiştir.



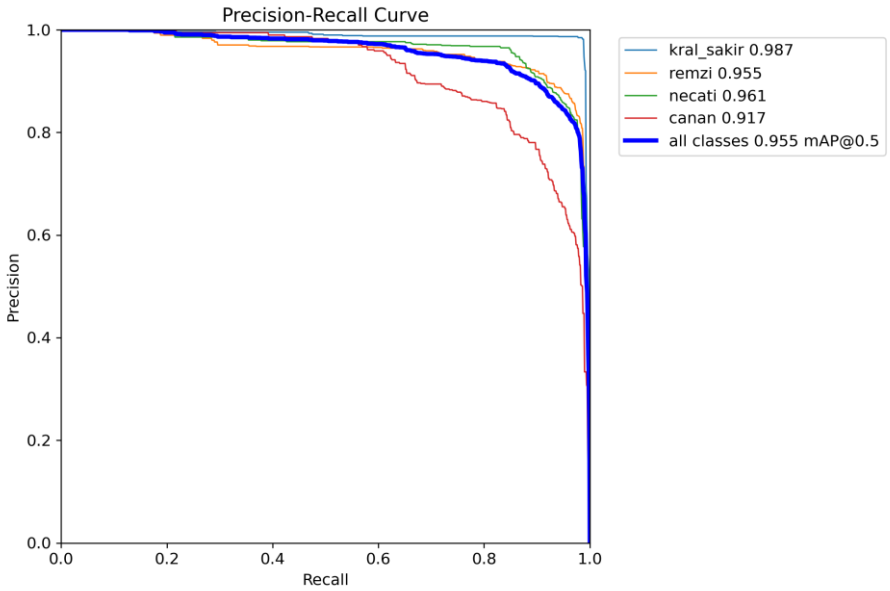
Şekil 55. Kesinlik Grafiği

Şekil 56'daki Duyarlılık grafiğine göre, sınıf bazında duyarlılık değeri bire yakınsamıştır. Tüm sınıflarda %95'in üstünde bir değer elde edilmiştir.



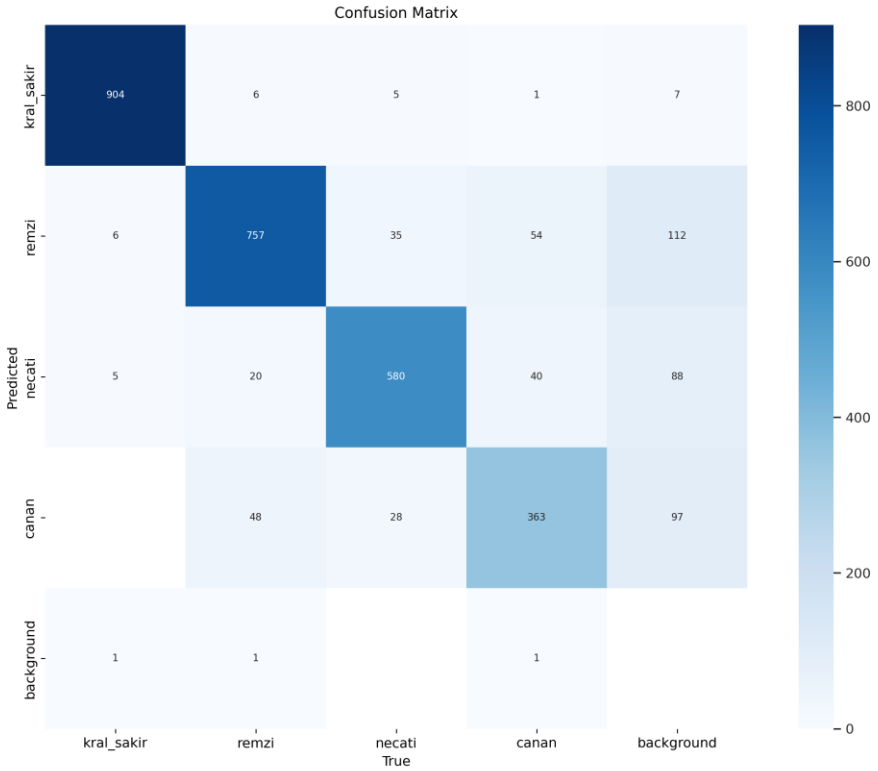
Şekil 56. Duyarlılık Grafiği

Şekil 57'deki Kesinlik - Duyarlılık Grafiğinde, farklı sınıflar için elde edilen Kesinlik ve duyarlılık değerleri görülmektedir. kral_sakir sınıfı için %98, remzi sınıfı için %95, necati sınıfı için %96, canan sınıfı içinse %91 değerleri elde edilmiştir.



