T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME TEMELLİ TÜRK İŞARET DİLİ HARFLERİNİN TANINMASI

Fatih BANKUR

Yüksek Lisans Tezi

ADLİ BİLİŞİM MÜHENDİSLİĞİ ANABİLİM DALI

TEMMUZ 2021

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Adli Bilişim Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

DERİN ÖĞRENME TEMELLİ TÜRK İŞARET DİLİ HARFLERİNİN TANINMASI

Tez Yazarı
Fatih BANKUR

Danışman
Dr. Öğr. Üyesi Mustafa KAYA

TEMMUZ 2021
ELAZIĞ

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Adli Bilişim Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Başlığı:	Derin Öğrenme Temelli Türk İşaret Dili Harflerinin Tanınması
Yazarı:	Fatih BANKUR
İlk Teslim Tarihi:	27.06.2021
Savunma Tarihi:	27.07.2021

TEZ ONAYI

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına göre hazırlanan bu tez aşağıda imzaları bulunan jüri üyeleri tarafından değerlendirilmiş ve akademik dinleyicilere açık yapılan savunma sonucunda OYBİRLİĞİ ile kabul edilmiştir.

	İmza	
Danışman:	Dr. Öğr. Üyesi Mustafa KAYA Fırat Üniversitesi Teknoloji Fakültesi Adli Bilişim Mühendisliği	Onayladım
Başkan:	Dr. Öğr. Üyesi Erkan DUMAN Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü	Onayladım
Üye:	Dr. Öğr. Üyesi Aytuğ BOYACI Milli Savunma Üniversitesi Siber Güvenlik Bölümü	Onayladım

Bu tez, Enstitü Yönetim Kurulunun ...../...../20..... tarihli toplantısında tescillenmiştir.

İmza

Doç. Dr. Kürşat Esat ALYAMAÇ
Enstitü Müdürü

BEYAN

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım "Derin Öğrenme Temelli Türk İşaret Dili Harflerinin Tanınması " Başlıklı Yüksek Lisans Tezimin içindeki bütün bilgilerin doğru olduğunu, bilgilerin üretilmesi ve sunulmasında bilimsel etik kurallarına uygun davrandığımı, kullandığım bütün kaynakları atıf yaparak belirttiğimi, maddi ve manevi desteği olan tüm kurum/kuruluş ve kişileri belirttiğimi, burada sunduğum veri ve bilgileri unvan almak amacıyla daha önce hiçbir şekilde kullanmadığımı beyan ederim.

27.07.2021

Fatih BANKUR

ÖNSÖZ

Konuşma engeli olan veya işitme engelli insanların sosyal hayatta var olmalarına olanak sağlayan etmenlerin başında gelen iletişim kurma yeteneği, konuşabilen insanların iletişim kurma yeteneklerine göre daha kısıtlıdır. Bu sebepten işaret dili kullanan insanlarla işaret dilini hiç bilmeyen insanlar arasındaki engelleri kaldırmak ve iletişimi kolaylaştırmak adına ilk etapta işaret dilinde bulunan harflerin tanınmasına yönelik bir derin öğrenme modelinin geliştirilmesinin faydalı olacağı düşünülmektedir.

Bu çalışmanın tamamlanmasında yardımlarını esirgemeyen danışmanım Sayın Dr. Mustafa KAYA' ya, Fırat Üniversitesi Adli Bilişim Mühendisliğinde görevli hocalarıma, sevgili eşim Nisa ve biricik kızım İnci'ye, Annem Emine ve Babam Aziz BANKUR' a, Kardeşim Çağrı BANKUR' a, Emre ve Büşra YILDIRIM'a ve manevi olarak her zaman yanımda hissettiğim Şehit Kardeşim Cüneyt BANKUR' a şükranlarımı sunarım.

Fatih BANKUR
ELAZIĞ, 2021

İÇİNDEKİLER

Sayfa

ÖNSÖZ iiv
İÇİNDEKİLER v
ÖZET viii
ABSTRACT ix
ŞEKİLLER LİSTESİ x
TABLOLAR LİSTESİ xi
SİMGELER VE KISALTMALAR xii

1.GİRİŞ 1

2.TÜRK İŞARET DİLİ 4

2.1. Türk İşaret Dilinin Tarihi Gelişimi 4
2.2. Türkiye'de Türk İşaret Dili Kullanıcıları 4
2.3. Türk İşaret Dili Alfabesi ve Rakamları 5

3.DERİN ÖĞRENMENİN GELİŞİMİ 7

3.1 Yapay Zekâ 7
3.2 Makine Öğrenmesi 8
3.2.1 Denetimli Öğrenme 8
3.2.2 Denetimsiz Öğrenme 9
3.2.3. Denetimli ve Denetimsiz Öğrenme Karşılaştırılması 9
3.3.Yapay Sinir Ağları 10
3.3.1. Tek Katmanlı Algılayıcılar (Preseptron) 11
3.3.2. Çok Katmanlı Algılayıcılar 12
3.4 Derin Öğrenme 13
3.5.Evrişimsel Sinir Ağları (CNN) 14
3.5.1. Girdi Katmanı 14
3.5.2. Evrişim Katmanı 14
3.5.3. Havuzlama (Pooling) Katmanı 15
3.5.4 Tamamen Bağlı Katman 15

4.NESNE TESPİTİ VE KULLANILAN MODELLER 16

4.1. Nesne Tespiti 16
4.2. İki Aşamalı Nesne Tespit Modelleri 17
4.2.1. Regions with CNN fatures (R-CNN) 17
4.2.2. Fast R-CNN 18
4.2.3. Faster R-CNN 19
4.3. Tek Aşamalı Nesne Tespit Modelleri 20
4.3.1. Single Shot Multibox Detector (SSD) 20
4.3.2. EfficientDet 21
4.3.3. Yolo Modelleri 23

5. VERİ SETİ VE UYGULAMANIN GERÇEKLEŞTİRİLMESİ 16

5.1 Programlama Dili ve Geliştirme Ortamı 29
5.1.1 Python Programlama Dili 29
5.1.2 Google Colab 30

5.2 Veri Setinin Hazırlanması 31
5.2.1 TİD Alfabesi Videoları 32
5.2.2 TİD Vidolarından Framelerin Seçilmesi 32
5.2.3 Resimlerin Etiketlenmesi ve Export İşlemi 33
5.2.4 Veri Çoğaltma 36
5.3 Eğitim Transferi ve Nesne Tanıma Modelinin Model Seçimi 38
5.3.1 Eğitim Transferi 38
5.3.2 SSD Mobilenet 39
5.3.3 EfficienNet 40
5.3.4 YOLOv5s 41
5.3.5 Model seçimi 42
5.4 Uygulamanın Gerçekleştirilmesi 43
5.4.1 Google Colab Ortamının Hazırlanması 44
5.4.2 Modelin Oluşturulması 45
5.4.3 Modelin Eğitimi 46
5.5 Model Değerlendirilmesinde Kullanilan Yöntemler 48
5.5.1 Precision (Kesinlik) 49
5.5.2 Recall (Duyarlılık) 49
5.5.3 mAP (Ortalama Averaj Kesinliği) 49
5.5.4 Modele Ait Veriler 50
6.SONUÇLAR 57
KAYNAKLAR 58
ÖZGEÇMİŞ

ÖZET

Derin Öğrenme Temelli Türk İşaret Dili Harflerinin Tanınması

Fatih BANKUR

Yüksek Lisans Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Adli Bilişim Mühendisliği Anabilim Dalı

Temmuz 2021, Sayfa: xii + 60

İşitme engelli olan insanların duyabilen insanlar kadar rahat bir yaşam sürebilmeleri insanlarla kurabildikleri iletişim ile mümkün olabilmektedir. Bu iletişimi çoğunlukla işaret dilini kullanarak gerçekleştirirler. İşitme engelli insanların kullandıkları işaret dili ne yazık ki duyabilen insanlar tarafından yeterince bilinmemektedir. Bu sebeple işitme engelli insanların günlük hayatlarını kolaylaştırmak adına işaret dili üzerine birçok akademik çalışma yapılmaktadır. Özellikle yapay zekâ alanındaki gelişmelerden sonra makine öğrenmesi ve derin öğrenme alanında İşaret Dili çevrimi ile alakalı olarak çalışmalar hız kazanmıştır.

Bu çalışmada günümüz problemlerinin bilgisayarlar kullanılarak çözümü için büyük faydalar sağlayan derin öğrenme modellerinden biri olan YOLOv5 kullanılarak Türk İşaret Dili Harflerinin hızlı bir şekilde tanınması amaçlanmıştır. Bu amaç doğrultusunda özgün ve kapsamlı bir veri seti oluşturulmuş ve YOLOv5 modeli ile % 92,3 oranında başarılı sonuçlar elde edilmiştir. Alınan bu başarılı sonuçlar neticesinde ilerleyen çalışmalarda Türk İşaret Dili Harflerinin yanı sıra Türk İşaret Dilinde bulunan rakamların, kelime ve sözcüklerin sınıflandırılabileceğinin mümkün olabileceği düşünülmektedir.

Anahtar Kelimeler: Türk İşaret Dili Harfleri, Yapay Zekâ, Derin Öğrenme, Evrişimsel Sinir Ağları,YoloV5

ABSTRACT

Deep Learning Based Turkish Signal Recognition

Fatih BANKUR

Master's Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Digital Forensic Engineering

July 2021, Pages: xii + 60

It is possible for people with hearing impairments to live as comfortably as people who can hear with the communication they can establish. They mostly communicate using sign language. The sign language used by hearing-impaired people is unfortunately not known enough by people who can hear it. For this reason, many academic studies are carried out on sign language in order to facilitate the daily lives of hearing- impaired people. Especially after the developments in the field of artificial intelligence, studies on machine learning and the Sign Language cycle in the field of deep learning have accelerated.

In this study, it is aimed to recognize Turkish Sign Language Letters by using YoloV5, which is one of the deep learning models that provides great benefits for solving today's problems by using computers. For this purpose, an original and comprehensive data set was created and it was observed that 92,3 % successful results were obtained with the YoloV5 model. As a result of this successful study, it is thought that it will be also possible to classify the numbers, words found in Turkish Sign Language as well as Turkish Sign Language Letters.

Keywords: Turkish Sign Language Letters, Artificial Intelligence, Deep Learning, Convolutional NeuralNetworks, YoloV5

ŞEKİLLER LİSTESİ

Sayfa

Şekil 2.1 Ülkemizdeki işitme engellilerin yaşa göre dağılım grafiği [26] 5
Şekil 2.2 Ülkemizdeki dil ve konuşma engellilerinin yaşa göre dağılım grafiği [26] 5
Şekil 2.3 Türk işaret dili alfabesi [27] 6
Şekil 3.1 Yapay zekâ kapsamı 7
Şekil 3.2 Biyolojik sinir hücresi [28] 11
Şekil 3.3 Tek katmanlı algılayıcı [28] 12
Şekil 3.4 Çok katmanlı algılayıcılar [28] 12
Şekil 3.5 Basit sinir ağıyla derin öğrenme sinir ağı karşılaştırması [29] 13
Şekil 3.6 Havuzlama işlemi 15
Şekil 4.1 R-CNN yapısı[32] 17
Şekil 4.2 Fast R-CNN yapısı[33] 18
Şekil 4.3 Modellerin Hız karşılaştırması [34] 20
Şekil 4.4 SSD modelinin genel Yapısı [35] 21
Şekil 4.5 EfficienNet mimarisinin genel görünümü [36] 22
Şekil 4.6 YOLO Mimarisi [37] 24
Şekil 4.7 Mavi Renkli Bağlantılı Kutular 25
Şekil 4.8 YOLOv3 modelinin kullandığı Darknet-53 mimarisi [40]. 26
Şekil 4.9 Tek aşamalı nesne tespiti genel yapısı [41]. 27
Şekil 4.10 YOLOv4 genel yapısı [42]. 28
Şekil 5.1 Yeni Colab Ortamı oluşturma 30
Şekil 5.2 Kullanılacak donanım seçimi 30
Şekil 5.3 Veri Setine Ait Akış Şeması 31
Şekil 5.4 Veri Setindeki Harflerin Sayısı 33
Şekil 5.5 Etiketli Resimlere Ait Görseller 34
Şekil 5.6 Veri Setine Ait bilgiler 35
Şekil 5.7 Export Formatları 35
Şekil 5.8 “M” Harfi Veri artırım Örnekleri 37
Şekil 5.9 SSD Mobilenet Kayıp Değerleri 39
Şekil 5.10 EfficienNet D-0 Kayıp Değerleri 40
Şekil 5.11 YOLOv5s Kayıp Değerleri 41
Şekil 5.12 Uygulamanın Akış Şeması 43
Şekil 5.13 Uygulama için Colabda kullanılan donanımlar 44

Şekil 5.14 Hazırlanan Colab Ortamı 44
Şekil 5.15 Model Katman Bilgileri 45
Şekil 5.16 Değiştirilen Argümanlar 46
Şekil 5.17 Model Eğitiminin Tamamlanması 47
Şekil 5.18 Karmaşıklık Matrisi Değerleri 50
Şekil 5.19 Hesaplanan Değerler 51
Şekil 5.20 Precision (Kesinlik) Grafiği 52
Şekil 5.21 Recall (Duyarlılık) Grafiği 52
Şekil 5.22 Precision (Kesinlik)-Recall (Duyarlılık) Grafiği 53
Şekil 5.23 mAP 0.5 ve mAP 0.5:0.95 grafiği 53
Şekil 5.24 "D" ve "J" Harflerine Ait Görseller 54
Şekil 5.25 "H" ve "A" Harflerine Ait Görseller 55
Şekil 5.26 "K" Harflerine Ait Görseller 55
Şekil 5.28 "Z" Harflerine Ait Görseller 56
Şekil 5.29 "M" ve "N" Harflerine Ait Görseller 56

TABLOLAR LİSTESİ

Sayfa

Tablo 3.1 Denetimli-denetimsiz öğrenme karşılaştırılması 10

Tablo 5.1 Veri seti için kullanılacak sistemin özellikleri.....32

Tablo 5.2 Veri setine ait özellikler 37

Tablo 5.3 Hazırlanan Modellere Ait Veriler.....42

Tablo 5.4 Karmaşıklık Matrisi.....48

SİMGELER VE KISALTMALAR

Kısaltmalar

TİD	: Türk İşaret Dili
TÜİK	: Türkiye İstatistik Kurumu
TDK	: Türk Dil Kurumu
YSA	: Yapay Sinir Ağları
DVM	: Destek Vektör Makinesi
CNN	: Evrişimsel Sinir Ağları
RCNN	: Regions with CNN Fatures
SDD	: Single Shor Multibox Interface
YOLO	: You Only Look Once

1. GİRİŞ

Tüm canlılar gibi iletişim kurmak insanlar için de vazgeçilmez bir olgudur. İnsanlar sosyal yaşamda var olmayı ve toplumun bir parçası olmayı arzularlar. İletişim kurmak her ne kadar evrensel olsa da bu iletişimi gerçekleştirmek insanların kullanmış oldukları farklı dillerin bilinmesi ve kullanılmasıyla gerçekleşir. Farklı dili konuşan insanların iletişiminin ne kadar zor olduğu göz önünde bulundurulduğunda aynı toplum içerisinde yaşayan işitme engellilerin aynı toplumda yaşamalarına rağmen işaret dilini kullanamayan insanlarla iletişiminin ne kadar zor olacağı aşikârdır.

İşitme engelliler sosyal hayatta genellikle vücut dilini kullanarak ya da yazarak iletişim halinde olurlar. Duyabilen insanlara göre sesleri algılayamayan işitme engelliler bu sebepten yazılı metinlerin tam olarak anlaşılmasında da zorluk yaşamaktadırlar. İşaret dilleri kişiden kişiye değişmeyen ve yoruma açık olmayan bir iletişim şekli olması nedeniyle işitme engelliler tarafından çoğunlukla tercih edilmektedir. Alfabeleri benzer olsa dahi her ülkenin farklı resmi dili olduğu gibi ülkelere özel olarak farklı işaret dilleri ve alfabeleri mevcuttur. Bu sebeple işitme engelliler genellikle ülkelerinin işaret dili alfabesini ve işaret dilini kullanarak iletişim sağlarlar. Türkiye'de Türkçe İşaret Dili (TİD) işitme engelliler tarafından yaygın olarak kullanılmaktadır.

İşaret dili yalnızca işitme engelli olan insanlar tarafından kullanılmamaktadır. İşittiği halde konuşamayan, konuşmak için çeşitli aletlere ihtiyacı olan, dil-dudak-damak yapısında bozukluk olan insanlarda işaret diline başvurmaktadır. Sadece işitme engelli insanlarla sınırlı kalmayan bu işaret dili probleminin çözümü olarak yabancı dillerde olduğu gibi tercüman kullanılması ile çözüm sağlanabilir. Ancak tercüman kullanımı hem pratik olmadığından hem de tercüman kişiye her an ulaşılamadığından bu problemin çözümü için yeterli olmadığı düşünülmektedir.Teknolojideki gelişmelerle birlikte İşaret dilini otomatik olarak tanıyabilecek ve iletişimi sağlayacak bir otonom sisteme ihtiyaç duyulmaktadır.

Birçok araştırmacı tanımlama ve sınıflandırma işlemlerini artık bilgisayarların yapılması gerektiğini düşünerek 1990 yılından itibaren birçok başarılı işaret dilinde bulunan sinyal ve verileri işlemişlerdir. İlk araştırmalar Charahpayan ve Marble tarafından Amerikan İşaret Dilinin oluşturulması amacıyla elin hızının kullanılması sistemine dayalı bir bilgisayar görmesi tabanlı sistem geliştirmişlerdir [1].

Takahashi ve Kishimo veri eldiveni vasıtasıyla alınan 46 örnek üzerinden Japon İşaret Dili alfabesini sınıflandırmaya ve tanımaya çalışmışlardır [2]. Yine 1990 ve 1996 yıllarında Kramerve Leifer veri eldiveni ile heceleme tabanlı olarak Amerikan İşaret Dili üzerinde çalışma yapmışlar ve bu çalışmadan doktora tezi çıkartmışlardır [3,4]. Murakami ve Taguchi 110 farklı Japon İşaret Dili işaretini tanımayı gerçekleştirmişlerdir [5].

1995'de Waldron ve Kim yapay sinirağları metodunu kullanarak, el şekli, el konumu ve şekli kullanarak 14 Amerikan İşaret dili kelimesi tanıma gerçekleştirmişlerdir [6] Wysoski, ve arkadaşları 2002'de görmeye dayalı bir Amerikan İşaret Dili tanıma sistemi gerçekleştirmişlerdir [7]. Yapay sinir ağı kullanarak 26 sabit el şekli tanıması gerçekleştirmişlerdir. Allen ve arkadaşları Amerikan işaret dili için heceleme sistemi gerçekleştirmişlerdir. Bu sistem 26 el şeklinden 24'ünü tanıyabilmekteydi [8]. 2004'de Wang, ÖZ ve arkadaşları, Amerikan İşaret Dili el şekilleri tanıma sistemi gerçekleştirdiler. Sistem veri eldivenleri ve motion traker kullanmakta, sınıflandırmadaise yapay sinir ağları ve Markov modeli kullanılmıştır [9] 2011 yılında sensör eldivenle toplanan Amerikan İşaret Dili verileri çok katmanlı bir yapay sinir ağı kullanılarak sınıflandırılması yapılmıştır [10].

Genel olarak yapılan çalışmalarda hem işaret dili ile ilgili veri seti hazırlama hem de bilgisayar destekli bir tanıma işlemi için kullanılmıştır. Çalışmaların büyük bir kısmını Amerikan İşaret Dili çevrimi ile ilgili yapılan çalışmalardan oluştuğu görülmektedir.

2002 yılında parmak hareketlerini takip ederek bir non-linear model oluşturulması amacıyla Bowden ve arkadaşları tarafından da bir çalışma yapılmıştır [11]. Hint İşaret Dili üzerinde evrişimsel sinir ağı kullanılarak ve eklem açısal yer değiştirme haritaları üzerinde bir tanıma işlemi gerçekleştirilmiştir [12,13]. Yine Yunan İşaret Dili üzerinde yapılan çalışmalara bakıldığında sensör eldiveni ile alınan verilerin anlamlandırılması ve derin öğrenme algoritmaları ile ilgili işaret dilinde çalışlar yapılmıştır [14,15]. Pakistan İşaret Dilinde yapılan çalışmalara bakıldığında klasik makine öğrenmesi algoritmaları ile çalışmalar yapıldığı görülmektedir [16]. 2017 yılında Vietnam işaret dili üzerinde yapılan yapay sinir ağı çalışmaları yapılmıştır [17].

Literatürde birçok yabancı işaret dili ile ilgili yapılan çalışmaların olduğu görülmüştür. TürkİşaretDili ile ilgili olarak yapılan çalışmalardan 2005 yılında Haberdar ve Albayrak tarafında yapılan çalışmaya bakıldığında HMM (Hiden Markov Model) bir sınıflandırma çalışması yapıldığı ve %95,7 gibi bir başarı oranı yakalandığı görülmüştür [18]. Yine 2006 yılında aynı kişiler tarafından yapılan çalışma ile 2 basamaklı görsel tanımlamaya dayalı global ve lokal özelliklerin tanımlanması ile ilgili bir çalışma yapılmıştır [19]. 2013 yılında Albayrak ve Memiş tarafından RGB video verileri ve Kinect tarafında alınan veriler ayrık kosinüs dönüşümü vasıtasıyla kümülatif hareket resimlenin tanınması işlemi gerçekleştirilmiş ve %90 başarı elde edilmiştir [20]. 2017 yılında TİD üzerinde veri eldiveni kullanılarak elde edilen dataların sınıflandırma işlemi gerçekleştirilmiştir [21]. 2020 Ağustos ayında Çelik ve Odabaş tarafından yapılan çalışmada kamera karşısında yapılan 10 rakam ve 29 harfin işaret dili hareketleri ile eğitilen CNN + LSTM modelleri kullanılarak tahmin edilmesinde %97 başarı oranı elde edildiği görülmüştür. [22].

TİD tanınması amacı ile bazı modellerde veri eldiveni kullanılarak modellemelerin yapıldığı görülmüş, günlük hayatta veri eldivenin temininin kolay olmayacağı aşikâr olduğundan veri eldiveni ile yapılan modellemelerin yeterli katkıyı sağlayamayacağı düşünülmektedir. Kinect sensörü kullanılarak yapılan modellemelerin ise genellikle bu sensöre sahip okul, hastane, banka vb gibi yerlerde faydalı olacağı düşünülmektedir. 2020 Ağustos ayında Çelik ve Odabaş tarafından yapılan çalışmada ise normal kamera karşısında elde edilen videoların ön işleme adımında işaret dilini oluşturan görüntülerdeki kafa bölümünün karartıldığı sonraki adımda ise ellerin tespiti amacı ile ten rengi ayrımından faydalanıldığı görülmüştür. Ancak İşaret dilini yapan şahısların her zaman uzun kollu giysiler giymeyeceği düşünüldüğünde görüntülerdeki ten rengi algılama ile ellerin tespit edilmesinin zor olacağı öngörülmektedir.

Literatür taraması sonucunda yapılan çalışmalarda kullanılan veri eldiveni ve Kinect sensörü gibi extra donanımlara sahip sistemlerin başarım oranlarının yüksek olduğu görülmekle birlikte, oluşturulacak sistemlerin maliyeti nedeniyle günlük yaşama sağlayacağı katkılar açısından yeterli olamayacağı değerlendirilmektedir. Normal kamera karşısında yapılan çalışmanın %97 oranındaki başarısının günlük yaşama yapacağı katkının daha fazla olacağı, daha kullanılabilir ve gerçekleştirilmesi daha kolay sistemler olduğu görülmekle birlikte, yapılan çalışmadaki ten rengi ile ellerin tespit edilmesinin insanların giysilerinde oluşacak değişim sonucunda istenilen başarıyı sağlayamayacağı düşünülmektedir. Yapılan bu çalışma ile kamera karşısında yapılan hareketlerdeki işaret dili harflerinin tespit edilmesi sağlanmıştır. Bunun için öncelikli olarak Türk İşaret Dili Alfabesi ile ilgili, herkesle açık şekilde internet ortamında paylaşılan video içeriklerinden ve çalışmada oluşturulan Türk İşaret Dili Alfabesindeki harfleri içeren videolardan elde edilen resimlerle bir veri seti oluşturulmuştur. Oluşturulan veri seti kullanılarak, gerçek zamanlı nesne tespiti konusunda başaralı sonuçlar elde edilmiş olan YOLO modeli kullanılarak video içeriklerinden Türk İşaret Dili alfabesindeki harflerin tanınması gerçekleştirilmiştir.

Bu tez çalışması 6 bölümden oluşmaktadır. Giriş bölümünde, tez çalışması ile alakalı genel bilgiler ve literatür taramasına yer verilmiştir. Bölüm 2'de Türk İşaret Dili ile ilgili bilgilere yer verilmiştir. Bölüm 3'de Yapay Zekâ, Makine Öğrenmesi, Yapay Sinir Ağları ve Derin öğrenme ile ilgili olarak genel bilgilere yer verilmiştir. Bölüm 4'de Nesne tanıma ile ilgili olarak kullanılan derin öğrenme modellerinin yapısı anlatılmıştır. Bölüm 5'de oluşturulan veri seti ile ilgili bilgilere yer verilmiş, nesne tanıma modellerinin karşılaştırılması yapılmış ve YOLOv5s modelinin kullanıldığı Türk İşaret Dili Alfabesindeki harflerin tanınmasına yönelik olarak hazırlanan uygulamanın gerçekleştirilme adımları anlatılmıştır. Bölüm 6'da geliştirilen uygulamanın performans değerlendirmesi yapılmış ve sonuçlara yer verilmiştir.

2. TÜRK İŞARET DİLİ

Türkiye'de işitme engelliler tarafından kullanılan dil Türk İşaret Dilidir (TİD). İşaret dilleri bulundukları ülkelerin dilbilimsel yapısından farklı bir özelliğe sahiptir. Bu kapsamda Türk İşaret Dilinin de kendine özgü dilbilimsel bir yapısı yani kendine özgü işaretleri, alfabesi, rakamları ve dilbilgisi vardır. İşaret dilleriyle konuşma dilleri arasında çok kuvvetli bir bağlantı yoktur, her iki dil ailesinin de kendine özgü bir içyapısı vardır. TİD hakkında TDK da Milli Eğitim Bakanlığı tarafından 2015 tarihinde yayımlanan "Türk İşaret Dili Sözlüğü" görsel bir sözlük bulunmaktadır [23]. 2607 kelimeden oluşan bu sözlük işitme engellilerin sıklıkla kullandığı kelimeleri içermektedir.

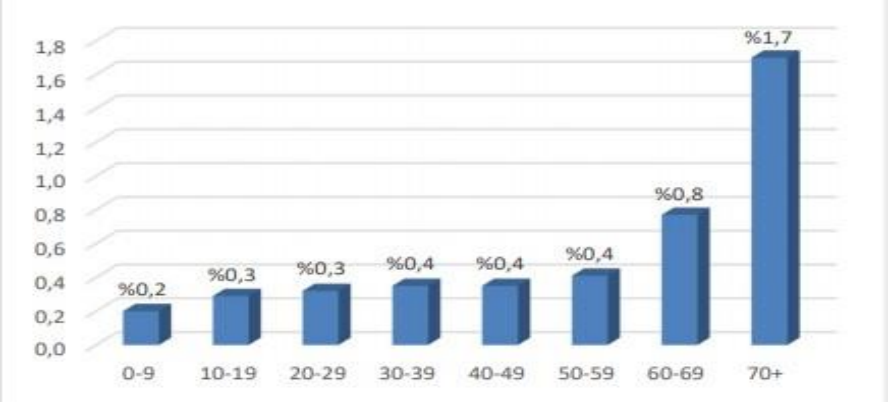
2.1. Türk İşaret Dilinin Tarihi Gelişimi

Türk İşaret Dili II. Abdülhamit zamanında İstanbul'da açılan sağırlar okulu ile birlikte yaklaşık olarak 120 yıllık bir geçmişe sahiptir. [24] Birçok işaret dilinden eski bir tarihe sahiptir. Ancak kullanılmakta olan Türk İşaret Dilinin Osmanlı zamanındaki işaret diline dayandığına dair kesin bilgiler bulunmamaktadır. [25]

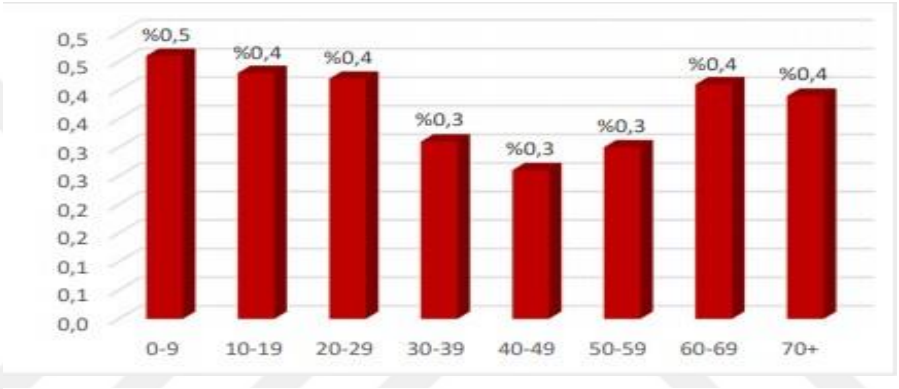
İşaret dillerinin tarihine bakıldığında konuşma dillerine göre daha yeni ve genç diller olduğu görülmektedir. İşaret dili ile ilgili olarak yapılan ilk çalışmaların 1970 yıllarda yapılması bu dillerin ne kadar yeni ve olduğunu göstermekle birlikte insanların bu dilleri binlerce yıldır kullandığı da anlaşılmaktadır. Yine de bu dillerin konuşma dili kadar eski bir tarihe sahip olduğu düşünülmektedir. Engelli bireyler günümüzde oluşturulan ulusların belirli bir standarda ulaşmış işaret dili yapısına göre olmasa da kendi aralarında bir iletişim aracı olarak bir dil kullandıklarının bilinmesi gerekmektedir.

2.2. Türkiye'de Türk İşaret Dili Kullanıcıları

Türkiye İstatistik Kurumu'nun verilerine göre Türkiye'de 89,043 kişi (53,543'i erkek 35,500'i kadın) işitme engelli ve 55,480 kişi de (34,672 erkek, 20,808 kadın) konuşma engellidir [26]. Şekil 2.1 ve Şekil 2.2 de TÜİK, Türkiye Engelliler Araştırması, 2002'ye göre işitme ve konuşma engelli nüfus yüzdesinin yaşa göre dağılımını gösteren şekillerden görüleceği gibi işitme engelli kişilerin yüzdesinin yaş ile beraber arttığını görebiliriz.



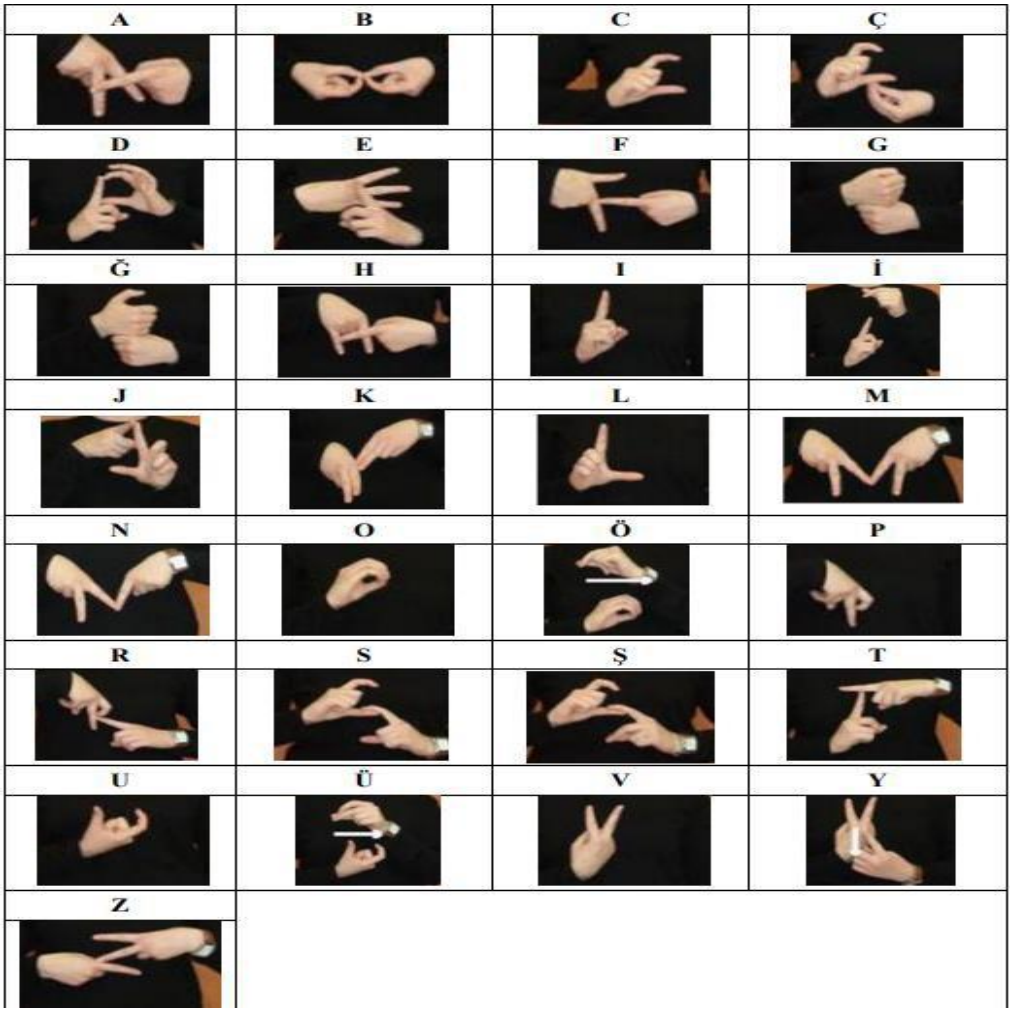
Şekil 2.1 Ülkemizdeki işitme engellilerin yaşa göre dağılım grafiği [26]



Şekil 2.2 Ülkemizdeki dil ve konuşma engellilerinin yaşa göre dağılım grafiği [26]

2.3. Türk İşaret Dili Alfabesi ve Rakamları

Türkçeden türetilen Türk İşaret Dili yüzde yüz oranı ile Türkçeye benzerlik göstermemektedir. Türk İşaret Dili dilbilimsel olarak Türkçeden farklı özelliklere sahiptir. Genel olarak kelimelerin bir araya gelmesi ile oluşan cümleler vasıtasıyla iletişimin sağlandığı Türk İşaret Dili kelimelerin yetersiz kaldığı durumlarda ya da özel isimler için harf ve rakam işaretlerini içeren bir yapıya sahiptir. Bu sebeple Türkçe de yer alan bütün harflerin ve rakamlarınTürk İşaret dilinde bir karşılığı mevcuttur. Türk İşaret dilindeki harfler Şekil 2.3'de görselleştirilmiştir.



Şekil 2.3 Türkçe işaret dili alfabesi [27]

3. DERİN ÖĞRENMENİN GELİŞİMİ

3.1 Yapay Zekâ

Yapay zekâ, genel anlamda belirli bir görevi yerine getirmek için insan zekâsını taklit eden sistemler veya makineler anlamına gelmektedir. Yapay zekâ sistemleri var olan durumu gözlemleyerek bu gözlemleri daha önceden belirlenen parametreler doğrultusunda işleyen ve problemi çözmek adına istenilen durumlara karşı tepki veren sistemlerdir.

Yapay zekânın doğuşu 1956 yılında Birleşik Devletler Dartmouth 'da dönemin yüksek seviyeli bilim adamlarını bir araya getiren bir konferans olduğu söylenebilir. Bu konferansta J. McCarthy, M. Minsky, C.Shannon, A.Newell ve H. Simon zekâ ile donatılmış bilgisayar programlarını yapay zekâ olarak nitelendirmişlerdir.

Yapay zekâ alanında ilk gelişmelerin meydana geldiği bu dönemde zekâ testlerindeki benzer geometrik şekillerin ayırt edilmesinde kullanılan bir programın geliştirildiği bilinmektedir. Yapay zekâ kavramı ile birlikte 1950'li yıllardan sonra makinelerin basit işlemleri yapabileceği ve bazı görevleri yerine getirebildiği görüldüğünden, makinelerin öğrenme yetisine sahip olabileceği düşüncesi ortaya çıkmıştır. Böylelikle derin öğrenme süreci başlamış olmaktadır. Şekil 3.1'de yapay zekâ, makine öğrenmesi, yapay sinir ağları ve derin öğrenme kapsamları görselleştirilmiştir.

Şekil 3.1 Yapay zekâ kapsamı

3.2 Makine Öğrenmesi

Yapay zekânın belirli bir sistemdeki verilerden öğrenmesine imkân tanımasıyla meydana gelen yapısı makine öğrenmesi olarak adlandırılabilir. Daha geniş anlamda makine öğrenmesi, bir sistemin insanlar gibi daha önce görmüş olduğu örnekleri kullanarak yeni karşılaşılan durumlarda yeni sonuçlar üretebilmesini sağlaması ile gerçekleşir. Makinelerin öğrenmesi gerçekleştirilirken genelde basit matematiksel modeller kullanılır fakat uzun işlem gücü gerektiren ve matematiksel modeli basit bir şekilde ifade edilmeyen durumlarda makine öğrenmesi algoritmalarından yararlanılır. Bu algoritmalar birçok veri kaynağından veri toplamaya, bu veri setlerini dönüştürmeye ve sonuçlara göre işlem yapmaya imkân sağlamaktadır. Makine öğrenmesi algoritmaları girdi olarak verilen veri setine ve bu veri setinin sonucunda oluşacak çıktıya göre denetimli öğrenme ve denetimsiz öğrenme olarak iki ana başlık altında toplanmıştır.

3.2.1 Denetimli Öğrenme

Makine öğrenmesi başlıklarından denetimli öğrenme basit olarak belirlenmiş bir veri kümesi ve bu verilerin nasıl sınıflandırılması gerektiğini içeren sistemdir. Denetimli öğrenmede veri seti üzerindeki özniteliklerin çıkarımı sağlanıp çıktı olarak belirli sınıflar oluşturması hedeflenir. Öğrenme işleminde veriler ve o verilerden çıkan sonuçlardan bir fonksiyon elde edilerek giriş verileri ile çıkan sonuçlar arasındaki ilişki öğretilmiş olur. Yeni girilecek verilerin doğru şekilde tahmin edilmesindeki hata oranı azalana kadar bu eğitim işlemine devam edilir.

Denetimli öğrenme algoritmalarının bazıları şunlardır:

1. Regresyon
2. Destek Vektör Makineleri
3. Karar Ağaçları
4. Naive Bayes Sınıflandırıcıları
5. Yapay Sinir Ağları
6. Sınıflandırma ve Regresyon Ağaçları

3.2.2 Denetimsiz Öğrenme

Denetimsiz öğrenme çıktıları veya sınıfları olmayan verileri kümeleme işlemi olarak tanımlanabilir. Denetimsiz öğrenme, denetimli öğrenmeye kıyasla daha karmaşık algoritmalar üzerinde çalışır, çünkü veriler hakkında nadiren veya hiç bilgi yoktur. Bizim için sonuç üretmeyi amaçlayan makine veya sistem olarak daha az yönetilebilir bir ortam yaratır. Denetimsiz öğrenmenin temel amacı, gruplar, kümeler, boyutluluk azaltma gibi varlıkları aramak ve yoğunluk tahminini gerçekleştirmektir.