Şekil 57. Kesinlik - Duyarlılık Grafiği

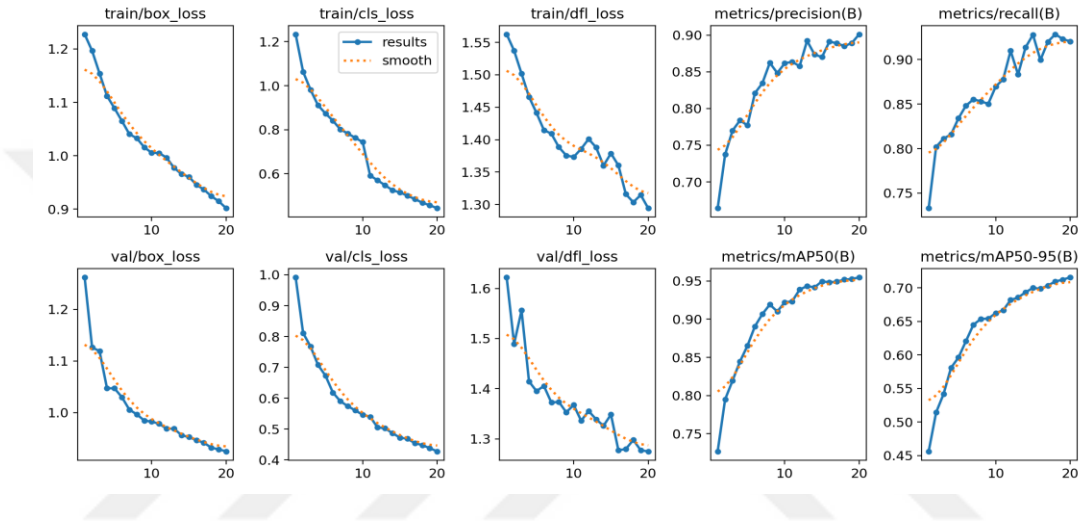
Karmaşıklık matrisinde modelin tahmin ettiği sınıfların, genel olarak doğru sınıflar olduğu görülmektedir. Bu matris, modelin belirli sınıfları doğru bir şekilde tanıma yeteneğini ve diğer sınıflarla karıştırma durumlarını göstermektedir. Background olarak adlandırılan sınıf, modelin nesne olarak tanımlamak yerine arka plan olarak gördüğü nesneleri ifade etmektedir. Bu durum, modelin genel performansını değerlendirmek için kullanılmakta ve geliştirmelerin nerede yapılması gerektiğini anlamak için önem arz etmektedir.



Şekil 58. Karmaşıklık Matrisi

Şekil 58'deki karmaşıklık matrisinde görüldüğü üzere, gerçek etiket değeri kral_sakir olan çizgi karaktere ait 916 görüntüden 904 tanesi doğru tahmin edilmiştir. Altı tanesi remzi etiket değerine, beş tanesi necati etiket değerine yanlış tahmin edilmiştir. Canan etiket değerine sahip yanlış bir görüntü tespit edilmemiştir. Background olarak yedi adet görüntü tespiti mevcuttur. Gerçek etiket değeri remzi olan çizgi karaktere ait 832 görüntüden 757 tanesi doğru tahmin edilmiştir. Altı tanesi kral_sakir etiket değerine, 20 tanesi necati etiket değerine, 48 tanesi de canan etiket değerine sahip görüntü olarak tahmin edilmiştir. Background olarak 112 adet görüntü tespiti mevcuttur. Gerçek etiket

değeri necati olan çizgi karaktere ait 658 görüntüden 580 tanesi doğru tahmin edilmiştir. Beş tanesi kral_sakir etiket değerine, 35 tanesi remzi etiket değerine, 28 tanesi de canan etiket değerine sahip görüntü olarak tahmin edilmiştir. Background olarak 88 adet görüntü tespiti mevcuttur. Gerçek etiket değeri canan olan çizgi karaktere ait 459 görüntüden 363 tanesi doğru tahmin edilmiştir. Bir tanesi kral_sakir etiket değerine, 54 tanesi remzi etiket değerine, 40 tanesi de necati etiket değerine sahip görüntü olarak tahmin edilmiştir. Background olarak 97 adet görüntü tespiti mevcuttur.



Şekil 59. Kayıp Değerleri Grafikleri

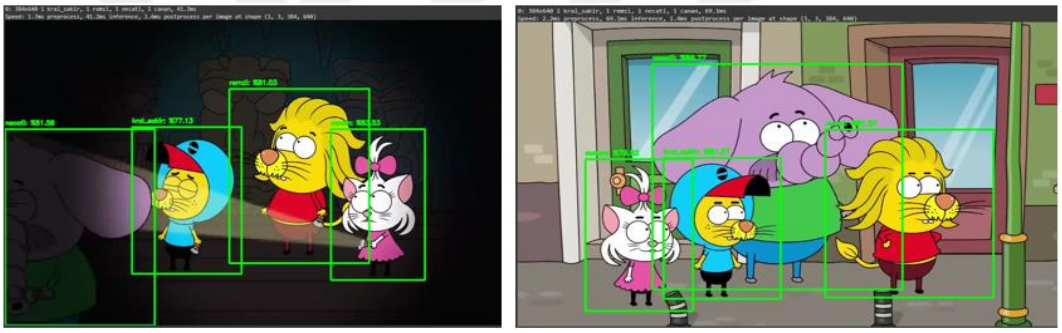
Kayıp fonksiyonu (Loss function), bir makine öğrenimi modelinin tahminlerinin gerçek değerlere ne kadar yakın veya uzak olduğunu ölçen bir fonksiyondur. Bu fonksiyon, modelin eğitim sürecinde kullanılarak parametrelerin (ağırlıkların) güncellenmesi ve modelin optimize edilmesi sağlamaktadır. Kayıp fonksiyonu, modelin performansını değerlendiren bir ölçüdür. Modelin ürettiği tahminler ile gerçek değerler arasındaki farkı ölçmekte ve bu farkın minimize edilmesi hedeflenmektedir. Fark, genellikle kayıp veya hata olarak adlandırılmakta ve eğitim sürecinde modelin bu hatayı azaltması amaçlanmıştır. Eğitim boyunca kayıp değerleri azalmıştır. Şekil 59'da kayıp değerleri grafikleri gösterilmiştir.

mAP50 (mean Average Precision at 50), nesne tespiti, nesne tanıma ve nesne algılama görevlerinde performansı değerlendirmek için kullanılmaktadır. Burada Ortalama kesinlik (AP), bir sınıf için önerilen nesnelerin doğruluk ve sıralama düzeyine dayanan bir metriktir. AP değeri, bir sınıfa ait nesnelerin doğru bir şekilde tespit edilmesini ve bu

tespitlerin sıralamasını değerlendirmektedir. AP50 ise bu metriği belirli bir eşik değeriyle (genellikle 0.5 olarak belirlenir) değerlendirmektedir. mAP50, bu AP değerlerinin bütün sınıflar için ortalamasını ifade etmektedir. Özellikle nesne algılama görevlerinde, algoritmanın nesneleri %50 veya daha fazla bir doğrulukla tespit edebilme yeteneğini değerlendirmek için kullanılmaktadır. Sonuçlarda görüldüğü üzere mAP50 değeri %90'ın üzerindedir. Tüm sonuçlara göre modelin tüm sınıfları hem doğru pozitifleri hem de gerçek pozitifleri başarılı bir şekilde tahmin etme konusunda iyi bir performans sergilediğini göstermektedir.