Denetimsiz öğrenme algoritmalarının bazıları şunlardır:

1. Kümeleme
   - 1.1 Hiyerarşik
   - 1.2 K-Means
   - 1.3 KNN
   - 1.4 Bulanık K-Means
   - 1.5 Gürültü Uygulamalarının Yoğunluk Temelli Kümelenmesi(DBSCAN)
   - 1.6 Karışım Modelleri
2. Anomali Tespiti
3. Sinir Ağları
   - 3.1 Oto-Kodlayıcırlar
   - 3.2 Derin İnanç Ağları
   - 3.3 Hebbian Öğrenimi
   - 3.4.Çekişmeli Ağlar
   - 3.5 Self – Organizing Map
4. Temel Bileşenler Analizi

3.2.3. Denetimli ve Denetimsiz Öğrenme Karşılaştırılması

Denetimli ve denetimsiz öğrenme algoritmalarının genel yapısına bakıldığında her ikisinde belli problemlere odaklandıkları görülmektedir. Denetimli ya da denetimsiz bir makine öğrenmesi algoritması kullanmayı seçmek, tipik olarak verilerinizin yapısı ve hacmi ile ilgili sorunların kullanım durumuna ve eldeki sorunun çözüm durumuna bağlıdır.

Denetimli ve denetimsiz öğrenme veri seti, hesaplama karmaşıklığı, çalışma zamanı, performans ve kullanım alanlarına göre Tablo 3.1'de karşılaştırılmıştır

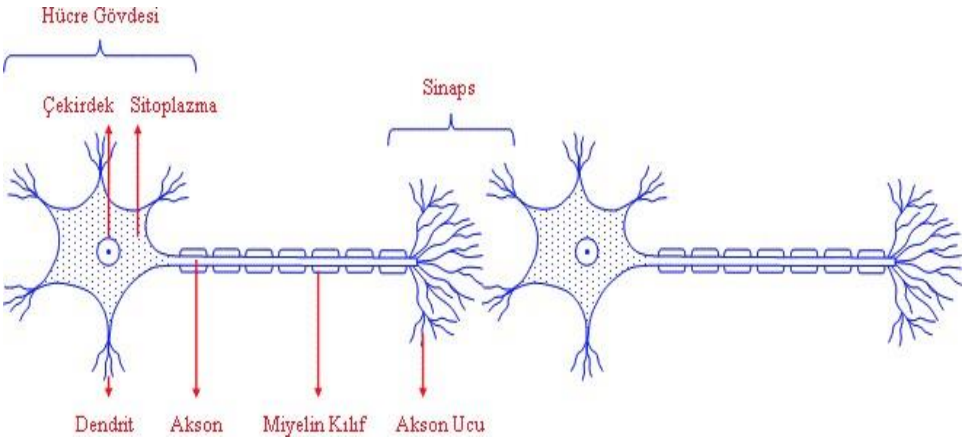
Tablo 3.1 Denetimli-denetimsiz öğrenme karşılaştırılması

Parametre	Denetimli Öğrenme	Denetimsiz Öğrenme
Veri Seti	Etiketli veriler ile ilgilenir	Etiketlenmemiş verilerle ilgilenir
Hesaplama Karmaşıklığı	Daha az hesap karmaşıklığı	Daha fazla hesap karmaşıklığı
Çalışma Zamanı	Çevrimdışı analiz de yapabilirler	Gerçek zamanlı analiz yaparlar
Performans	Doğru ve güvenilir sonuçlar Üretir	Daha az doğru ancak güvenilecek Sonuçlar
Kullanım Alanları	Sınıflandırma ve Regresyon	Kümeleme ve Birliktelik Analizi

3.3. Yapay Sinir Ağları

Yapay sinir ağları, insan beyninin çalışma mekanizmasını taklit ederek beynin öğrenme, hatırlama genelleme yapma yolu ile yeni bilgiler türetebilme gibi temel işlevlerini gerçekleştirmek üzere geliştirilen mantıksal yazılımlardır.

İnsan beyninin çalışma mekanizması içerisinde yer alan biyolojik nöronlar birbirleriyle iletişim kurmak için sinapsleri kullanırlar. İşlenen bilgiler axon adı verilen yapı aracılığıyla diğer hücrelere gönderilir. Sinir ağları bağlantılı işlem elemanlarından meydana gelir ve kurulmuş her bağlantı ağırlık değerine sahiptir. Sinir ağının öğrenmiş olduğu bilgi ağırlık değerinde saklıdır. Biyolojik nöronlara benzer şekilde YSA hücreleri bilgi toplama fonksiyonu vasıtasıyla bilgileri toplar, bu bilgileri bir aktivasyon fonksiyonundan geçirerek anlamlı çıktı üretir. Üretilen çıktı ağın diğer elemanları ile paylaşılır. Bilgi toplama aşamasında ihtiyaca göre değişik toplama ve aktivasyon fonksiyonları kullanılabilir. Biyolojik Sinir Hücresi Şekil 3.2'de gösterilmiştir.



Şekil 3.2 Biyolojik sinir hücresi [28]

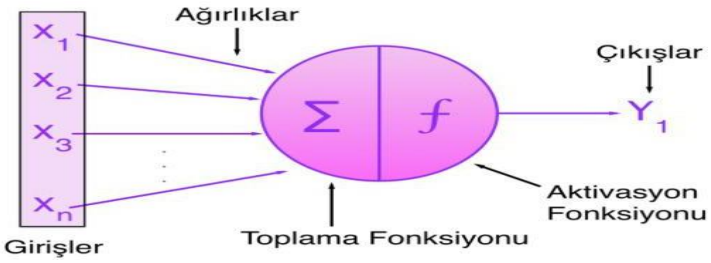
3.3.1. Tek Katmanlı Algılayıcılar (Perseptron)

Perseptron tek katmandan oluşan sinir ağ yapısının en basit şeklidir. Bu ağ yapısı yalnızcagiriş ve çıkış katmanlarından oluşur. Eğitilebilecek tek bir yapay sinir hücresine sahiptir.

Perseptron yapısı hata tabanlı öğrenme yapar. Giriş değerleri başlangıçta rastgele verilen ağırlık değerleriyle bir çıkış üretir. Üretilen bu çıktı değeriyle, beklenen değer arasındaki farkyani hata değerine göre ağırlıklar güncellenir ve kabul edilebilir hata değerine ulaşılana kadar bu adımlar güncellenmiş ağırlık değerleriyle tekrarlanır. Bu yapının basit sınıflandırma ve kümeleme problemlerini çözerken, problemler zorlaştıkça başarısız olduğu görülmüştür. Perseptron modeliyle yalnızca lineer olarak ayrıştırılabilen sınıflar ayrıştırılabilirken, XOR problem gibi lineer olarak ayrıştırılamayan problemlerin çözülememesi yapay sinir ağları alanında uzun bir dönem çalışmaların durmasına sebep olmuştur [29].

Bir yapay sinir ağının, girdilerden doğru çıktıları üretebilmesi için ağırlık değerlerinin doğru olarak belirlenmesi gerekir. Sistemin doğru çıktıları üretmesi için doğru ağırlıkların belirlenmesi işlemi, sistemin eğitilmesi ile yapılır. Bu ağırlık değerleri başlangıçta rastgele değerler olarak atanır ve ağın öğrenme kuralına göre güncellenirken doğru ağırlık değerlerisaptanır [30].

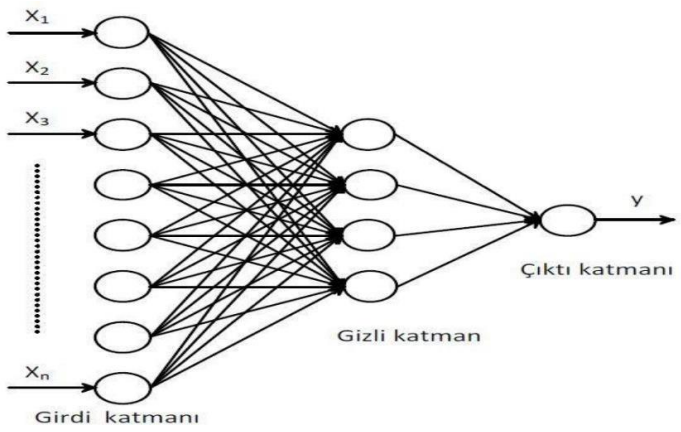
Yapay sinir ağının öğrenmesi için problem uygun ağ modelinin seçilmesi önemli rol oynar. Ağ modeli oluşturulurken; ağın topolojisi, kullanılan toplama ve aktivasyon fonksiyonunun özelliği, öğrenme stratejisi ve kuralları göz önünde bulundurulmalıdır [31].Şekil 3.3 tek katmalı algılayıcıları görselleştirmektedir.



Şekil 3.3 Tek katmanlı algılayıcı [28]

3.3.2. Çok Katmanlı Algılayıcılar

Çok katmanlı algılayıcıları birden çok nöronun birbirlerine bağlanması ile elde edilmiş bir yapıya benzetmek mümkündür. Her nöron çıktısından elde edilen değer diğer nörona girdi olarak iletilmekte ve bu işlem tüm ağ boyunca devam etmektedir. Çok katmanlı algılayıcılar nöronlara giriş yapacak olan değerler ve bias değerinden oluşan giriş katmanı, birden çok nöron hücresinden oluşan ara katmanlar (Gizli Katmanlar) ve son olarak toplam fonksiyonu ileoluşturulan sonucunun aktivasyon fonksiyonu ile sınıflandırma işleminin yapıldığı çıktı katmanından meydana gelir. Çok katmanlı algılayıcıların modellemesi Şekil 3.4 'da gösterilmiştir.



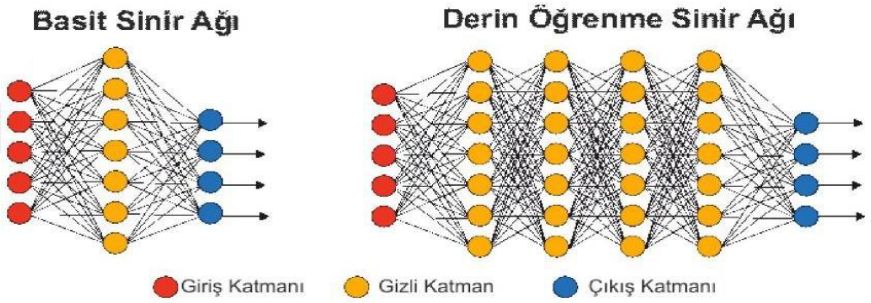
Şekil 3.4 Çok katmanlı algılayıcılar [28]

3.4 Derin Öğrenme

Derin öğrenme, büyük miktarda denetimsiz veri kullanarak sınıflandırma problemlerinde özellik çıkarımı maliyetini ortadan kaldırmış, yapay sinir ağlarının özelleştirilmiş pek çok gizli katmandan ve işlem elemanından oluşan bir çeşididir.

Derin öğrenme algoritmaları veri seti üzerinde gizli kalmış özniteliklerin çıkarılması işlemini otomatik olarak gerçekleştirerek makine öğrenmesi algoritmalarına göre birçok problemikolaylıkla çözebilmektedir. Bu sebepledir ki derin öğrenme algoritmaları öznitelik çıkarım işlemleri için büyük veri setine ihtiyaç duyar.

Derin öğrenme ağ yapısını basit bir yapay sinir ağından ayıran en temel özellik Şekil 3.5'den görülebileceği gibi birden fazla gizli katmana sahip olması ve daha karmaşık bir ağ yapısına sahip olmasıdır



Şekil 3.5 Basit sinir ağıyla derin öğrenme sinir ağı karşılaştırması [29]

Derin öğrenme veya Derin Sinir Ağları 4 farklı temel kategoride sınıflandırılabilir;

- Evrişimsel Sinir Ağları
- Tekrarlı Sinir Ağları
- Kısıtlanmış Boltzmann Makineleri
- Otomatik Kodlayıcırlar

3.5. Evrişimsel Sinir Ağları (CNN)

Convolutional Neural Network olarak adlandırılan CNN 2010 yılı itibari ile kullanılmaya başlanmıştır. Türkçeye Evrişimsel Sinir Ağları olarak geçen bu sinir ağı yapısı ImageNet veri seti üzerinde birçok farklı görevi başarılı bir şekilde yerine getirmiştir. Bilgisayarlı görü alanının yanında Doğal Dil İşlemeye dayalı sınıflandırma, Otomatik Tanımlayıcılar, Nesne Tanıma, Metinve Video işleme gibi alanlarda başarılı bir sinir ağı yapısıdır. Evrişimsel sinir ağları, girdi katmanı, özellik çıkarmaya yarayan konvolüsyon/evrişim, parametre azaltan pooling (havuzlama), nöronların aşırı öğrenmesini engellemeye yarayan dropout (seyreltme), ve tamamen bağlantılı bir derin sinir ağından oluşmaktadır.

3.5.1. Girdi Katmanı

Evrişimsel sinir ağlarının ilk katmanı olan girdi katmanı genel olarak ham veri setinin veya boyutla ilgili düzenlemelerin yapıldığı veri setlerinin ağa verildiği, giriş verisine göre ağın derinliğinin belirlendiği ve genel olarak ağın başarımını etkileyen katmandır.

3.5.2. Evrişim Katmanı

Girdi verileri içeriğinden özelliklerin çıkarımının yapıldığı katmandır. Görüntü işlemede pikseller arasındaki uzamsal ilişkiyi koruyarak giriş verisi üzerinden görüntü özelliklerini öğrenir. Filtreler içinde sayılar bulunduran matrislerdir ve girdiden özellik çıkarmak için kullanılırlar. Özellikleri çıkarımında konvolüsyon filtresi olarak adlandırılan filtreler kullanır. Yani evrişim işlemi resim üzerinde kaydırılan bir filtre ile gerçekleşmektedir. Filtrenin elemanları her adımda üzerine gelen resim piksel değerleri ile çarpılmakta ve tüm filtreden gelen çarpım değerleri toplanarak aktivasyon matrisi veya özellik matrisi adı verilen matrislere kaydedilmektedir. Bir konvolüsyon katmanında birden fazla konvolüsyon filtresi bulunabilir. Bu durumda konvolüsyon filtreleri klasik sinir ağlarındaki nöron katsayılarına benzetilebilir.

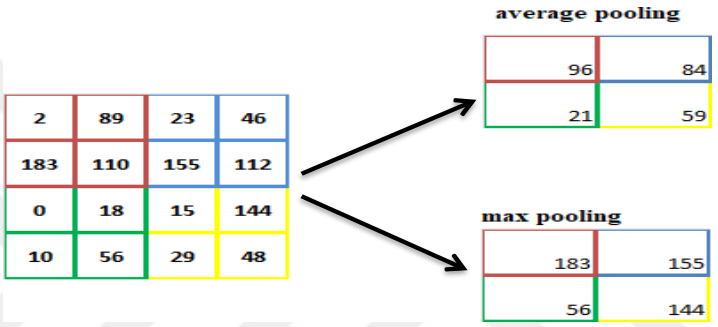
Her konvolüsyon işleminden sonra çıktı verisinin boyutunun, girdi verisinden daha küçük olacaktır. Bu nedenle konvolüsyon işleminden önce, girdi ile çıktı matrisinin boyutlarının istenilen değerde olması için bir doldurma (padding) işlemi gerçekleştirilmelidir. Padding bir resmin kenarlarının çerçeve şeklinde yeni bitler ile doldurulması işlemidir. Padding işlemi yapılırken genellikle sıfır değeri kullanılmaktadır. Evrişim katmanından sonra değerleri belli aralıkta almak adına aktivasyon fonksiyonları kullanılır. Problemin farklılığına göre farklı aktivasyon fonksiyonları kullanılmaktadır.

3.5.3. Havuzlama (Pooling) Katmanı

Evrişimsel sinir ağlarında belirli yöntemler kullanarak büyük veri miktarlarında azaltma işlemlerinin yapıldığı katmandır. Bu katmanda boyut azaltma işlemi gerçekleşir ve herhangi bir öğrenme işlemi gerçekleştirilmez. En yaygın olarak kullanılan pooling katmanları Maksimum Havuzlama (Max-pooling) ve Ortalama Havuzlama (Average-pooling) yöntemleridir.

Maksimum havuzlama yönteminde her defasında yöntemi oluşturan filtrenin kapsadığı alandaki en büyük değer çıktı olarak verilmektedir.

Ortalama havuzlama yönteminde filtrenin kapsadığı alandaki değerlerin ortalaması alınarak çıktı olarak verilmektedir. Şekil 3.6 Havuzlama işlemine ait görseli içermektedir.



Şekil 3.6 Havuzlama işlemi

3.5.4 Tamamen Bağlı Katman

Tamamen bağlı katman, evrişim ve havuzlama katmanlarıyla ile özellikleri çıkarılan verilerin klasik çok katmanlı yapay sinir ağına özelliklerin girdi olarak verilmesinden oluşmaktadır. Tamamen bağlı bir katman eklemenin amacı, bu özelliklerin doğrusal olmayan kombinasyonlarını öğrenilebilmesidir. Evrişim ve havuzlama katmanlarından öğrenilen özelliklerin çoğu iyi olabilir ancak bu özelliklerin kombinasyonları daha da iyi olabilmektedir. Yapay sinirağı klasik olarak çok katmanlı ağların yapısıdır ve ağın sonunda bir sınıflandırıcı aktivasyon fonksiyonu kullanılmaktadır. Uygulamalara bakıldığında binary sınıflandırma için softmax, çoklu sınıflandırma için ise en efektif fonksiyonun sigmoid olduğu gözükmektedir. Sigmoid fonksiyonunun çoklu sınıflandırmadaki başarısı sınıf sayısına göre 0-1 değerleri arasında çıktı elde etmesi olarak gösterilebilir.

4. NESNE TESPİTİ VE KULLANILAN MODELLER

4.1. Nesne Tespiti

Nesne tanıma veya nesne tespiti girdi olarak verilen görüntü içeriğinden istenilen nesnenin algılanması ve ayrımının yapılması olarak tanımlanabilir. Nesne tespiti için veri girişi ve ön işleme, girdi olarak verilen resmin özniteliklerinin çıkarılması işlemleri ve çıkarımı yapılan bu özniteliklerin seçimi için geliştirilen farklı yöntemler kullanılmaktadır. İlk aşamada tespit edilmesi arzulanan nesnenin de bulunabileceği giriş verisinin formatının ayarlanması ve gürültülerden arındırılması gibi giriş verilerinin amaca uygun bir şekle getirilmesi işlemleri ön işleme adımları olarak adlandırılabilir. Tespiti yapılması aşamasında planlanan bir nesnenin önceden tanımlanmış özellikleri ve ölçütleri kullanılarak girdi verisi üzerinden tespit işleminin gerçekleşmesi şeklinde tanımlanabilir. Tespiti yapılacak nesnenin ayrıntılarını içeren bilgilerin elde edilmesi ise özniteliklerin çıkarılması adımını oluşturmaktadır.

Nesne tanıma işleminin anlaşılabilmesi adına öncelikle sınıflandırma ve kümeleme kavramlarının anlaşılması gerekmektedir. Sınıflandırma olarak tanımlanan kavram girdi olarak verilen datanın daha önceden tanımlanmış olan sınıflardan sistem çıkışına göre uygun olan sınıfa dâhil edilmesi olarak tanımlanmaktadır. Kümeleme işlemi gerçekleştirilirken ise birbirine benzer giriş verilerinden sistem çıkışına göre benzer özellikler barındıran nesnelerin aynı grup içerisinde toplanması olarak tanımlanmaktadır. Bu tanımlamalardan sonra nesne tespiti işlemi gerçekleştirilirken giriş veirisi içerisinde aynı nesneden birden fazla bulunması veya aynı nesneden birkaç tanesinin yanyana bulunabilme ihtimali mevcuttur. Bu ihtimallerde dahi her bir nesnenin tespitinin yapılabilmesi nesne tespitini sınıflandırma işleminden ayıran en önemli farktır.

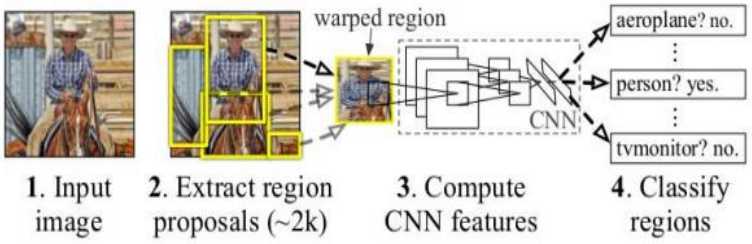
Literatüre bakıldığında nesne tanıma probleminin çözümü için arka plan çıkarımı, görüntüler arasındaki farkın alınması ve optik akış gibi farklı yöntemler mevcuttur. Nesne tanıma problemlerinde sıkça kullanılan evrişimsel sinir ağları genel olarak iki temel üzerinde işlemlerini gerçekleştirir. Bu temeller iki aşamalı ve tek aşamalı nesne tespiti şeklinde isimlendirilir.

4.2. İki Aşamalı Nesne Tespit Modelleri

Nesne tespiti amacıyla geliştirilen ilk modeller iki aşamalı modellerden oluşmuştur. İki aşamalı nesne tespiti yönteminde öncelikli olarak resim üzerinde nesnenin bulunabileceği muhtemel olan bölge önerilir ve daha sonrasında önerilen bu bölge içeriğinde nesne tespiti yapılır. İki aşamalı nesne tespiti modelleri R-CNN, Fast-RCNN ve Faster-RCNN modelleridir.

4.2.1. Regions with CNN fatures (R-CNN)

R. Girshick ve arkadaşları 2013 yılında R-CNN modelini önermişlerdir. SIFT ve HOG gibi algoritmaların kullanıldığı nesne tespiti modellerine nazaran % 30 civarında bir daha başarlı sonuçların elde edilebildiğini öne sürmüşlerdir. R-CNN daha önceki modellerden farklı olarak özellik çıkarım işlemleri için CNN yapılarından faydalanmış ayrıca nesne tespiti için bölge önerilerinin yapıldığı algoritmalardan da faydalanmıştır. R-CNN adımlarından birincisi verilen giriş görüntüsü içeriğinde ilgi alanlarının (Regions of Interest) oluşturulması işlemidir. Her hangi bir kategoriye ait olmadan elde edilen bu ilgi alanları tespiti yapılacak olan nesnenin bulunma ihtimalinin yüksek olduğu sınırlayıcı kutular olarak tanımlanabilir. Bu sınırlayıcı kutuların tespiti amacıyla Selective Search algoritmasından faydalanılmış olup Selective Search algoritması dışındaki algoritmalarınında sınırlayıcı kutuların tespitinde kullanılabilir olduğu modeli ortaya atanlar tarafından dile getirilmiştir. Belirlenen bu ilgi alanları kırpılarak AlexNet CNN ağ yapısına giriş verisi olarak verilerek ikinci aşama işlemi gerçekleştirilmektedir. 227 X 227 boyutundaki renkli resimlerin özellik çıkarım işlemi beş konvolüsyon katmanı ve iki tam bağlantı katman kullanılarak tamamlanmış olur. İlk iki aşamada kullanılan yöntemlerle giriş görüntüsünden elde edilen özellikler sayesinde nesnenin giriş görüntüsünde bulunup bulunmadığını eğer bulunuyorsa hangi sınıfa ait olduğunu destek vektör makinesi belirler. Aşağıdaki Şekil 4.1'de R-CNN modelinin genel yapısı gösterilmektedir.

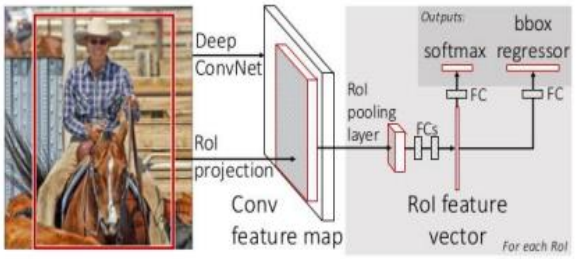


Şekil 4.1 R-CNN yapısı[32]

CNN yapısına uygun olarak önerilen ilk nesne tespiti modei R-CNN modelidir. Nesne tespiti alanında ilk defa kullanıldığından bazı eksiklikleri modeli oluşturan ekip tarafından daha sonrasında tespit edilerek R-CNN modelinin geliştirilmesi sağlanmıştır. Bu eksikliklerin başında model eğitiminin maliyetli olması gelmekle birlikte ikinci eksiklik model eğitimin ayrı ayrı aşamalardan meydana gelmesidir. Bu iki eksikliğin temel nedeni ilgili her bölge teklifi için algoritmaların ve Destek Vektör Makinelerinin kullanılarak ayrı ayrı kayıt işlemi gerçekleştirmesidir. Üçüncü eksiklik ise modelin nesne tespiti amacıyla kullanılabilmesi için yaklaşık olarak bir dakika gibi uzun sayılabilecek bir sürenin gerekmesidir. R-CNN modelini ortaya atan kişiler ilerleyen zamanlarda bu eksikliklerin giderilmesi amacıyla modeli geliştirmeye devam etmişlerdir.

4.2.2. Fast R-CNN

R-CNN modelindeki eksikliklerin giderilmesi amacıyla R-CNN modelini oluşturan R.Girshick ve arkadaşları tarafından 2015 yılında ortaya atılmmış bir modeldir. Bu model ile eksikliği giderilmeye çalışılan en büyük zaaf CNN katmanındaki her bölge için ayrı ayrı oluşturulan hesaplamalardır. Bu problemin çözümü tüm resim için bir defa hesaplama yapılması hedeflenerek aşılmaya çalışılmıştır. Giriş görüntüsü olarak Fast R-CNN modeline verilen görüntü maksimum havuzlama ve konvolüsyon katmanlarından geçirilir. Bölge teklifleri R-CNN modelinde olduğu gibi CNN yapısından ayrı bir şekilde oluşturulur. CNN ile elde edilen özellikler ve bölge teklifleri havuzlama katmanına aktarılır. Bölge önerileri n x m şeklindeki alanlara ayrılır ve bu alanlar maksimum havuzlama işlemine tabi tutulur. Bu sayede elde edilen bölge tekliflerindeki boyut farkları ne olursa olsun sonuç olarak n x m boyutunda özellik çıkarım işlemi gerçekleştirilecektir. Elde edilen bu özellikler tam bağlantı katmanı kullanılarak sınıflandırılacak ve istenilen nesnenin tespiti daha hızlı yapılacaktır. Her bölge teklifi için ayrı CNN hesaplamasının yapılmaması Fast R-CNN modelinin öne çıkan özelliğidir. Bu özellik sayesinde eğitim süresi 9 kat azalmış olmaktadır. Bu modelin başka bir farkı da tespiti yapılacak her nesne için ayrı bir DVM kullanmak yerine sınıf olasılıklarını hesaplayan Softmax katmanının kullanılmasıdır. Aşağıdaki Şekil 4.2'de Fast R-CNN modelinin yapısı görülmektedir.



Şekil 4.2 Fast R-CNN yapısı[33]

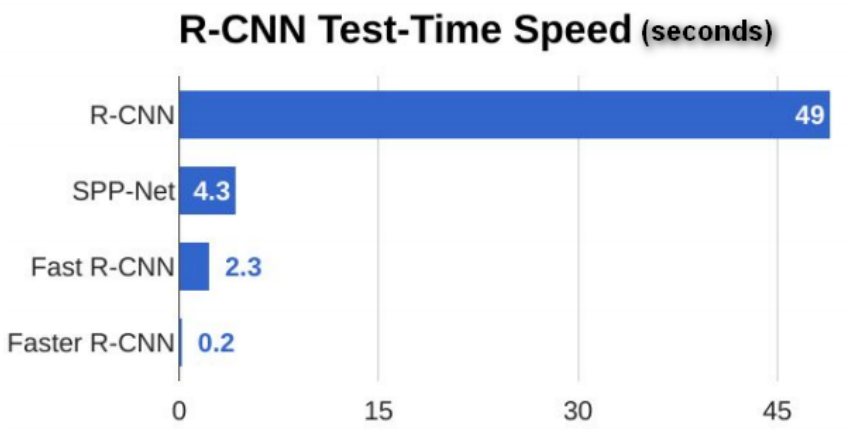
Fast R-CNN modeli R-CNN modeline nazaran 9 kat daha hızlı eğitim işlemi gerçekleştirmekte olup bunun yanı sıra test süresini yaklaşık olarak 213 kata kadar azaltmaktadır. Fast R-CNN modelinin hız yönünden sıkıntı yaşadığı alan ise bölge tekliflerinin üretilmesi aşamasıdır. Bu aşama yaklaşık olarak 2 saniyeyey yakın bir zamanla tamamlanmaktadır. Bu problemin çözümü Faster R-CNN modelinin geliştirilmesi ile sağlanmaya çalışılmıştır.

4.2.3. Faster R-CNN

Fast R-CNN modelinin en çok zaman harcadığı bölüm bölge tekliflerinin üretildiği bölümdür. R.Girshick ve arkadaşları bu problemin çözümü için Faster R-CNN modelini geliştirmişlerdir. Geliştirdikleri bu modelde konvolüsyon katmanları bölge teklifi üretimi ve nesne tespiti işlemlerini yerine getirmektedir. Faster R-CNN modelinin Fast R-CNN yapısından farkı bölge tekliflerinde kullanılan fonksiyonların bu modelde kullanılmamasıdır. Region Proposal Network (RPN) adı verilen konvolüsyon katmanı bölge tekliflerini üretme işlemini yerine getirmektedir. RPN yapısı, verilen giriş görüntüsünden elde edilen özellik haritasını alır. Bu bilgilerle her birinin nesne olma olasılığı ile nesnenin bulunması öngörülen dikdörtgen şeklindeki sınırlayıcı kutuların üretimini gerçekleştirir. RPN ve CNN bu işlemleri yaparken bazı katmanların ortak kullanımını gerçekleştiriler.

Faster R-CNN modelinin genel yapısını oluşturan ilk aşama hem bölge teklifi hemde nesne tespiti için kullanılacak olan ağların eğitilmesi işlemidir. Eğitimi tamamlanan bu ağlar birleştirilerek modelin ana yapısını oluşturan ağ ortaya çıkarılmıştır. Elde edilen bu ağ giriş verisi olarak bir resim almakta olup ortak kullanılan konvolüsyon katmanları ile giriş verisindeki resmin özelliklerinin elde edilmesi işlemi yapılmaktadır.

Bu elde edilen özellikler RPN ağına girdi olarak verilerek nesnelerin bulunabileceği bölge tekliflerinin üretim işlemi gerçekleştirilir. Sonuc olarak giriş görüntüsüne bağlı olarak elde edilen özellikler ve nesnenin bulunabileceği bölge teklifleri Fast-R-CNN modelindeki adımlardan da geçirilerek çıktılar elde edilmesi sağlanır. Faster R-CNN modelindeki konvolüsyon katmanlarının paylaşımı ile bölge teklifleri oldukça uygun maliyetli bir şekilde üretilmiş olmaktadır. Aşağıdaki Şekil 4.3'te Hızları ile bilgilere yer verilmiştir.



Şekil 4.3 Modellerin Hız karşılaştırması [34]

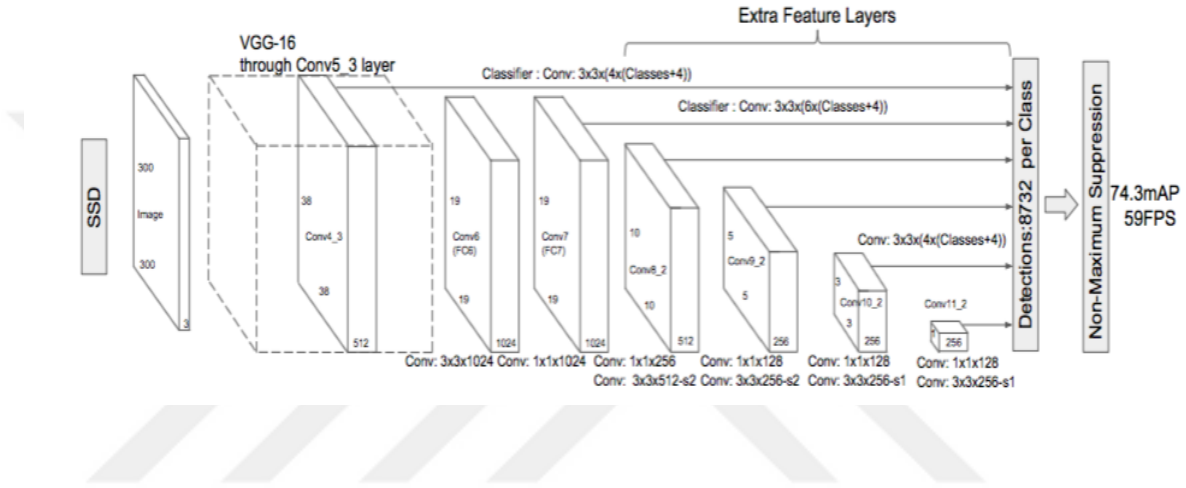
4.3. Tek Aşamalı Nesne Tespit Modelleri

Nesne tespiti uygulamalarında nesnelerin daha hızlı bir şekilde tespit edilmesi ihtiyacı gün geçtikçe önem kazanmaktadır. Bu nedenle iki aşamalı nesne tespiti modellerine göre çok daha hızlı nesne tespiti yapan tek aşamalı nesne tespit modelleri geliştirlmiş ve geliştirilmeye halen devam edilmektedir. Tek aşamalı nesne tanıma modellerinde iki aşamalı nesne tespit modellerinin aksine resmin tamamı bütün olarak ele alınır ve bu şekilde nesne tespiti yapılır. Tek aşamalı nesne tespit modelleri ise SSD, EfficienNet ve YOLO modellerini kapsamaktadır.

4.3.1. Single Shot Multibox Detector (SSD)

SSD modeli nesne tespiti işlemini gerçekleştirmek adına çözümlenecek bir regresyon probleminin faydalı olacağı üzerinde durmuştur. Nesneyi içeren bölge tekliflerinin yapılmamasını bunun yerine resmin bütün olarak ele alınmasının faydalı olacağı savunulmuştur. Oluşturulacak konvolüsyonel sinir ağı ile giriş görüntüsü tek bir defa işlenerek çok hızlı bir şekilde sonuç alınması hedeflenmiştir. SSD modelinin eğitiminin yapılabilmesi için giriş görüntülerinin yanı sıra nesnelerin yerlerini belirten sınırlayıcı kutu bilgilerine ihtiyaç duyulmaktadır. Girdi olarak verilen resim matris hücrelerine bölünür. Birbirinden farklı boyutlarda ayarlanan matrisler kullanılarak bu matrislerin içeriğindeki özelliklerin haritaları çıkarılır. Bu özellik haritalarından farklı en boy özelliklerine sahip sınırlayıcı kutular kullanılarak nesne tespit işlemi gerçekleştirilir.

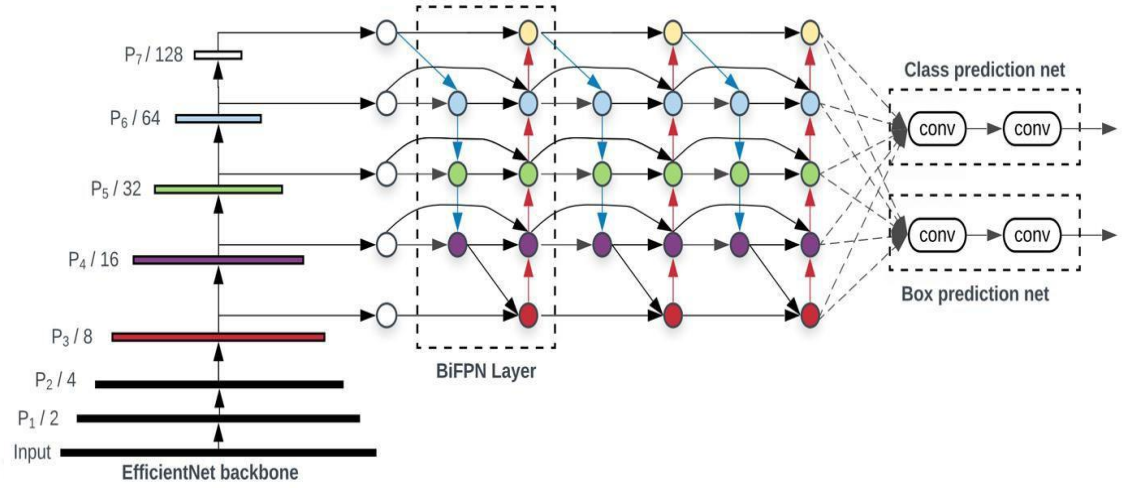
Farklı boyutlardaki bu sınırlayıcı kutular sayesinde farklı boyutlardaki nesnelerin tespit edilmesi işelmi de yapılmaktadır. Nesneyi içerisinde barındıran her sınırlayıcı kutu için hem her sınıf olasılık puanı hemde sınırlayıcı kutunun gerçekte olması gereken boyutunun hesaplanması sağlanır. Eğitimde ilk yapılacak işlem gerçekte olması gereken sınırlayıcı kutular ile varsayılan olarak belirlenen sınırlayıcı kutuların eşleştirilmesidir. Eğitim işleminin bu adımında doğru bir şekilde eşleşen kutlar pozitif tahminleri, yanlış eşleşen kutular ise negatif değerler olarak adlandırılır. Modelin hata oranı olasılık hatası ve lokalizasyon hatalarının ağırlıklarının toplamı şeklinde ortaya çıkar. Bu hata oranı istenilen seviyeye düşene kadar eğitim işlemi devam eder. Aşağıdaki Şekil 4.4.' te SSD modelinin genel yapısı gösterilmiştir.



Şekil 4.4 SSD modelinin genel Yapısı [35]

4.3.2. EfficientNet

Tek aşamalı nesne tespiti modellerinden biride EfficienNet modelidir. Bu modelde diğer tek aşamalı nesne tespiti modelleri gibi daha hızlı çalışıp sonuçlar elde ederken doğruluk oranının gelişimini ikinci planda tutmuştur. EfficienNet modelinin temelini hem yüksek doğruluk hemde yüksek verimlilik elde etmek için kullanılan kaynakların ölçekli bir şekilde kullanılması oluşturmaktadır. EfficienNet modelindeki en önemli farklılıklardan birincisi birbirinden farklı giriş görüntülerinin model içerisinde nasıl farklar yaratabileceğini öğrenmek adına öğrenilebilir ağırlıklar sunan iki yönlü özellik piramit ağının kullanılmasıdır. İkinci olarak geliştirilen modelin hem sınıf tahmini ağı olarak hemde omurga ağı olarak çözünürlük, derinlik ve genişlik şeklinde birleştirilmiş bir ölçekleme yöntemini kullanmasıdır. EfficienNet mimarisi daha önceden ImagaNet ile eğitilmiş olan EfficientNet omurga ağını kullanarak geliştirilmiş bir mimariye sahiptir. EfficienNet mimarisinin genel görünümü Şekil 4.5.'te gösterilmektedir.