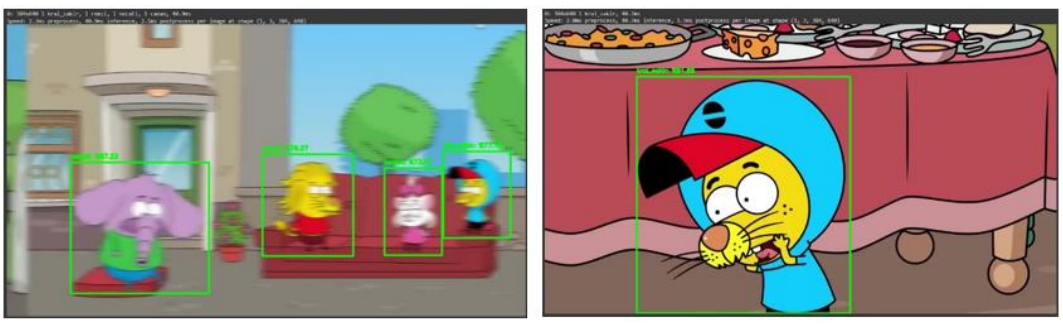
4.2.4 Yöntemin tahmin sonuçları

KralSakir veri setine ait görsellerden modelin yüksek tahmin oran sonuçları Şekil 60 ve Şekil 61'de verilmiştir.



Şekil 60. Nesne Tanıma Tahmin Sonuçları 1

Şekil 60'taki ilk görselde kral_sakir %77.13, remzi %81.03, necati %81.56 ve canan %83.53 oranlarında tahmin edilmiştir. İkinci görselde ise kral_sakir %81.77, remzi %81.57, necati %86.77 ve canan %79.52 oranlarında tahmin edilmiştir.



Şekil 61. Nesne Tanıma Tahmin Sonuçları 2

Şekil 61'deki ilk görselde kral_sakir %77.78, remzi %78.27, necati %87.23 ve canan %73.80 oranlarında tahmin edilmiştir. İkinci görselde ise kral_sakir %91.55 oranında tahmin edilmiştir. Tahmin sonuçları üzerinden genel olarak YOLOv8 ile yapılan eğitim başarılı olarak görülmüştür.

4.3 Yöntemlerin Karşılaştırılması ve KralSakir Veri Seti Üzerinden Değerlendirilmesi

Çok etiketli görüntü sınıflama yöntemi ve nesne tespiti yöntemi arasında bazı ortak noktalar bulunmaktadır. Her iki yöntemde de bir görüntünün içerdiği nesnelerin veya özelliklerin tanınması görevi bulunmaktadır. Hem çok etiketli görüntü sınıflama hem de nesne tespiti, bir görüntüde birden çok sınıfın veya nesnenin bulunabileceği durumları ele almaktadır. Her iki yöntem bir nesneyi tanıma sürecini içermektedir. Çok etiketli görüntü sınıflama ve nesne tespiti yöntemlerinde görüntüdeki nesnelerin hangi sınıflara ait olduğunun tanımlanması amaçlanmaktadır. Her iki yöntemde de bir görüntüde birden çok etiketin bulunmasına izin verilmektedir. Her iki yöntemde bir görüntüde birden çok sınıfın olabileceği durumları ele almaktadır. İki yöntemde de öğrenme tabanlı yaklaşımlar kullanılmaktadır. Modeller, genellikle derin öğrenme ağları gibi tekniklerle büyük veri setlerinde eğitilmekte ve ardından yeni görüntülerde sınıflandırma veya nesne tespiti yapabilmektedirler.

Bu benzerliklerin yanı sıra her iki yöntem arasında önemli farklar da bulunmaktadır. Nesne tespiti, nesnelerin konumlarını da belirlemeyi hedeflerken, çok etiketli görüntü sınıflama sadece sınıfları tanımayı amaçlar yani sınıfların hangi nesneleri temsil ettiğini belirlemeye odaklanmaktadır. Çok etiketli görüntü sınıflamasında çıktılar, görüntüde bulunan tüm sınıfların olasılıklarını içermektedir, ancak sınıflara ait konum bilgisi bulunmamaktadır.

Nesne tespitinde ise çıktılarda sınıfların yanı sıra, her sınıfın görüntü içindeki konumlarını da bulunmaktadır. Bu genellikle sınırlayıcı kutuların (bounding boxes) koordinatlarını içermektedir. Çok etiketli görüntü sınıflamasında genellikle ESA gibi derin öğrenme ağları kullanılmaktadır. Nesne tespitinde ise hem sınıf tahmini hem de konum tahmini yapabilen bölge tabanlı ESA (R-CNN), daha hızlı R-CNN, YOLO gibi mimariler kullanılmaktadır. Çok etiketli görüntü sınıflamasında sadece sınıf etiketleri kullanılırken

nesne tespitinde hem sınıf etiketleri hem de nesnelerin konumlarını içeren etiketlemeler kullanılmaktadır.

Oluşturulan KralSakir veri seti ile iki yöntem denenmiştir. Çok etiketli görüntü sınıflandırma ile uçtan uca eğitilen bir mimari oluşturulmuş ve Tablo 6 ve Tablo 7'de yer alan sonuçlar elde edilmiştir.