Şekil 4.5 EfficienNet mimarisinin genel görünümü [36]

Şekil 4.5.'te belirtilen P1 ve P7 EfficienNet omurgası giriş görüntüsü haricinde toplamda 7 farklı seviyeden meydana gelmektedir. Bazı seviyelerde çıkarılan özelikler kendinden soraki katmanda defaten özellik füzyonu işlemine uğramaktadır. Bu seviyeler P3 ve P7 seviyeleride dâhil olmak üzere omurgayı oluşturan seviyelerdir. P3 ve P7 seviyeleri ile füzyon işlemleri sonucunda elde edilen birleştirilmiş özellikler, nesnenin hangi sınıfa dâhil olduğunun belirlendiği ve nesneyi çevreleyen kutu tahminlerinin üretildiği kutu ve sınıf ağının giriş verisini oluşturmaktadır.

Geliştirilen modelin ikinci en önemli özelliği olan birleşik ölçekleme özelliği Imagenet ile önceden eğitime tabi tutulmuş olup EfficienNet mimarisinde bulunan omurga ağı genişlik, çözünürlük ve derinlik değerlerini temel alan bir ölçeklendirme kullanmıştır. Kullanılan bu temel ölçeklendirme değerlerinden en verimli sonuçların elde edilmesi amacıyla genişlik değeri üssel, derinlik değeri ise doğrusal olarak artırılmış olup; sonuçta en verimli genişlik değerinin 1,35 olduğu belirlenmiştir. Bu ölçeklendirme parametrelerinden diğer bir tanesi olan çözünürlük ölçeklemesi için toplam 8 değer kullanılmıştır. Çözünürlük değerleri 540 pikselden başlayarak her bir sonraki değer için 2^7 oranında artırılmaktadır. EfficienNet modelinde kullanılabilecek en yüksek piksek değeri 1536 pikseldir.

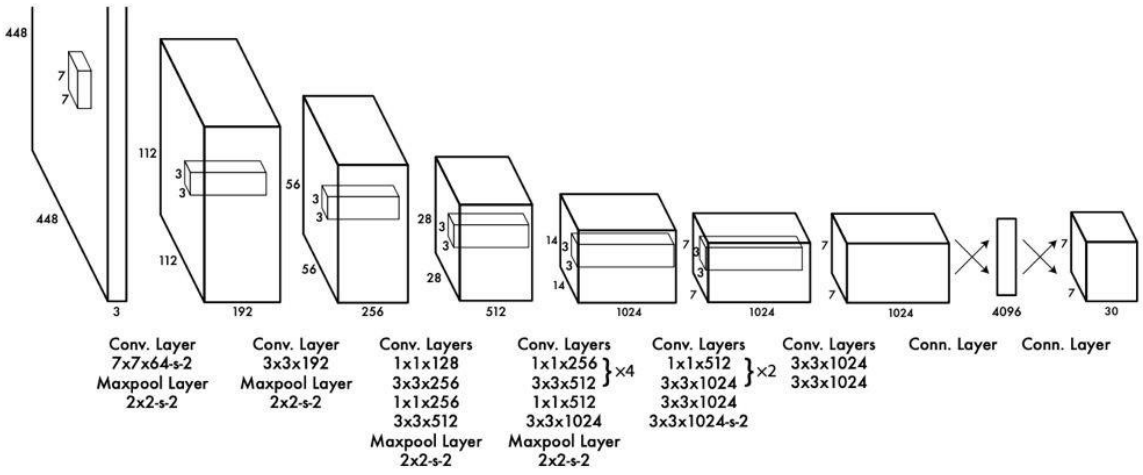
4.3.3. Yolo Modelleri

YOLO modeli nesne tespiti ve nesne takibi uygulamalarında başarılı sonuçların elde edildiği bir model yapısıdır. Diğer nesne tespiti modellerine göre en büyük avantajı nesne tespiti işlemini daha hızlı yapabiliyor olmasıdır. YOLO modelini hızlı yapan ise R-CNN veya Faster R-CNN gibi iki aşamalı nesne tespiti yapan modelerden farklı olarak girdi olarak verilen resim üzerinde yalnızca birkere işlem yapıyor olmasıdır. Mesela R-CNN modelinde ilk etapta girdi olarak verilen resimde tespit edilecek nesnenin bulunabileceği muhtemel yerler önerilir ve daha sonrasında önerilen bu yerlerden hangisinde nesne bulunduğuna bakılarak nesnenin sınıflandırılması işlemi gerçekleştirilmeye çalışılır. Geliştirilen YOLO modellerinde ise tespit edilecek nesnenin bulunabileceği herhangi bir yer önerisi yapılmadan verilen resim üzerinde yalnızca birkez işlem yapılarak nesnenin sınıfı ve bulunduğu yer bilgileri çıkarılmış olur. YOLO bu bilgileri çıkarırken bunu çözümünü bulduğu bir regresyon problem ile gerçekleştirmektedir. Bu regresyon işlemi resimdeki piksellerden nesneyi içeren bir sınırlayıcı kutu koordinatlarını ve nesnenin hangi sınıfa dâhil olacağının olasılıksal sınıflandırmasını bir arada yapmaktadır.

YOLO modeli ile oluşturulan evrişimsel sinir ağı girdi olarak verilen resim içeriğinde aynı anda birçok farklı nesneye ait sınıf olasılıklarını ve bu nesnelere ait sınırlayıcı kutuları bulmayı sağlar. Bu şekilde iki işlemi birlikte yerine getiren YOLO modeli hızının yanı sıra arka plan çıkarımında yapılan hataların daha az olması ve daha fazla genellenebilir çıkarımların elde edilmesi ile diğer modellere göre biraz daha fazla öne çıkmaktadır. Bunun yanında nesne tespitini hızlı yapmasına karşın tespitini yaptığı nesnenin sınıfını belirleyip sınırlayıcı kutusunun çizilmesi aşamasında daha az başarılıdır.

YOLO modelini oluşturan yapay sinir ağının ilk evrişimsel katmanları girdi olarak verilen resmin özniteliklerinin çıkarılmasını sağlarken tespiti yapılan nesnenin bulunabileceği sınıf olasılığı ve nesnenin resim içerisindeki koordinatlarının tahmin edilmesi işlemleri tam bağlı katmanlar tarafından gerçekleştirilir.

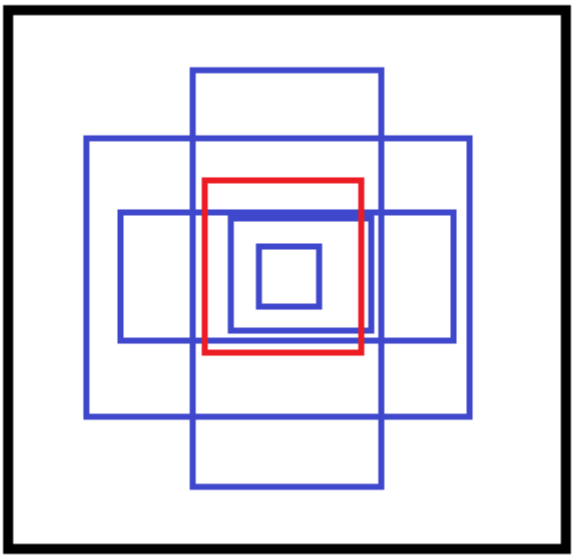
İlk olarak 2015 yılında ortaya atılan YOLO modeli GoogLeNet mimarisinden esinlenilerek geliştirilen bir model olarak ortaya çıkmıştır. Ortaya çıkan bu model kapsamında 24 evrişimsel katman ve 2 tam bağlantılı bulunmaktadır. Genel oalrak başarısının artırılabilmesi amacı ile giriş görüntüsü 224x224 piksel olarak ImageNet sınıflandırmasında önceden eğitilmiş daha sonra giriş resimleri 448x448 piksel olarak modele verilmiştir. Nesne içermeyen sınırlayıcı kutu tahminlerinin eğimi düşürmesini engellemek için nesne içeren kutular için kullanılan değişken değeri 5, nesne içermeyen kutular için kullanılan değişken değeri ise 0.5 olarak ayarlanmış ve bu şekilde toplam-kare hatası işlemine tabi tutulmuştur. VGG-16 ağı kullanılarak tasarlanan YOLO ve Faster R-CNN modelleri Pascal VOC 2007 ve 2012 veri seti kullanılarak eğitime tabi tutulmuş ve YOLO hem eğitim sürecinde hem de gerçek zamanlı tahminde daha hızlı olmasına karşın ortalama doğruluk oranı olarak daha az başarılıdır. Şekil 4.6'da ilk geliştirilen YOLO modeline ait mimari tasarım gösterilmiştir [37].



Şekil 4.6 YOLO Mimarisi [37]

Nesne tespiti işlemlerinde hız ve doğruluk faktörlerindeki üstünlüğün yanı sıra çeşitli sayıdaki nesnenin tespitinin yapılması önem kazanmaktadır. Geliştirilen ikinci Yolo modeli olan Yolov2 tüm evrişimli katmanlara Batch Normalizasyonu ekleyerek iyileştirmeler yapılmıştır. Bunun yanında Yolov2de kullanılan CNN mimarisinde nesne tespiti yalnızca tek sefer ileri besleme ile yapıldığından daha hızlıdır. Bu modelde ağ yapısı basitleştiirlerek temsili öğrenme gerçekleştirilmesi hedeflenmiş ayrıca toplu normalleştirme kullanılarak doğruluk oranı % 2 oranında artırılmmıştır. Yolov2 modeli eğitiminin ImageNet ve MS COCO veri setleri ile yapılması neticesinde 9000 nesnenin tespitinin yapılabildiği Yolo9000 modeli ortaya çıkmıştır.

Nesne tespiti modelleri genel olarak Imagenet tarafından eğitilmiş sınıflandırıcıları kullanmaktadır. Birçok sınıflandırıcı 256x256 boyutundan daha küçük boyutlara sahip giriş görüntüleri üzerinde işlemler yapar. Yolo sınıflandırma işlemi için 224x224 boyutunda görüntülerle çalışır ve sonrasında nesne tespiti için çözünürlüğü 448x448 boyutuna çıkararak işlem yapar. Modelin artan % 4 oranındaki doğruluğu giriş görüntüsünün boyutunun artırılması ile sağlanmıştır. Yolov2 modeli ile birlikte sınırlayıcı kutuları tahmin etmek amacı ile bağlantılı kutuları kullanmıştır. İlk modellerde kullanılan tamamen bağlı katmanlar çıkarılmıştır. Şekil 4.7'de sarı mavi renkli kutular bağlantılı kutulara örnektir.



Şekil 4.7 Mavi Renkli Bağlantılı Kutular

Ayrıca giriş görüntüsü 448x448 piksel yerine 416x416 piksel olarak ayarlanmış, büyük nesnelerin görüntüde merkezde yer alabileceği tek bir konum tahmin edecek ağ tasarımı yapılmıştır. YOLOv2 modelinin farklı bir özelliği de ağın her 10 rastgele partide görüntü boyutlarının 32x32 ile 608x608 arasında 32 değerinin katları olacak şekilde değiştirilmesidir. Bu şekilde ağ farklı görüntü boyutlarında iyi tahmin yapmayı öğrenmeye zorlanır. Bu sayede hız ve doğruluk olarak denge sağlanır. YOLOv2 modeli sınırlayıcı kutu koordinat tahmini ve yerelleştirmenin yanı sıra ortak nesnelerin nasıl sınıflandırılacağı gibi algılamaya özgü bilgileri de öğrenmek üzerine tasarlanmış bir algoritma olduğu için eğitim sırasında hem algılama hem de sınıflandırma kümelerinden veriler karıştırılmıştır. YOLOv2 modeli Pascal VOC 2007 ve 2012 veri seti üzerinde eğitilen Faster R-CNN ve SSD modeli ile karşılaştırıldığında hem daha hızlı hem daha doğru sonuçlar üretmiştir [38].

YOLOv2 modeli üzerinde birtakım geliştirmeler yapılarak ağ daha derin hale getirilmiş ve YOLOv3 modeli üretilmiştir. YOLOv3 modeli de diğer modellerde olduğu gibi ilk olarak görüntü üzerinde sınırlayıcı kutuyu çizer ve ardından kutu içerisinde bulunabilecek nesne sınıfını tanımlayabilmek için nesne algılamayı iki aşamalı bir problem durumu olarak ele almaktadır. YOLOv3 modeli özellik çıkarma ağı olarak Darknet-53 mimarisini kullanmaktadır. Geliştirme aşamasında YOLOv2'de kullanılan havuzlama katmanı çıkarılmıştır ve tespit edilen nesne büyüklüğüne göre kullanılan özellik haritasının büyüklüğü de değişmektedir. Geliştirilen modelin hem SSD hem de RetinaNet modeline göre daha hızlı ve daha başarılı olduğu tespit edilmiştir [39,40]. Şekil 4 .8'de YOLOv3 mimarisinde kullanılan Darknet-53 kütüphanesinin filtre sayısı, filtre boyutu ve çıktı boyutları gösterilmektedir.

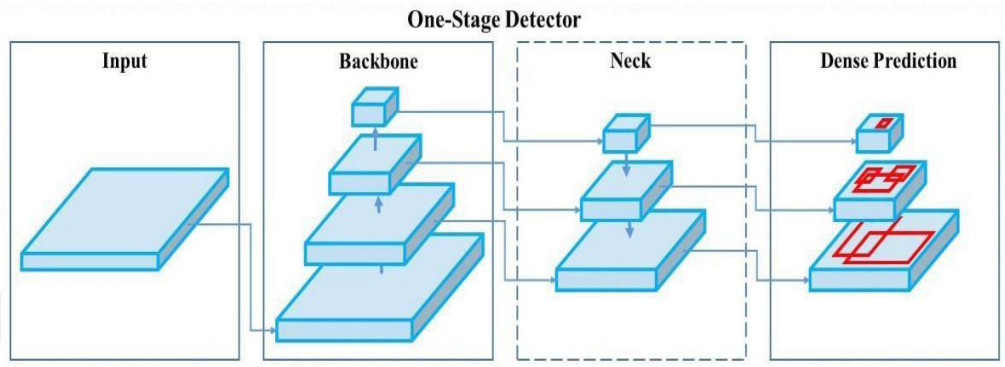
	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Şekil 4.8 YOLOv3 modelinin kullandığı Darknet-53 mimarisi [40].

2020 yılının nisan ayında Yolo modellerene bir yenisi daha eklendi Yolov4. Bu modelde farklı olan model eğitiminin tek bir GPU üzerinden gerçekleştiriliyor olabilmesinin yanı sıra Bag-of-Freebies ve Bag-Of-Specials şeklinde isimlendirilen tekniklerle performans artırımını sağlıyor olamasıdır.

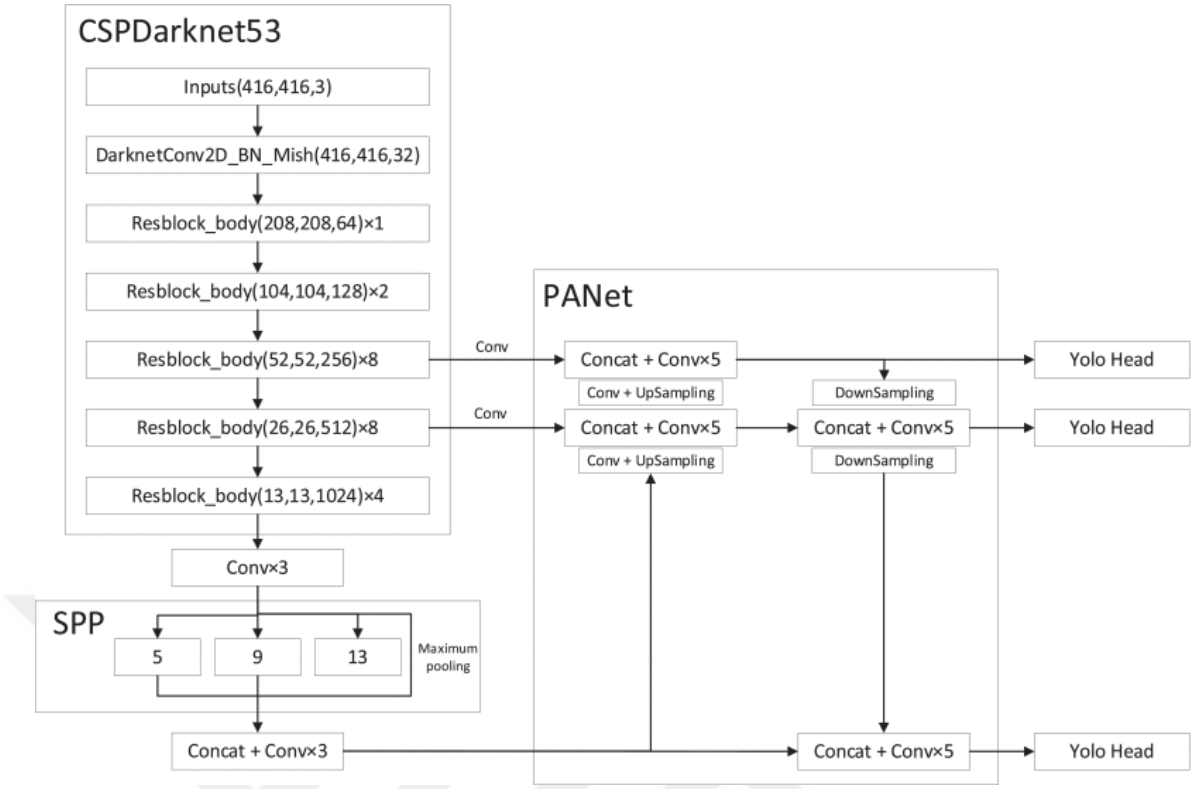
Bag-of-Freebies, modelin doğruluğunun geliştirilmesi amacı ile eğitim stratejisini veya eğitim maliyetini değiştiren tekniklerin genel adıdır. Bu teknikler içerisinde veri çoğaltma, anlamsal dağılım yanlılığını önleme çalışmaları ve iyileştirme adına kullanılan birçok BBox regresyonunun amaç fonksiyonları bulunmaktadır. Bag-Of-Specials ise farklı yöntemler kullanarak çıkarım maliyetini küçük bir oranda artıran ancak nesne algılayıcısının doğruluğunu büyük ölçüde artırabililen modülleri kapsamaktadır. Bu modüller Mish Aktivasyonu, CSP (Çapraz Aşamalı Kısmi Bağlantılar), FCN- Uzamsal Piramit Havuzu Oluşturma, SAM (Uzamsal Dikkat Modülü), PANet(Yol Toplama Ağları) ve DıoU NMS modülleri olarak tanımlanmaktadır.

Tek aşamalı nesne tespit uygulamalarının genel yapısı omurga olarak adlandırılan Backbone, omurga ile kafa bölümleri arasındaki boyun olarak adlandırılan neck ve kafa olarak tanımlanan head kısımlarından oluşur. Verilen giriş görüntüsü ile başlayan süreç head kısmında nesnenin sınıfının ve bulunduğu koordinatların tespiti ile nihai amacına ulaşmış olmaktadır. Tek aşamalı nesne tespiti yapan modellerin Şekil 4.9 da tek aşamalı nesne tespiti uygulamalarının genel yapısı gösterilmektedir.



Şekil 4.9 Tek aşamalı nesne tespiti genel yapısı [41].

Kendisi de tek aşamalı nesne tespit modeli olan Yolov4 de yukarıda belirtilen kısımları aynen içermektedir. Bu kısımlardan Backbone katmanında özellik çıkarımı işlemleri gerçekleştirilir. Yolov4 modeli Backbone kısmındaki işlemleri gerçekleştirmek için CSPDarknet53 modelinden faydalanır. Neck bölümü Backbone ve head olarak adlandırılan bölümler arasında bulunmaktadır. Ara katman olarak tanımlanan bu bölümün amacı nesne tespti yapılırken daha fazla bilgi temini yapmaktır. Yukarıdan aşağı veya aşağıdan yukarı akış ile temin edilen bilgilerden faydalanılarak komşu özellik haritalarından ayrıntılı bilgilerin çıkarımı sağlanmıştır. Diğer modellerden farklı olarak Yolov4 bu kısımda PAN ve SPP kullanmıştır. Yolov4 modelinin son kısmı olan head kısmında bounding box (Sınırlayıcı Kutular) mevcuttur ve bu mevcut sınırlayıcı kutuların sınıflarının tahmin edilmesi işlemi bu kısımda gerçekleştirilir. Yolov4 modelinin genel yapısı şekil 4.10. ’da gösterildiği gibidir.