Tablo 6. Eğitim Seti Sonuçları

Metrik	**Değer**	**Açıklama**
Ortalama Kayıp	0.0097	Düşük kayıp, iyi öğrenilmiş bir modeli gösterir.
Doğruluk	99.87%	Yüksek doğruluk, eğitim setinde iyi performans.
Kesinlik	99.98%	Pozitif tahminlerin doğruluğu.
Duyarlılık	99.73%	Gerçek pozitiflerin doğru tahmin edilmesi.
AUC	99.97%	Yüksek sınıflandırma performansı.
F1 Skor	99.85%	Kesinlik ve duyarlılık dengeli, yüksek performans.

Tablo 7. Doğrulama Seti Sonuçları

Metrik	**Değer**	**Açıklama**
Ortalama Kayıp	0.0659	Eğitim setine kıyasla daha yüksek kayıp.
Doğruluk	98.76%	Yeni verilere genelleme kabiliyeti.
Kesinlik	99.25%	Pozitif tahminlerin doğruluğu.
Duyarlılık	98.11%	Gerçek pozitiflerin doğru tahmin edilmesi.
AUC	99.42%	Eğitim setine benzer performans.
F1 Skor	98.66%	Kesinlik ve duyarlılık dengeli, genelleme kabiliyeti.

Nesne tespiti ile oluşturulan YOLOv8 mimarisinde ise Tablo 8'te yer alan sonuçlar elde edilmiştir.

Tablo 8. Sonuçları Alınan Metrikler

Metrik	**Değer**	**Açıklama**
Kesinlik	%90	20 epokta %90'a ulaşmıştır.
Duyarlılık	%92	%92 seviyesine yükselmiş.
Eğitim Kaybı	%44	Modelin genelde iyi bir şekilde öğrenme sağladığını gösterir.
Doğrulama Kaybı	%4	Düşük kayıp, iyi öğrenilmiş bir modeli gösterir.
mAP50	%90	Modelin tüm sınıfları başarılı bir şekilde tahmin etme yeteneği.
F1 Skor	%91	Sınıf bazında yüksek F1 skor değerleri.

Her iki yöntemin de yüksek doğruluk, kesinlik, duyarlılık ve düşük kayıp değerleri ile başarılı oldukları söylenebilmektedir. İki yöntemde de sınıflar arasında dengeli bir performans gözlemlenmiş; eğitim setinde iyi bir öğrenme sergiledikleri ve doğrulama setinde de genelleme yeteneğine sahip oldukları görülmektedir.

5. SONUÇ ve ÖNERİLER

Çizgi karakterlerin otomatik olarak algılanması ve tanınması yöntemleri son yıllarda büyük bir ilgi görmektedir. Buna paralel olarak makine öğrenmesi alanındaki gelişmeler de araştırmacılar tarafından yoğun bir şekilde kullanılmaktadır. Özellikle de ESA'lar kullanılarak çizgi karakterlerin tespit edilmesi için doğru ve verimli yöntemlerin geliştirilmesi mümkün hale gelmiştir. Bu nedenle bu çalışmada zorlu problemlerden biri olan çizgi karakter tanıma makine öğrenmesi yöntemleri kullanılarak ele alınmıştır.

Çalışmada ilk olarak üzerinde çalışılması planlanan çizgi karakterler belirlenmiştir. Çalışmanın etik kurallara uyması amacıyla karakterlerin yaratıcısı Varol Yaşaroğlu ile iletişime geçilerek gerekli yasal izinler alınmıştır (Yaşaroğlu, 2023). Kral Şakir çizgi filminin dört karakteri olan Kral Şakir, Remzi, Necati ve Canan ile çalışma yapılmıştır.

Literatür araştırması yapıldığında karşılaşılan her bir çalışma farklı metodolojiler ve uygulamalar kullanarak bu alandaki mevcut problemlere çözümler getirmeye çalışmıştır. Bu çalışmalar arasında çizgi karakter tespiti konusunda da çeşitli yaklaşım ve tekniklerin kullanıldığı araştırmalara rastlanılmıştır. Araştırmalarda kullanılan her bir yöntem, çizgi karakterlerin özelliklerini ve görünümlerini dikkate alınarak tasarlanmasından dolayı farklı sonuçların elde edildiği gözlemlenmiştir.

Bu çalışmada, karakterlerin dijital fotoğrafları gerekli düzenlemeler yapılarak toplanmış ve literatürde var olmayan ulusal düzeyde yeni bir çizgi karakter veri seti manuel olarak hazırlanmıştır. Elde edilen bu veri setinden yola çıkarak yeni bir uçtan uca model tasarlanmış ve veri setine YOLOv8 ile nesne tespiti yapılmıştır. Oluşturulan KralSakir veri seti ile her modelde çalışabileceği kanıtlanmıştır.

Makine öğrenmesi teknikleriyle derin öğrenme kullanılarak ESA tabanlı uçtan uca oluşturulan modelde çok etiketli görüntü sınıflama yöntemi kullanılmış ve hem sınıf hem de konum tespiti yapılan modelde YOLOv8 mimarisi kullanılarak transfer öğrenme yapılmıştır. Kullanılan iki model oluşturulan KralSakir veri seti üzerinden eğitilmiştir.

Yapılan deneysel çalışmalarda çok etiketli görüntü sınıflama yönteminde eğitim seti performans sonuçları; ortalama kayıp değeri 0.0097, doğruluk değeri 0.9987 (%99.87), kesinlik değeri 0.9998 (99.98%), duyarlılık değeri 0.9973 (99.73%), AUC değeri 0.9997 (99.97%), F1 skor değeri 0.9985 (99.85%) olarak elde edilirken, doğrulama seti performans sonuçları ise ortalama kayıp değeri 0.0659, doğruluk değeri 0.9876 (98.76%), kesinlik değeri 0.9925 (99.25%), duyarlılık değeri 0.9811 (98.11%), AUC değeri 0.9942 (99.42%), F1 skor değeri 0.9866 (98.66%) olarak elde edilmiştir. Bu sonuçlara göre model eğitim setinde yüksek performans göstermektedir ve doğrulama setinde de iyi bir genellemeye sahip olduğu görülmektedir.

Yapılan deneysel çalışmalarda YOLOv8 mimarisi kullanılarak yapılan nesne tespiti yönteminde performans sonuçları; eğitim seti kayıp değeri %44, doğrulama seti kayıp değeri %4'e göre kesinlik değeri %90, duyarlılık değeri %92, F1 skor değeri %91, mAP50 değeri %90 olarak elde edilirken sınıflar baz alınarak hesaplanan kesinlik - duyarlılık grafik sonuçlarına göre kral_sakir sınıfı için %98, remzi sınıfı için %95, necati sınıfı için %96, canan sınıfı içinse %91 olarak değerler elde edilmiştir.

Elde edilen bu rakamsal sonuçlara göre çalışmada kullanılan metriklerin genel olarak başarılı olduğu görülmektedir. Metriklerden kesinlik ve duyarlılığın yüksek değerlerinin, olması modelin doğru tahminlerde bulunduğunu ve sınıf kaçırmadığını, düşük kayıp değerleri ve yüksek F1 skoru ise genel olarak iyi bir performans gösterdiğini ifade etmektedir. Ayrıca mAP50 değerinin yüksek olması, modelin nesneleri belirleme yeteneğinin oldukça yüksek olduğunu göstermektedir. Önerilen veri seti üzerinde yapılan çalışmalar sonucunda çizgi karakter tespiti yöntemlerinden çok sınıflı sınıflandırma ve nesne tanıma yöntemlerinde başarılı bir şekilde çalıştığı gözlemlenmiştir.

Ayrıca bu tezde yapılan deneysel çalışmalara göre oluşturulan KralSakir veri setinin çizgi karakterlerle ilgili çeşitli görevlere uygulanabilir olduğu kanıtlanmıştır.

Son olarak bu çalışma referans alınarak bir sonraki çalışmalarda araştırmacılar farklı yöntemler ve yazılımlar kullanılarak oluşturulan mimariler geliştirilebilirler ayrıca, oluşturulan KralSakir veri seti ile transfer öğrenme uygulamaları araştırmanın çalışma alanını genişletebilirler. Bu çalışmadaki veri setinde yapılan manuel etiketleme, sonraki

çalışmalar için teknolojik yöntemlerle otomatik olarak yeniden etiketlenebilir. Veri setindeki görsellerin sayısı çoğaltılabilir ve sınıf sayısına ekleme yapılabilir.

KAYNAKLAR

Aizawa, K., Fujimoto, A., Otsubo, A., Ogawa, T. M., Matsui, Y., Tsubota, K., vd. (2020). Building a manga dataset "Manga109" with annotations for multimedia applications. *IEEE MultiMedia, 27(2),* 8-18.

Barkana, B. D., Sarıçiçek, İ., ve Yıldırım, B. (2017). Performance analysis of descriptive statistical features in retinal vessel segmentation via fuzzy logic, ANN, SVM, and classifier fusion. *Knowledge-Based Systems, 118,* 165-176.

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

Canu, S. (2023). *Flatten and Dense layers*. 06 10, 2023 tarihinde https://pysource.com/2022/10/07/flatten-and-dense-layers-computer-vision-with-keras-p-6/ adresinden alındı.

Cengiz, E. (2020). *Yapay Zeka Tabanlı Drone Optimizasyonu* (Yayın no. 641893). [Yüksek Lisans Tezi, Gazi Üniversitesi]. https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp

Chaurasia, A., ve Jocher, G. (2023). *Ultralytics YOLOv8 Docs*. 10 01, 2023 tarihinde https://docs.ultralytics.com/#yolo-a-brief-history adresinden alındı.

Chollet, F. (2021). *Python ile Derin Öğrenme.* Ankara: Buzdağı Yayınevi.

Chu, W. T., ve Li, W. W. (2019). Manga face detection based on deep neural networks fusing global and local information. *Pattern Recognition, 86,* 62-72.

Team, S. (2023). *CNN: Step 3 - Flattening*. 06 10, 2023 tarihinde https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening adresinden alındı.

Dutta, A., ve Biswas, S. (2019). CNN based extraction of panels/characters from bengali comic book page images. *In 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW) (Vol. 1), IEEE.*, 38-43.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters 27(8), Elsevier*, 861–874.

Goodfellow, I., Bengio, Y., ve Courville, A. (2015). *Deep learning.* Cambridge: MIT press.

Grafi2000. (2023). *Grafi2000/Kral Şakir*. 12 01, 2023 tarihinde https://grafi2000.com/markalar/kral-sakir/ adresinden alındı.

Haibing, W., ve Gu, X. (2015). Towards dropout training for convolutional neural networks. *Neural Networks, Volume 71,Elsevier*, 1-10.

Han, D., Liu, Q., ve Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. *Expert Systems with Applications, 95*, 43-56.

Hao, W., Yizhou, W., Yaqin, L., ve Zhili, S. (2020). The Role of Activation Function in CNN. *2nd International Conference on Information Technology and Computer Application (ITCA)*, 429-430.

Haykin, S. (1998). *Neural networks: a comprehensive foundation.* Hamilton, Ontario, Canada: Prentice Hall.

Heaton, J. (2015). *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks.* Chesterfield: Heaton Research, Inc.

Hinton, G. E., Krizhevsky, A., Sutskever, I., ve Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Ho, H. N., Rigaud, C., Burie, J. C., ve Ogier, J. M. (2013). Detecting recurring deformable objects: an approximate graph matching method for detecting characters in comics books. *In International Workshop on Graphics Recognition, Springer*, 122-134.