Şekil 4.10 YOLOv4 genel yapısı [42].

Yolov5 modelinin ilk sürümü 29 Haziran 2020 tarihinde Glen Jocher tarafından yaynlanmıştır.Yolov5s, Yolov5m, Yolov5l ve Yolov5x şeklinde faklı modelleri içermektedir. Yolov5 modeli Darknet kütüphanesinde geliştirilen Yolov4 modelinin PyTorch Kütüphanesi üzerinde gerçekleştirilmesi ile oluşturulan bir modeldir. Diğer tek aşamalı nesne tespiti modellerinde olduğu gibi Yolov5 modeli de baş, boyun ve omurga kısımlarından meydana gelmektedir. Baş ve boyun kısımları Yolov4 modeli ile aynı bölümlerden oluşmaktadır.

Yolov5 modelinin en büyük farkı önceki modellerde kullanılan Darknet yapısından farklı olarak PyTorch yapısını kullanmasıdır.[43] PyTorch ekosisteminin dağıtımı daha kolay ve destek olanağı daha basittir. Özellikle mobil cihazlara dağıtımının daha basit olması en yeni özelliklerinden biridir. Yolov5 modeli için oluşturulan ağırlık dosyaları diğer yolo modellerine göre oldukça küçük boyuttadır. Yolov5 için bir ağırlık dosyası 27 megabaytken Yolov4 modelindeki ağırlık dosyası 244 megabayttır.Yolov5 modeli bu özelliği sayesinde gömülü cihazlara daha kolay uygulanabilir olduğunu göstermiştir.

5. VERİ SETİ VE UYGULAMANIN GERÇEKLEŞTİRİLMESİ

5.1 Programlama Dili ve Geliştirme Ortamı

Yazılım geliştirmek amacıyla kullanılan birçok programlama dili mevcuttur. Gerçekleştirilmek istenilen derin öğrenme modeli ile ilgili olarak ilk etapta yapay nöronların oluşturulabileceği, arka planda birçok matematiksel işlemin rahatlıkla gerçekleştirebileceği ve üzerinde sürekli geliştirmeler yapılan kütüphanelere sahip bir programlama dilinin seçimi önem arz etmektedir. Bu sebeple gerçekleştirilecek olan derin öğrenme modeli için Python programlama dili kullanılması uygun görülmüştür.

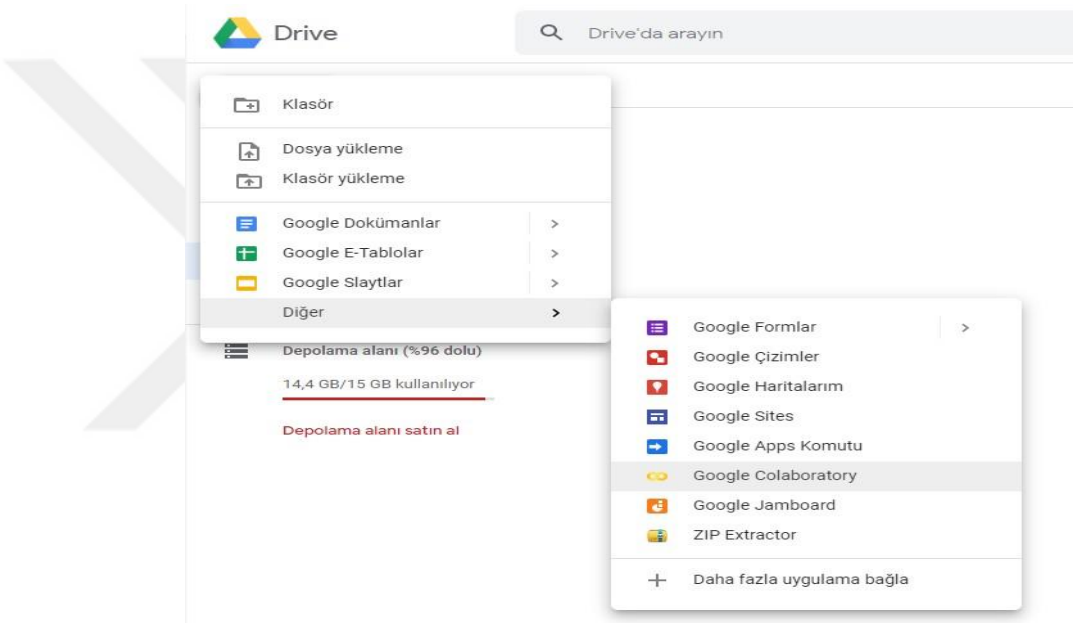
Geliştirilecek derin öğrenme modeli için yerel ve bulut ortamı olarak sınıflandırabileceğimiz geliştirme ortamları mevcuttur. Derin öğrenme modeli gerçekleştirilirken yüksek hacimli verilerle işlemler yapılacağından yerelde kullanılacak olan CPU veya düşük kapasiteli GPU donanımlarına sahip yerel makinaların hem zaman yönünden hem de maliyetli olması bakımından değerlendirildiğinde kullanımının uygun olmayacağı değerlendirilmektedir. Bunun yanı sıra ücretli veya ücretsiz olarak kullanılabilen, yüksek hacimli veriler üzerinde birçok işlemin yerel makinalara göre çok daha hızlı yapılmasına olanak sağlayan Google Colab, Kaggle Notebook ve FloydHub gibi bulut ortamları mevcuttur. Uygulamanın gerçekleştirilmesi amacı ile bulut ortamında Python programlama dilinde uygulama geliştirmeye imkân sağlayan Google Colab kullanılacaktır.

5.1.1 Python Programlama Dili

Python programlama dili 1991 yılında geliştirilmeye başlanan bir programlama dilidir. Python programlama dilinin simgesi sanıldığının aksine piton yılanından esinlenerek değil geliştiricisi Guido Van Rossum bir komedi gösterisinden esinlenmesi ile oluşmuştur. Python programlama dili kolay öğrenilen ve kolay okunan bir dildir. Basit söz dizimine sahip olmasının yanı sıra açık kaynaklı bir dildir. Derlemeye gerek olmadan çalıştırılabilir ve Object Oriented Programming (OOP) özellikleri sayesinde sınıflar ve nesneler oluşturarak kolay bir şekilde yazılım geliştirmeye imkân sağlar. Python programlama dili, NumPy, Pandas, Keras, Scikit-Learn, PyTorch gibi kapsamlı kütüphane desteğiyle özel olarak belirlenecek uygulamaların gerçekleştirilmesine büyük faydalar sağlamaktadır.

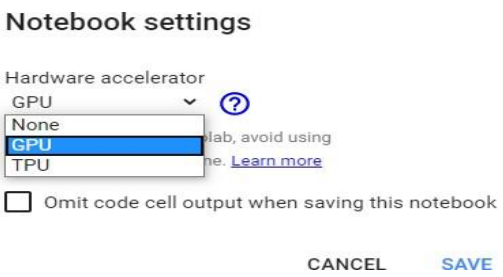
5.1.2 Google Colab

Geliştirilecek olan derin öğrenme modelinde yüksek hacimli verilerle işlem yapılacağından donanım olarak hızlı CPU hatta mümkünse GPU kullanımı kaçınılmazdır. Yüksek hacimli bu verilerle yapılacak işlemlerde maliyet ve zaman açısından bu donanımlar yerine Google şirketinin kendi bünyesinde ücretsiz olarak hizmete sunduğu GPU ve derin öğrenme için özel olarak üretilmiş TPU (Tensor Process Units) desteğiyle bulut tabanlı bir derin öğrenme geliştirme ortamı oluşturarak uygulama geliştirmeye imkân tanıyan Google Colab ortamının kullanılması uygun görülmüştür. Google hesabı ile giriş yapılan Drive bölümünde yeni sekmesi altında Google Colaboratory sekmesi açılarak yeni .ipyn uzantılı notebook dosyası oluşturulur. Şekil 5.1 de görselleştirilmiştir.



Şekil 5.1 Yeni Colab Ortamı oluşturma

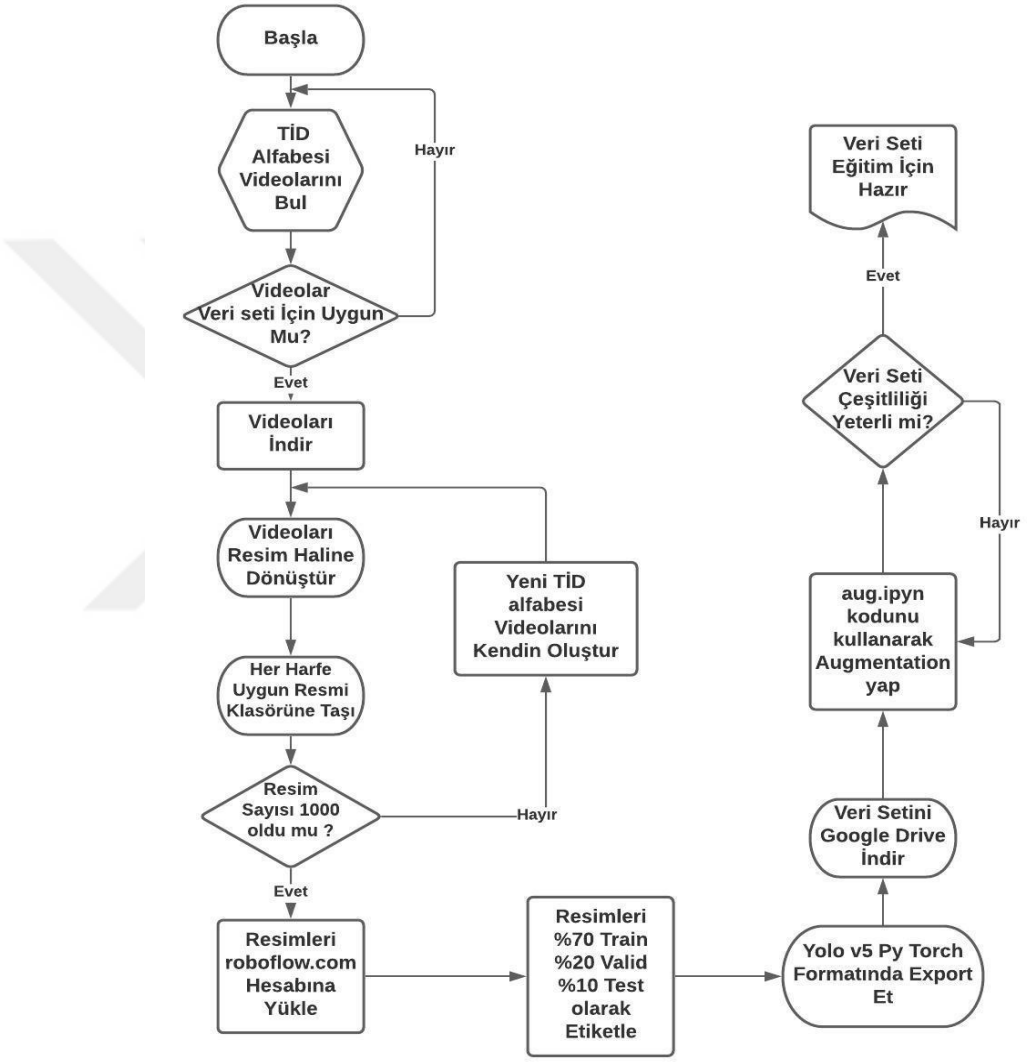
Kullanılacak donanım seçimi için Runtime sekmesi altındaki Change Runtime Typetıklanarak GPU veya TPU seçimlerinden biri yapılabilmektedir. Donanım seçimi işleminden sonra kod yazma işlemine geçilebilir. Şekil 5.2 de görselleştirilmiştir.



Şekil 5.2 Kullanılacak donanım seçimi

5.2 Veri Setinin Hazırlanması

Türk İşaret dili bilgisayar bilimleri açısından yeni ve bakir bir alandır. Derin öğrenme yöntemlerinin gelişmesiyle birlikte Türk İşaret dilinin tanınmasına yönelik olarak veri setleri de oluşturulmaya başlanmıştır ancak nesne tespitinde kullanıma uygun bir veri setine rastlanılmadığından yenin bir veri seti oluşturulmasına karar verilmiştir. Yeni veri seti alfabemizde bulunan 29 harfe ait sınıflardan oluşmaktadır. Hazırlanan veri setine ait akış şeması Şekil 5.3.' de gösterilmiştir.



Şekil 5.3 Veri Setine Ait Akış Şeması

5.2.1 TİD Alfabesi Videoları

Türk İşaret Dilinin herkes tarafından öğrenilmesi ve insanlara kaynak oluşturması adına internet ortamında herkese açık şekilde paylaşılan birçok video içeriğinin olduğu görülmüştür. Hazırlanacak projenin de insanlara fayda sağlayacağı düşünüldüğünden herkese açık olarak paylaşılan bu videoları kullanarak veri seti elde etmenin uygun olacağı ve veri çeşitliliği ile birlikte daha kapsamlı bir veri seti oluşturulmasına karar verilmiştir. Özellikle Youtube üzerinden paylaşılan videoların içeriği kontrol edildiğinde Türk İşaret Dili alfabesini oluşturan harflerin çeşitli yaş grubundaki kadın erkek hatta çocuklar tarafından yapıldığı görülmüştür. Gerçekleştirilecek proje için bu videolardan veri seti oluşturmaya uygun olan videolar tespit edilmiştir. Toplamda 15 ayrı kişiye ait çeşitli çözünürlükte, çeşitli uzunluklarda ve çok çeşitli arkaplanlara sahip videolar indirilmiştir. İndirilen videoalrın toplam boyutu 446 MB büyüklüğündedir.

5.2.2 TİD Vidolarından Görüntülerin Seçilmesi

Videolar resimlerin arka arkaya sıralanmasıyla oluşan yapılardır. Bir önceki bölümde belirtilen ve 15 farklı kişiye ait olan her bir video dosyasından görüntülerin elde edilmesi işlemi için Lenovo marka özellikleri Tablo 5.1'de belirtilen dizüstü bilgisayar kullanılmıştır. Hazırlanan Python kodu ile videoyu oluşturan görüntülerin belirli aralıklarla kayıt işlemi yapılarak Türk İşaret Dili alfabesini oluşturan görüntüler elde edilmiştir. Elde edilen bu görüntülerden uygun olanlar Türk İşaret Dili alfabesindeki her harf için ayrı ayrı hazırlanan klasörler içerisine taşınmıştır. İndirien videolardan istenilen sayıda görüntünün temin edilemediği durumlarda özellikleri Tablo 5.1'de belirtilen dizüstü bilgisayar ve kamerası kullanılarak Türk İşaret Dili harflerini içeren videolar tarafımca oluşturulmuştur.

Tablo 5.1 Veri seti için kullanılacak sistemin özellikleri

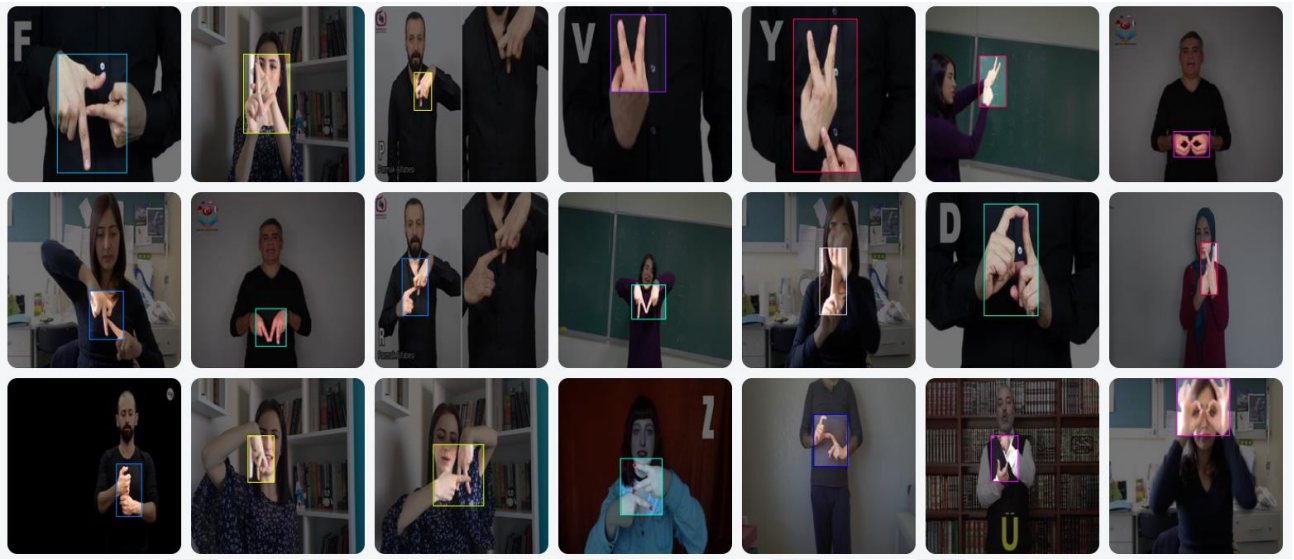
İşletim sistemi	Windows 10 Home Single Language
İşlemci	Intel(R) Core (TM) İ7-9750H CPU@ 2,60 GHz
Ram	16.0 GB
Sistem türü	64 Bit İşletim Sistemi, x64 tabanlı işlemci

5.2.3 Resimlerin Etiketlenmesi ve Export İşlemi

Veri seti oluşturulurken veri seti hazırlama konusunda birçok kolaylık barındırması nedeniyle https://roboflow.com/ isimli internet sitesi kullanılmıştır. Ücretsiz olarak 1000 adet görüntüyü etiketlemeyi ve etiketlenerek hazırlanan bu veri setlerini istenilen formatta dışa aktarım (Export) yapmayı sağlayan bu internet sitesi üzerinden veri seti oluşturulması uygun bulunmuştur. Bu sebeple Türk İşaret Dili alfabesini oluşturan görüntülerden toplamda 1000 adet olacak şekilde klasörleme işlemi yapılmış olup veri setinin homojen olması amacıyla her harften en az 34 adet görüntünün olmasına özen gösterilmiştir. Şekil 5.4 harflerin veri setindeki dağılımı göstrilmiştir.