Ho, H. N., Rigaud, C., Burie, J. C., ve Ogier, J. M. (2014). Redundant structure detection in attributed adjacen y graphs for character detection in comics books. *In 10th IAPR International Workshop on Graphics Recognition*.

Huo, J., Li, W., Shi, Y., Gao, Y., ve Yin, H. (2017). Webcaricature: a benchmark for caricature recognition. *arXiv preprint arXiv:1703.03230. doi: 10.48550/arXiv.1703.03230*.

Ismail, A., Ahmad, S. A., Soh, A. C., Hassan, K., ve Harith, H. H. (2019). Improving Convolutional Neural Network (CNN) Architecture (miniVGGNet) with Batch Normalization and Learning Rate Decay Factor for Image Classification. *The International Journal of Integrated Engineering, Vol:11, No:4*, 5159.

Iwata, M., Ito, A., ve Kise, K. (2014). A study to achieve manga character retrieval method for manga images. *In 2014 11th IAPR International Workshop on Document Analysis Systems, IEEE*, 309-313.

James, G., Witten, D., Hastie, T., ve Tibshirani, R. (2013). *An Introduction to Statistical Learning.* New York: Springer.

Jha, D., Yazidi, A., Riegler, M. A., Johansen, D., Johansen, H. D., ve Halvorsen, P. (2020). LightLayers: Parameter Efficient Dense and Convolutional Layers for Image Classification. *PDCAT 2020, Springer International Publishing*, 285-296.

Jocher, G. (2023). *Ultralytics YOLOv8 Docs*. 12 01, 2023 tarihinde https://docs.ultralytics.com/tr/modes/train/ adresinden alındı.

Khan, F. S., Anwer, R. M., Van De Weijer, J., Bagdanov, A. D., Vanrell, M., ve Lopez, A. M. (2012). Color attributes for object detection. *In 2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3306-3313.

Kim, H., Lee, E.-C., Seo , Y., Im, D.-H., ve Lee, I.-K. (2020). Character Detection in Animated Movies Using Multi-Style Adaptation and Visual Attention. *IEEE Transactions on Multimedia, 23*, 1990-2004.

Kim, K., Park, S., Lee, J., Chung, S., ve Lee, J. (2021). Animeceleb: Large-scale animation celebfaces dataset via controllable 3d synthetic models. *CoRR, vol. abs/2111.07640*.

Kingma, D. P., ve Ba, J. L. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kochenderfer, M. J., ve Wheeler, T. A. (2019). *Algorithms for Optimization.* England: The MIT Press.

Kotlicki, B. (2021). *HaarCNN: Detection and Recognition of Comic Strip Characters with Deep Learning and Cascade Classifiers.* 07 01, 2023 tarihinde http://resolver.tudelft.nl/uuid:8d9fc510-8b5a-4853-b849-2b608bd5fe11 adresinden alındı.

Kotsiantis, S. B., Zaharakis, I., ve Pintelas, P. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Emerging artificial intelligence applications in computer engineering*, 3-24.

Krizhevsky, A., Sutskever, I., ve Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097–1105.

Kulkarni, A., Batarseh, F. A., ve Chong, D. (2020). Foundations of Data Imbalance and Solutions for a Data Democracy. *Data democracy. Academic Press*, 83-106.

Larsen-Freeman, D. (2013). Transfer of Learning Transformed. *Language Learning,63*, 107-129.

Le, T. N., Luqman, M. M., Burie, J. C., ve Ogier, J. (2015). A comic retrieval system based on multilayer graph representation and graph mining. *In International*

Workshop on Graph-Based Representations in Pattern Recognition, Springer, 355-364.

Le, T. N., Luqman, M. M., Burie, J. C., ve Ogier, J. (2015). Content-based comic retrieval using multilayer graph representation and frequent graph mining. *In 2015 13th International Conference on Document Analysis and Recognition (ICDAR), IEEE*, 761-765.

LeCun, Y., Bengio, Y., ve Hinton, G. (2015). Deeplearning. *Nature,521(7553)*, 436-44.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R., Hubbard, W., ve Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural computation, 1(4)*, 541-551.

LeCun, Y., Bottou, L., Bengio, Y., ve Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11)*, 2278–2324.

Lee, K., Sung, S. H., Kim, D.-h., ve Park, S.-h. (2019). Verification of Normalization Effects Through. *In 2019 International Conference on Multimedia Analysis and Pattern Recognition (MAPR),IEEE*, 1-5.

Liu, Q., ve Wu, Y. (2011). Supervised learning. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, 63-132.

Loshchilov, I., ve Hutter, F. (2019). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lu, D., ve Weng, Q. (2007). A survey of image classification methods and techniques for improvingclassification performance. *International journal of Remote sensing, 28(5)*, 823-870.

Martin, S. (2023). *What's the Difference Between a CNN and an RNN?* 07 28, 2023 tarihinde https://www.edge-ai-vision.com/2018/09/whats-the-difference-between-a-cnn-and-an-rnn/ adresinden alındı.

Mashood Nasir, I., Attique Khan, M., Alhaisoni, M., Saba, T., Rehman, A., ve Iqbal, T. (2020). A hybrid deep learning architecture for the classification of superhero fashion products: An application for medical-tech classification. *Computer Modeling in Engineering ve Sciences, 124(3)*, 1017-1033.

MathWorks. (2023). What is Convolution? 08 12, 2023 tarihinde https://www.mathworks.com/content/mathworks/www/en/discovery/convolution.html adresinden alındı.

Matsui, Y., Ito, K., Aramaki, Y., Fujimoto, A., Ogawa, T., Yamasaki, T., ve Aizawa, K. (2017). Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications, 76(20)*, 21811-21838.

McCulloch, W. S., ve Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5*, 115–133.

Minsky, M. L., ve Papert, S. A. (1969). An introduction to computational geometry. *Cambridge tiass., HIT, 479(480)*, 104.

Mitchell, T. M. (1997). *Machine Learning.* Portland: McGraw-Hill Science/Engineering/Math ISBN: 0070428077.

Murugan, P., ve Durairaj, S. (2017). Regularization and optimization strategies in deep convolutional neural network. *arXiv preprint arXiv:1712.04711*.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. B, 4. DOI 10.20544/HORIZONS.B.04.1.17.P05*, 5162.

Nebauer, C. (1998). Evaluation of convolutional neural networks for visual recognition. *IEEE Transactions on Neural Networks, 9 (4)*, 685- 696.

Ng, A. Y. (2004). Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning (p. 78). USA: Stanford Üniversity.

Nguyen, N. V., Rigaud , C., ve Burie, J. C. (2018). Digital comics image indexing based on deep learning. *Journal of Imaging, 4(7)*, 89.

Nguyen, N. V., Rigaud, C., ve Burie, J. C. (2017). Comic characters detection using deep learning. *In 2017 14th IAPR international conference on document analysis and recognition (ICDAR) (Vol. 3), IEEE*, 41-46.

Nguyen, N. V., Rigaud, C., ve Burie, J. C. (2019). Comic MTL: optimized multi-task learning for comic book image analysis. *International Journal on Document Analysis and Recognition (IJDAR), 22(3)*, 265-284.

Nir, O., Rapoport, G., ve Shamir, A. (2022). CAST: Character labeling in Animation using Self-supervision by Tracking. *In Computer Graphics Forum. Vol. 41, No. 2*, 135-145.

Özkan, M., ve Kar, G. (2022). Türkçe Dilinde Yazılan Bilimsel Metinlerin Derin Öğrenme Tekniği Uygulanarak Çoklu Sınıflandırılması. *Mühendislik Bilimleri ve Tasarım Dergisi, 10(2)*, 504-519.

Özkesici, M. Y., ve Yılmaz, S. (2021). Oral Ve Maksillofasiyal Radyoloji'de Yapay Zeka. *Sağlık Bilimleri Dergisi , 30 (3), DOI: 10.34108/eujhs.1040476*, 346-351.

Öztemel, E. (2003). *Yapay Sinir Ağları.* İstanbul: Papatya Yayıncılık.

Park, S., ve Kwak, N. (2017). Analysis on the Dropout Effect in Convolutional. *In Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, Revised Selected Papers, Part II 13*, 189-204.

Patterson, J., ve Gibson, A. (2017). *Deep Learning.* United States of America: O'Reilly Media, Inc.

Peixoto, F. (2023). *A Simple overview of Multilayer Perceptron(MLP)*. 08 20, 2023 tarihinde https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/ adresinden alındı.

Prathammodi. (2023). *Convolutional Neural Networks for Dummies*. 07 28, 2023 tarihinde https://medium.com/@prathammodi001/convolutional-neural-networks-for-dummies-a-step-by-step-cnn-tutorial-e68f464d608f adresinden alındı.

Qin, X., Zhou, Y., He, Z., Wang, Y., ve Tang, Z. (2017). A faster R-CNN based method for comic characters face detection. *In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) (Vol. 1), IEEE.*, 1074-1080.

Quattoni, A., Collins, M., ve Darrell, T. (2008). Transfer learning for image classification with sparse prototype representations. *In 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, 1-8.

Ranzato, M., Huang, F. J., Boureau, Y.-L., ve LeCun, Y. (2007). Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. *In 2007 IEEE conference on computer vision and pattern recognition*, 1-8.

Rawat, W., ve Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation, 29(9)*, 2352-2449.

Read, J., ve Perez-Cruz, F. (2014). Deep Learning for Multi-label Classification. *arXiv preprint arXiv:1502.05988*.

Redmon, J., Divvala, S., Girshick, R., ve Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 779-788.

Ribani, R., ve Marengoni, M. (2019). A Survey of Transfer Learning for Convolutional Neural Networks. *In 2019 32nd SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T), IEEE.*, 47-57.

Rigaud, C., Burie, J.-C., Ogier, J.-M., ve Karatzas, D. (2014). Color descriptor for content-based drawing retrieval. *11th IAPR International Workshop on Document Analysis*, 267-271.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review, 65(6)*, 386-408.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747.*, 1-14.

Rumelhart, D. E., Hinton, G. E., ve Williams, R. J. (1986). Learning representations by back-propagating errors. *Natura, Vol,323*, 533-536.

Samuel, A., Shannon, C., ve Simon, H. (1996). Artificial intelligence in computer-aided design: Progress and prognosis. *Computer-Aided Design, Vol. 26, No. 3, Elsevier Science Ltd Printed in Great Britain*, 153-154.

Sarkar, D., Bali, R., ve Ghosh, T. (2018). *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras.* Birmingham: Packt Publishing, ISBN 978-1-78883-130-7.

Sharma, N., Sharma, R., ve Neeru, J. (2021). Machine Learning and Deep Learning Applications-A Vision. *Global Transitions Proceedings 2(1)*, 24-28.

Smith, L. N. (2017). Cyclical Learning Rates for Training Neural Networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE*, 464-472.

Ssawla. (2023). *Partial AUC Scores: A Better Metric for Binary Classification*. 10 3, 2023 tarihinde https://www.analyticsvidhya.com/blog/2023/02/partial-auc-scores-a-better-metric-for-binary-classification/ adresinden alındı.

Sun, W., ve Kise, K. (2010). Similar partial copy detection of line drawings using a cascade classifier and feature matching. *In International Workshop on Computational Forensics*, 126-137.

Sun, W., ve Kise, K. (2013). Detection of exact and similar partial copies for copyright protection of manga. *International Journal on Document Analysis and Recognition (IJDAR), 16(4)*, 331-349.