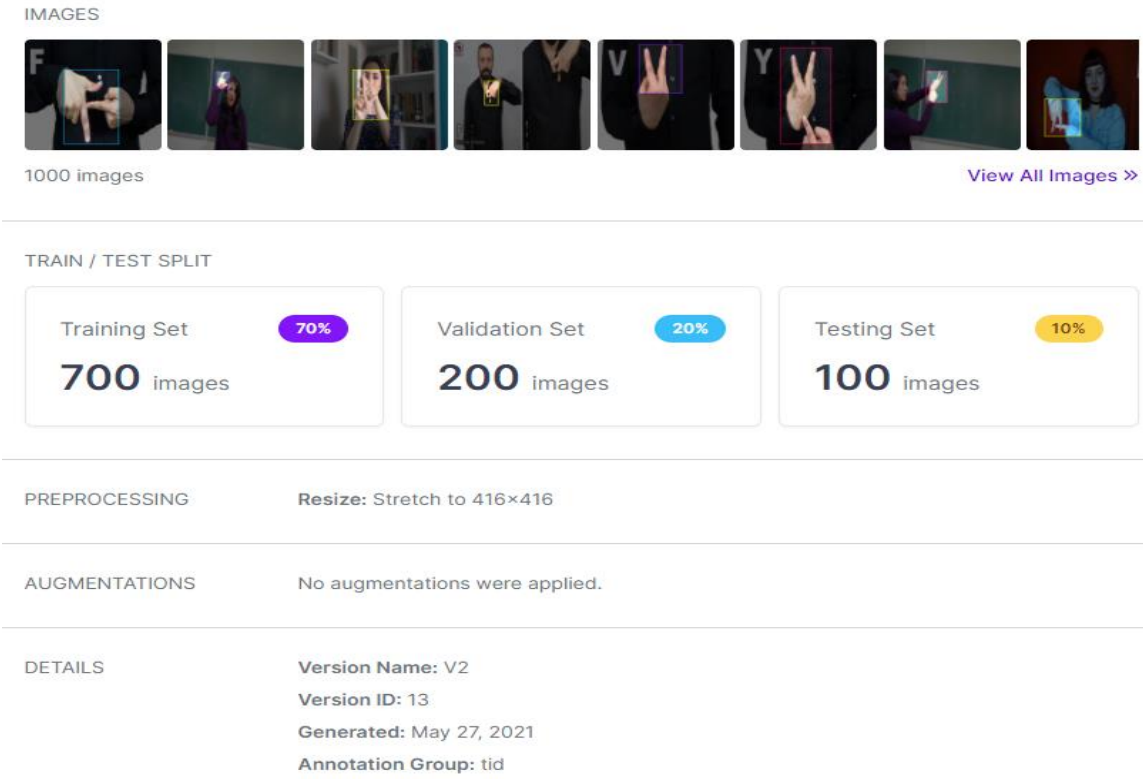
Şekil 5.4 Veri Setindeki Harflerin Sayısı

Basit bir kayıt işlemi sonrasında https://roboflow.com/ isimli internet sitesi üzerinden veri seti oluşturma işlemi gerçekleştirilebilmektedir. Veri seti oluşturulurken her bir klasör ayrı ayrı https://roboflow.com/ isimli internet sitesine yüklenerek her harfin etiketleme işlemi gerçekleştirilmiştir. Türkçe karakter desteği olmadığından 'Ç' harfi 'CCC' şeklinde , 'İ' harfi 'III' şeklinde, 'Ğ' harfi 'GGG' şeklinde, 'Ö' harfi 'OOO'şeklinde, 'Ş' harfi 'SSS'şeklinde ve 'Ü' harfi 'UUU'şeklinde etiketlendirilmiştir. Etiketlenen resimlere ait görsellerden bir bölümü Şekil 5.5'te gösterilmektedir.



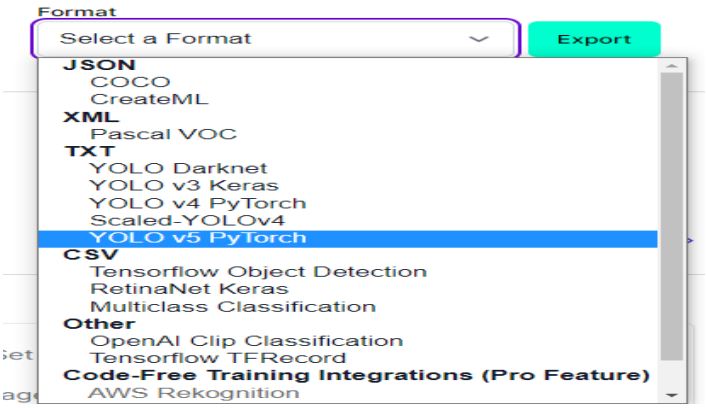
Şekil 5.5 Etiketli Resimlere Ait Görseller

Etiketlemesi tamamlanan her harf sonrasında veri seti %70 Train , %20 Valid ve %10 Test verisi olacak şekilde organize edilmiştir. Toplamda 1000 adet görüntü 416X416 boyutunda olacak şekilde resize edilerek veri seti Export işlemine hazır bir şekle getirilmiştir. Şekil 5.6 organize edilen veri setine ait bilgilere yer verilmiştir.



Şekil 5.6 Veri Setine Ait bilgiler

Hazırlanan veri setinin Şekil 5.7 de belirtilen formatlara uygun olarak export işlemi gerçekleştirilebilmesi mümkündür. Gerçekleştirilen projede Yolov5 PyTorch kullanılacağından bu formatta export işlemi gerçekleştirilmiştir.



Şekil 5.7 Export Formatları

5.2.4 Veri Çoğaltma

Görüntü işleme alanında oluşturulan veri setlerindeki görüntülerin sayısının ve çeşidinin fazla olması hazırlanacak modelin başarımı açısından büyük önem taşımaktadır. Veri artırma ile hazırlanan kısıtlı miktardaki verilerden en iyi şekilde yararlanılması hedeflenmektedir. Hazırlanan kısıtlı miktardaki veri setlerinin hayatın olağan akışındaki tüm olasılıklarla tespit edilmiş görüntüler olmasını beklemek doğru olmayacaktır. Bu sebeple veri setini oluşturan görüntüleri oluşabilecek diğer olasılıktaki durumara genelleyebilmek adına ayarlamalar yapılmaktadır. Yapılan bu ayarlamalar ile eğitilecek modelin günlük hayatın akışında karşılaşabileceği durumlar dizisinin daha geniş tutulması sağlanarak modelin başarımının artmasına katkı sağlayacaktır. Bu ayarlamalar içerisinde fotometrik bozulmaların yapıldığı yani orijinal görüntü üzerindeki parlaklık, doygunluk, karşıtlık ve parazit değişimlerinin yapıldığı yöntemler mevcuttur. Başka bir ayarlama yöntemi giriş görüntüsü üzerinde yapılan geometrik ayarlamaları kapsamaktadır. Belirtilen iki yöntemde pixseller üzerinde gerçekleştirilen ayarlamlar ile gerçekleştirilmektedir.

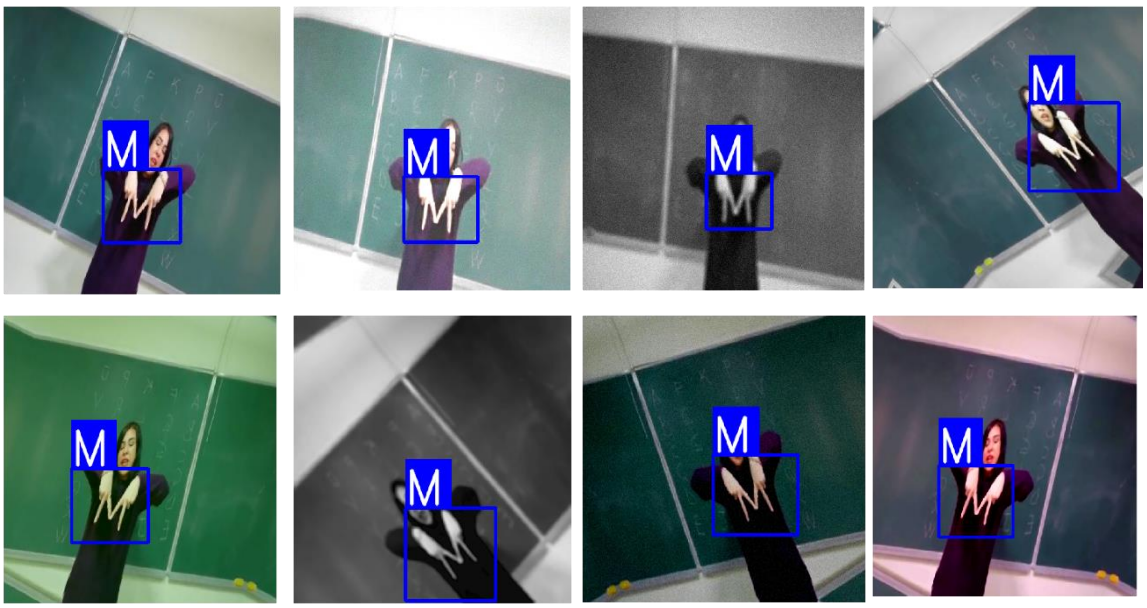
Bazı yöntemler ise giriş görüntüsü içeriğinde yapılan değişiklikleri içermektedir. Rastgele silme, kesme, gizle ve ara, ızgara maskesi ve Mixup bu yöntemlerdendir. Rastgele silme ile orijinal görüntü üzerinden rastgele değerlerle vey eğitim setinin ortalama piksek değerini değiştiren bir veri büyütme tekniğidir. Rastgele oranlarda silinen görüntü ve silinen alanın en-boy oranıyla uygulanır. Kesme yönteminde kare bölgeler eğitim sırasında maskelenir. Rastgele silmeye benzerdir fakat kesme yönteminde kesilecek alanla ilgili sabit bir değer mevcuttur. Gizle ve ara yönteminde giriş görüntüsü SxS yamalardan oluşan bir ızgaraya bölünür ve bu ızgaralardan bazılarının silinmesi sağlanır. Böylelikle tespit edilecek nesnenin yalnızca belirli bir bölümünün neye benzediğinin tespitinin yapılması hedeflenmektedir. Izgara maskesi yönteminde orijinal görüntü ızgara benzeri şekilde gizlenir. Bu yöntem ile tek bir nesne içeren görüntülerde tespit edilmesi hedeflenen nesnenin belirli parçalarından çıkarım yapılması hedeflenir. MixUp yöntemi ise görüntü çiftlerinin ve etiketlerinin dışbükey olarak üst üste bindirilmesiyle gerçekleştirilmektedir.

Geliştirilen Yolov4 modeli ile görüntü işleme alanındaki en büyük sorunlardan biri olan veri seti ile ilgili problemlerin çözülmesinde önemli katkılar sağlamıştır. Yolov4 modelinde kullanılan bazı veri artırma teknikleri CutMix, Mozaik veri artırma, sınıf etşketi yumuşatma ve Kendi kendine çekişmeli eğitim yöntemlerini içermektedir.

Oluşturulan veri seti Google Drive içerisine Yolov5 PyTorch formatında export edildikten sonra yukarıda bahsedilen yöntemlerden parlaklık, doygunluk, kontrast, parazit değişimleri ve giriş görüntüsü üzerinde yapılan geometrik ayarlamaların yapılması amacıyla hazırlanan python kodu ile veri artırımı işlemi gerçekleştirilmiştir. Gerçekleştirilen bu veri artırma yöntemleri sonrasında toplam 10000 etiketli resim verisi elde edilmiştir. Veri artırma işlemi sonrasında projede kullanılmak üzere hazır hale getirilen veri setine ait özellikler Tablo 5.2'de gösterilmiş ayrıca veri artırımının resimler üzerindeki etkisini göstermek adına "M" harfine ait veri artırımı örnekleri ise Şekil 5.8'de belirtilmiştir.

Tablo 5.2 Veri setine ait özellikler

Veri setinde toplam sınıf sayısı	29
Veri setindeki toplam görüntü sayısı	10000
Train %70 Görüntü Sayısı	7000
Valid %20 Görüntü Sayısı	2000
Test %10 Görüntü Sayısı	1000
Değişkenler	Uzaklık –Parlaklık–Kontrast-Renk-Doygunluk-Gürülütü(Parazit) - Yaş - Cinsiyet- Arka Plan-Ten Rengi Farkı
Veri seti dosyasının büyüklüğü	446 MB



Şekil 5.8 "M" Harfi Veri artırım Örnekleri

5.3 Eğitim Transferi ve Nesne Tanıma Modelinin Model Seçimi

5.3.1 Eğitim Transferi

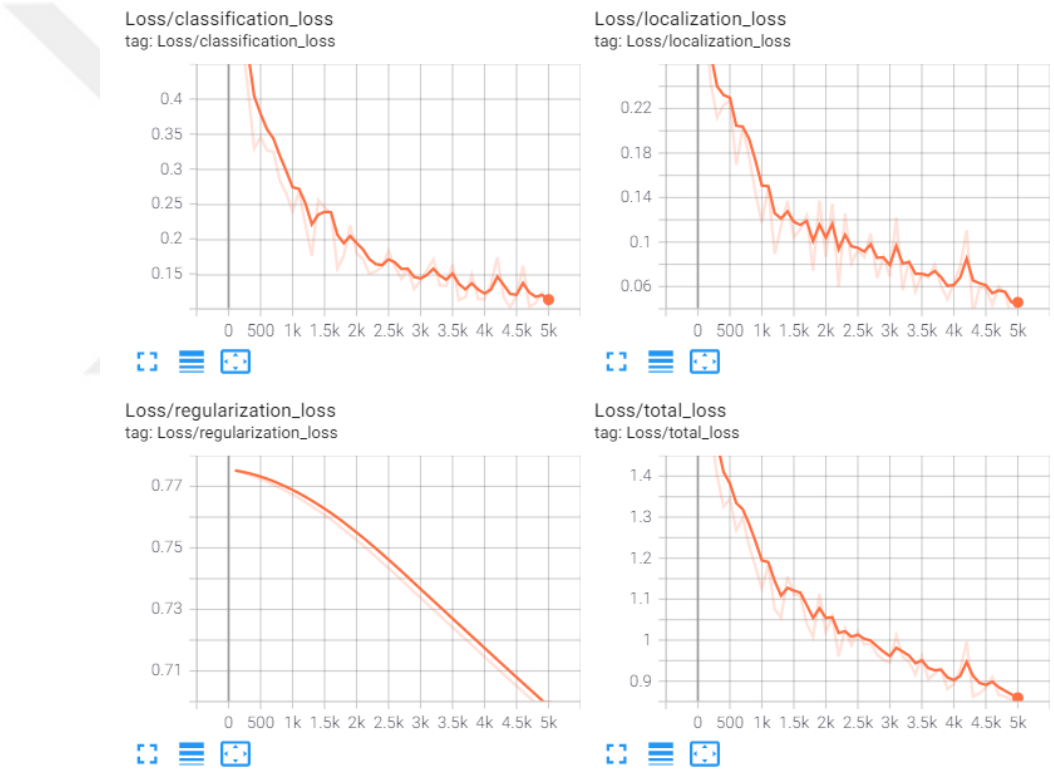
Eğtim transferi işlemi günlük hayattaki tecrübe kavramıyla benzer bir yapıya sahiptir. Tecrübe ise insanların yaşadıkları süre boyunca karşılaştıkları durumlardan öğrendiği bilgilerin tümü olarak ifade edilebilir. Bu sebepledir ki insanoğlu karşısına gelen yeni problemleri çözerken geçmişte edinmiş oldukları tecrübelerden faydalanmayı da yararlı bulurlar. Eğitim transferi yöntemi de daha önceden çeşitli değişkenlere göre eğitilmiş olan modellerin tecrübelerinden faydalanarak yeni problemlerin çözümünde kullanılması şeklinde tanımlanabilir. Eğitim transferi kavramını bir örnek ile tanımlayacak olursak; meyvelerin tanınması amacıyla oluşturulan ve yüksek başarım elde edilmiş bir model, sportif faaliyetlerde kullanılan topların çeşidinin tanınması amacıyla oluşturulacak bir sistemde eğitim transferi yöntemi sayesinde kullanılabilmektedir.

Yeni bir uygulama oluşturmak için uygun özelliklere sahip bir ağ tasarımının yapılması, yeni veri seti ile çok daha uzun süren eğitim işlemlerinin yapılması ve oluşturulacak ağ yapısının istenilen düzeyde başarı sağlaması hem yüksek işlem gücü gerektirmekte hemde çok uzun süren eğitim işlemlerinin yapılmasına neden olmaktadır. Bu gibi değişkenler nedeniyle eğitim transferi yöntemi son zamanlarda geliştirilen uygulamalarda sıklıkla kullanılmaktadır. Eğitim transferi yöntemi üç farklı şekilde kullanılmaktadır.

Daha önceden eğitilmiş bir yapay sinir ağına ait ağırlık bilgilerinin değiştirilmeden yalnızca yapay sinir ağının sonunda bulunan sınıflandırma amacıyla kullanılan katman yeniden eğitime tabi tutuluyorsa bu eğitim transferi yönteminin birinci şeklini oluşturmaktadır. Eğitim transferi yönteminin ikinci şeklinde daha önceden eğitilmiş olan yapay sinir ağının son katmanında yapılan değişikliklerin yanı sıra kullanılan yeni veri seti ile yapay sinir ağındaki bazı katmanlarında mevcut olan ağırlıklarında güncellenmesi ile gerçekleştirilmektedir. Eğitim transferi yönteminin üçüncü şeklinde ise daha önceden eğitilmiş olan yapay sinir ağı modellerinin doğrudan kullanılmasıyla gerçekleştirilmektedir. Günümüzde birçok farklı mecrada daha önceden eğitilmiş olan yapay sinir ağı modelleri ve bu modellere ait ağırlıkların paylaşımı yapılmaktadır. Paylaşılan bu veriler gerçekleştirilecek olan yeni modeller için sınıflandırma ve özellik çıkarımında kullanılabilinmektedir. Eğitim transferi yöntemi ile tek aşamalı nesne tespiti yapan modellerden SSD-Mobilnet, Efficiendet-D0 ve YOLOv5s modelleri hazırlanan veri seti eğitime tabi tutulmuştur. Tek aşamalı modellerin kullanılmasının nedeni iki aşamalı modellere göre hız açısından daha verimli sonuçların elde edilmiş olmasıdır.

5.3.2 SSD Mobilenet

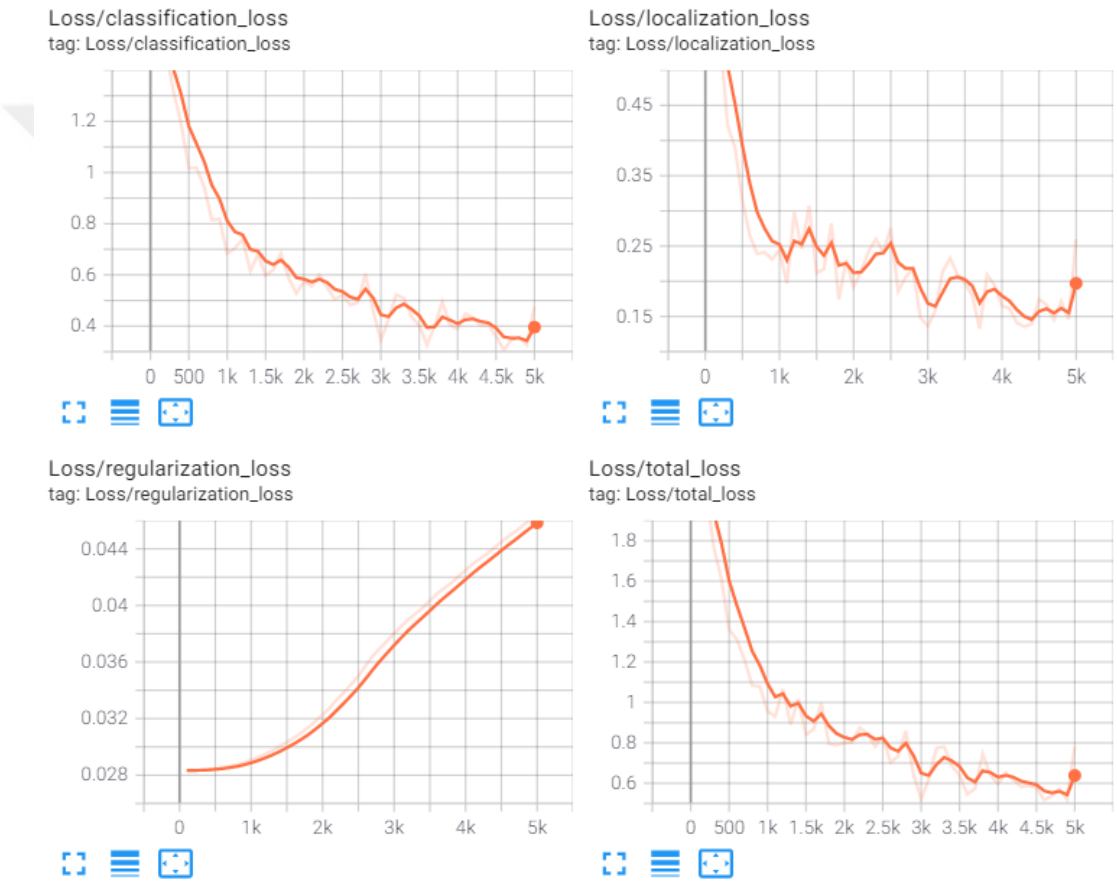
SSD modellerinin genel yapısı bölüm dörtte anılmış olup; Eğitim transferi yöntemi kullanılarak hazırlanan ilk model SSD-Mobilenet-v1 modelidir. Geliştirilen model nesne tanıma ve sınıflandırma alanında birçok eğitilmiş ağ yapısını içerisinde barındıran Tensorflow API ile oluşturulmuştur. Yapılan düzenleme ile record ve .pbtxt şeklindeki dosyalar oluşturulmuştur. Oluşturulan modele giriş verisi olarak 7000 adet train ve 2000 adet valid görüntüsü verilmiştir. Modelde giriş görüntüleri tekrar resize işlemine tabi tutularak giriş verilerinin boyutu 640x640 olacak şekilde model tarafından ayarlanmış, Batch Size 14 olarak belirlenmiş ve 5000 adımda eğitim işleminin yapılması sağlanmıştır. Toplam olarak 2 saat 5 dakika süren eğitim işlemi Google Colab ortamında Tesla T4 GPU kullanılarak gerçekleştirilmiştir. Eğitim sonucunda hesaplanan toplam kayıp değeri 0,84 olarak hesaplanmış ve diğer kayıp değerleri il birlikte aşağıdaki Şekil 5.9 da gösterilmektedir.