Sun, W., Burie, J. C., Ogier, J. M., ve Kise, K. (2013). Specific comic character detection using local feature matching. *In 2013 12th International Conference on Document Analysis and Recognition. IEEE*, 275-279.

Sunitha, R., Sharmila, B., ve Srinivasan, K. (2019). Feature Extraction Methods using Image Processing Techniques. *International Journal of Engineering and Advanced Technology (IJEAT), 8(4)*.

Takayama, K., Johan, H., ve Nishita, T. (2012). Face detection and face recognition of cartoon characters using feature extraction. *In Image, Electronics and Visual Computing Workshop*, 48.

Tang, A., Tam, R., Cadrin-Chenevert, A., Guest, W., Chong, J., Barfett, J., . . . Geis, R. (2018). Canadian Association of Radiologists White Paper on Artificial Intelligence in Radiology. *Canadian Association of Radiologists Journal 69*, 120-135.

Thakkar, V., ve Tewary, S. (2018). Batch Normalization in Convolutional Neural Networks – A comparative study with CIFAR-10 data. *In 2018 fifth international conference on emerging applications of information technology (EAIT), IEEE.*, 1-5.

Van Veen, F., ve Leijnen Stefan. (2023). *Neural Network Zoo*. 09 12, 2023 tarihinde https://www.asimovinstitute.org/neural-network-zoo/ adresinden alındı.

Wikipedia (2023). *Varol Yaşaroğlu*. 11 2, 2023 tarihinde https://tr.wikipedia.org/wiki/Varol_Ya%C5%9Faro%C4%9Flu adresinden alındı.

Viola, P., ve Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision, 57(2)*, 137-154.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., ve Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust., Speech, Signal Process., vol. 37, no. 3*, 328–339.

Xu, S., Wang, J., Shou, W., Sadick, A.-M., ve Wang, X. (2021). Computer vision techniques in construction: a critical review. *Archives of Computational Methods in Engineering, 28*, 3383-3397.

Yanagisawa, H., Yamashita, T., ve Watanabe, H. (2018). A study on object detection method from manga images using CNN. *In 2018 International Workshop on Advanced Image Technology (IWAIT)*, 1-4.

Yaşaroğlu, V. (2023). *Grafi2000*. 10 3, 2023 tarihinde https://grafi2000.com/markalar/kral-sakir/ adresinden alındı.

Yılmaz, A., ve Kaya, U. (2021). *Derin Öğrenme.* İstanbul: Kodlab Yayınevi.

Yu , T., ve Zhu, H. (2020). Hyper-Parameter Optimization: A Review of Algorithms. *arXiv preprint arXiv:2003.05689.*, 1-56.

Yu, J., ve Seah, H.-S. (2011). Fuzzy diffusion distance learning for cartoon similarity estimation. *Journal of computer science and technology, 26(2),* 203-216.

Zhang, A., Lipton, Z. C., Li, M., ve Smola, A. J. (2023). *Dive into deep learning.* Cambridge University Press, arXiv preprint arXiv: 2106.11342.

Zhang, W., Tanida, J., Itoh, K., ve Ichioka, Y. (1988). Shift-invariant pattern recognition neural network and its optical architecture. *In Proceedings of Annual Conference of the Japan Society of Applied Physics (Vol. 564)*, 734.

Zhao, Z.-Q., Zheng, P., Xu, S.-T., ve Wu, X. (2019). Object Detection With Deep Learning: A Review. *IEEE transactions on neural networks and learning systems, 30(11)*, 3212-3232.

Zou, Z., Chen, K., Shi, Z., Guo, Y., ve Ye, J. (2023). Object Detection in 20 Years: A Survey. *Proceedings of the IEEE, 111, (3)*, 257-276.

Zuo, C., Qian, J., Feng, S., Yin, W., Li, Y., Fan, P., vd. (2022). Deep learning in optical metrology: a review. *Light: Science ve Applications, 11(1), 39*, 1-54.

ÖZGEÇMİŞ

Candan TÜMER

Eğitim

Yüksek Lisans	2014	Gazi Üniversitesi, Bilişim Enstitüsü Bilgisayar Eğitimi Anabilim Dalı
Lisans	2010	Mersin Üniversitesi, UTİYO Bilgisayar Teknolojisi ve Bilişim Sistemleri

İş/İstihdam (Varsa)

2023 -	Öğretim Görevlisi. Gazi Üniversitesi, Bilişim Enstitüsü
2015 – 23	Öğretim Görevlisi. Ardahan Üniversitesi Teknik Bilimler MYO
2011- 15	Bilgisayar İşletmeni. Ardahan Üniversitesi Rektörlük Ofisi
2010- 11	Bilgisayar Öğretmeni. Keçiören Belediyesi, Keçmek Kursları

Yayınlar ve Diğer Bilimsel/Sanatsal Faaliyetler

Tunalı V., Tümer C. (2021). Analysis of Book-borrowing Network using Complex Network Analysis. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi, 10(1), page 133-142.*

Tümer C., Tüfekci A. (2017). Türkiye'de Bilgisayar ve Öğretim Teknolojileri Eğitimi (BÖTE) Alanında Tamamlanmış Yüksek Lisans ve Doktora Tezlerindeki Güncel Eğilimler (2005-2014). *4th International Conference on Humanities and Educational Research Istanbul 8 – 10, 2017, Istanbul, Turkey Hosted by Maltepe University, page 41.*

Tümer C., Tüfekci A. (2011). Mobil İletişim ve Mobil Öğrenme Süreçleri, *Türkiye Bilişim Derneği Dergisi, 28 Ekim 2011.*

Tümer C., Tüfekci A. (2011). Sınıf İçi Uygulamalarda Akıllı Tahta Kullanılabilirliği Üzerine Bir Çalışma, *Yıldız Teknik Üniversitesi Sigma Fen ve Müh. Bililmleri Dergisi.*