Şekil 5.9 SSD Mobilenet Kayıp Değerleri

5.3.3 EfficienNet-D0

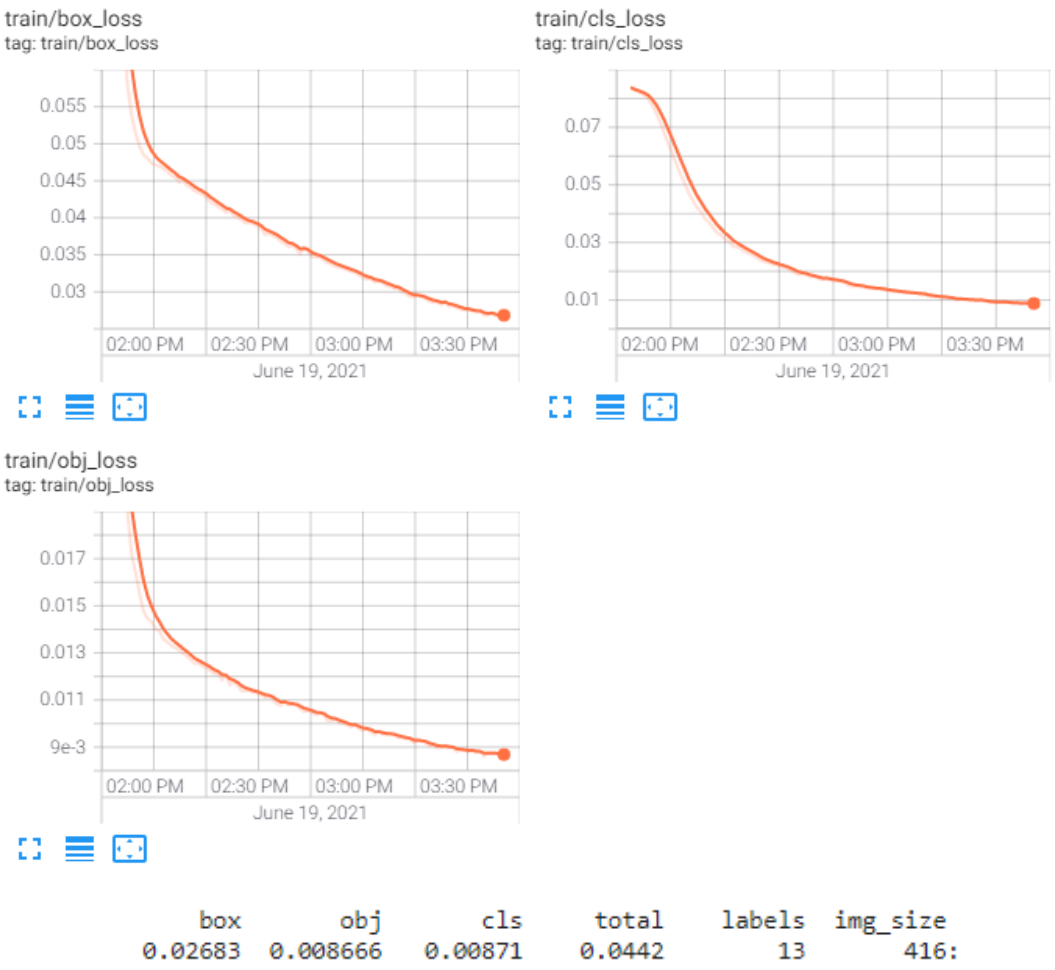
EfficienNet modelinin genel yapısı bölüm dörtte anılmış olup; Eğitim transferi yöntemi kullanılarak hazırlanan bu model EfficienNet D-0 modelidir. Bu model de Tensorflow API ile oluşturulmuş ve bir önceki modelde oluşturulan veri seti dosyaları aynı şekilde bu mdeolde de kullanılmıştır. Modelde giriş görüntüleri tekrar resize işlemine tabi tutularak giriş verilerinin boyutu 512x512 olacak şekilde model tarafından ayarlanmış, Batch Size 16 olarak belirlenmiş ve 5000 adımda eğitim işleminin yapılması sağlanmıştır. Toplam olarak 2 saat süren eğitim işlemi Google Colab ortamında Tesla T4 GPU kullanılarak gerçekleştirilmiştir. Eğitim sonucunda hesaplanan toplam kayıp değeri 0,78 olarak hesaplanmış ve diğer kayıp değerleri ile birlikte aşağıdaki Şekil 5.10 da ayrı ayrı gösterilmektedir.



Şekil 5.10 EfficienNet D-0 Kayıp Değerleri

5.3.4 YOLOv5s

YOLOv5 modelinin genel yapısı bölüm dörtte anılmış olup; Eğitim transferi yöntemi kullanılarak hazırlanan bu model YOLOv5s modelidir. Bu model Pytorch kütüphanesini kullandığı için organize edilen veri seti dosyaları .txt uzantılarına sahip olmaktadır. Modelde giriş görüntüleri 416x416 olacak şekilde ayarlanmış, Batch Size 16 olarak belirlenmiş ve 100 epoch eğitim işleminin yapılması sağlanmıştır. Toplam olarak 1 saat 59 dakika ve 47 saniye süren eğitim işlemi Google Colab ortamında Tesla T4 GPU kullanılarak gerçekleştirilmiştir. Eğitim sonucunda hesaplanan toplam kayıp değeri 0,04 olarak hesaplanmış ve diğer kayıp değerleri ile birlikte aşağıdaki Şekil 5.11 de ayrı ayrı gösterilmektedir.



box	obj	cls	total	labels	img_size
0.02683	0.008666	0.00871	0.0442	13	416:

Şekil 5.11 YOLOv5s Kayıp Değerleri

5.3.5 Model seçimi

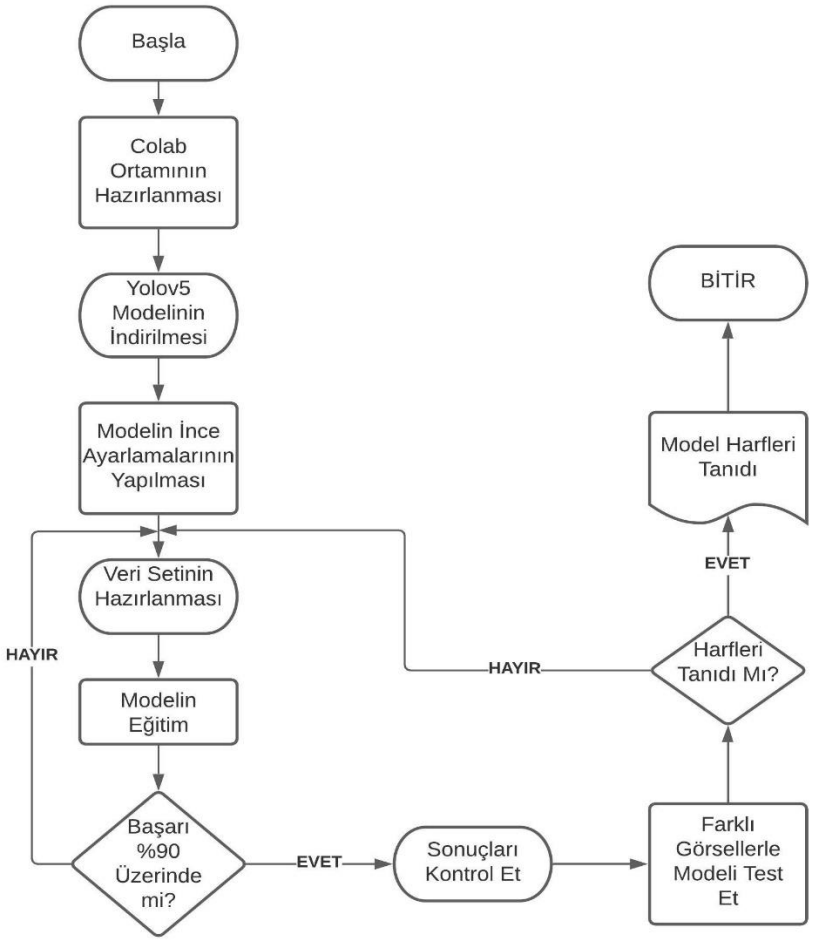
Türk İşaret Dili Alfabesinde bulunan harflerin tanınmasına yönelik olarak geliştirilecek uygulamada kullanılan modelin hem hızlı hem de doğru sonuçlar açısından tatmin edici olması gerekmektedir. Geliştirilen modellerin hepsinin yaklaşık olarak 2 saat gibi bir süre ve giriş boyutlarınında birbirine yakın olduğu göz önüne alındığında topla kayıp değeri 0.04 şeklinde olan YOLOv5s modelinin ön plana çıktığı görülmektedir. Bunun üzerine Yolov5s modeli ile yapılan denemelerde Tesla K4 çalıştıran bir Colab dosyası bir görüntüyü yaklaşık olarak 0.009 saniyede çıkarım yapmıştır. Yani saniyede yaklaşık olarak 110 FPS işlem yapılmaktadır. Diğer modellerle kıyaslama yapıldığında hem kayıp değerinin en düşük olması hemde diğer modellerden hızlı bir şekilde işlem yaptığından Türk İşaret Dili Alfabesinde bulunan harflerin tanınması amacıyla hazırlanacak olan uygulamada YOLOv5s modelinin kullanılması uygun bulunmuştur. Hazırlanan tüm modellere ait bilgiler Tablo 5.3.'de gösterilmektedir.

Tablo 5.3 Hazırlanan Modellere Ait Veriler

Model Adı	**Giriş Verisinin Boyutu**	**Adım / Epoch**	**Batch Size**	**Cls Loss**	**Lclzsn / Bbox Loss**	**Total Loss**	**Eğitim Süresi**
SSD Mobilenet	640x640	5000 Adım	14	0,10	0,04	0,84	2S 5 dk
Efficientdet D0	512x512	5000 Adım	16	0,47	0,25	0,78	2 S
YOLOv5s	416x416	100 Epoch	16	0,008	0,02	0,04	1 S 59 dk 47 sn

5.4 Uygulamanın Gerçekleştirilmesi

Gerçekleştirilen uygulama ile girdi olarak uygulamaya verilen resimlerden Türk İşaret Dili alfabesinde yer alan 29 adet harfin tanınması hedeflenmiştir. Bu amaçla girdi olarak verilen resim veya video içerisinde Türk İşaret Dili Harflerini oluşturan hareketlerin tanınması ve bu harflerin lokasyon bilgilerinin tespit edilerek kutu(Bounding Box) içerisine alınması sağlanmıştır. Bounding Box kutuları tespiti yapılan harfin hangi harf olduğunu üst köşelerinde belirtmektedir. Tespiti yapılan her harf için farklı renklerde Bounding Box kutucukları kullanılmıştır. Gerçekleştirilen uygulamanın akış şeması Şekil 5.12 de gösterilmiştir.



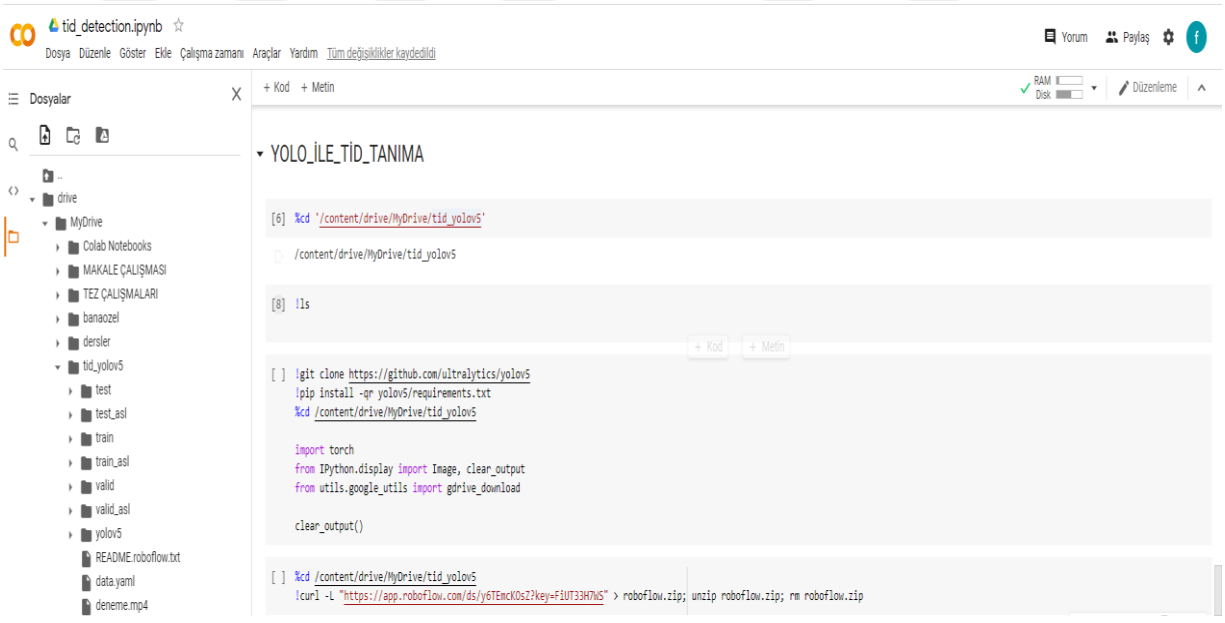
Şekil 5.12 Uygulamanın Akış Şeması

5.4.1 Google Colab Ortamının Hazırlanması

Geliştirilen bu uygulama için ilk etapta veri seti hazırlama işlemi yerel bilgisayarda gerçekleştirilmiş ancak Yolov5 modeli ile ilgili işlemlerin yüksek iş gücü gerektiren işlemlerden olması nedeniyle daha hızlı donanımlara sahip Google-Colab ortamının kullanılması uygun bulunmuştur. Google hesabı kullanılarak uygulamanın geliştirileceği yeni bir klasör oluşturulmuş, oluşturulan klasör içeriğine veri seti dosyaları aktarılmıştır. Sonrasında yukarıdaki bölümde anlatıldığı şekilde Google-Colab ortamı hazırlanmıştır. Hazırlanan ortamda uygulamalar GPU üzerinde gerçekleştirilecek ve Google sunucularının kullanılacağı uygulama geliştirmede aşağıdaki Şekil 5.13 de belirtilen özellikteki donanımlar kullanmıştır. Şekil 5.14 hazırlanan Colab ortamını göstermektedir.

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 2427612240146241028, name: "/device:GPU:0"
 device_type: "GPU"
 memory_limit: 14509932544
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 4677362159916227259
 physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5"]
```

Şekil 5.13 Uygulama için Colabda kullanılan donanımlar



Şekil 5.14 Hazırlanan Colab Ortamı

5.4.2 Modelin Oluşturulması

Google Colab ortamı hazırlandıktan sonra kullanılacak olan YOLOv5 modeli " https://github.com/ultralytics/yolov5 " bağlantısı kullanılarak işlem yapılacak Google Drive klasörüne indirilmiştir. Veri setinin hazırlanması bölümünde ayrıntılı olarak anlatılarak hazırlanan veri seti dosyaları da indirilen YOLOv5 modeli ile aynı Google Drive klasörü içerisinde yer alacak şekilde organize edilmiştir. Türk İşaret Dilinde bulunan harflerin tanınmasına yönelik olarak YOLOv5 model mimarisi eğitilecektir. Daha önceden COCO veri kümesi ile eğitilen bu modelde sınıf sayısı 80'dir. Bağlantı kutusu (anchors) otomatik öğrenmesi entegre edildiğinden, bağlantı kutusu parametreleri varsayılan olarak yoksayılmıştır. Tanımlanacak her harfin sınıfı ve sınıf sayısı, eğitim ve doğrulama resimlerinin yollarının da kaydedildiği "data.yaml" şeklindeki dosya ile Yolov5s modeli hazırlanır. Modeli oluşturan katman bilgileri aşağıdaki Şekil 5.15 de belirtilmştir.

```
# parameters
nc: 29  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, BottleneckCSP, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, BottleneckCSP, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, BottleneckCSP, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, BottleneckCSP, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

Şekil 5.15 Model Katman Bilgileri

5.4.3 Modelin Eğitimi

Modelin eğitimi çalıştırılacak olan "train.py" python kodu ile gerçekleştiirlecektir. Çalıştırılacak bu python kodunda bazı argümanların düzenlenmesi gerekmektedir. Değiştirilen argümanlarla ilgili görsel Şekil 5.16 de görselleştirilmiştir.

```
# train yolov5s on custom data for 200 epochs
# time its performance
%%time
%cd /content/drive/MyDrive/tid_yolov5/yolov5
!python train.py --img 416 --batch 16 --epochs 200 --data '../data.yaml' --cfg ./models/yolov5s.yaml --weights '' --name yolov5s_results  --cache
```

Şekil 5.16 Değiştirilen Argümanlar

Bu argümanlar arasında ilk sırada 'img' argümanı yer alır bu argüman ile eğitime dahil edilecek görüntülerin boyutu belirlenir. Eğitime katılacak verilerin boyutu 416x416 olacak şekilde düzenlenmiştir. Veriler "batch" parametresiyle partiler halinde eğitime dahil edilir. Veri kümesi batch parametresine verilen değer ile görüntü yığınları haline getirilir ve bu şekilde toplu eğitime bölünür. Her grubun sonuçları reme kaydedilir ve tüm grupların eğitiminin tamamlanmasının ardından bu değerler toplanır. Her grup için öğrenilen ağırlıklar reme kayıt edildiğinden bu grupların büyüklüğünün fazla olması rem kullanımını artırmaktadır. Bu model için Batch parametresi 16 olarak belirlenmiştir.

Modelin eğitimini etkiliyen faktörlerden biri de döngü sayısı veya dönem olarak adlandırılan epoch sayısıdır. Dönem (epoch) sayısı, modelin tüm girdileri eğitme ve temel doğruluk etiketlerine yaklaşmak için ağırlıkları güncelleme sayısını temsil eder. Eğitilen bu modelde dönem (Epoch) sayısı 200 olarak belirlenmiştir. Eğitim işleminde veri kümesi ile alakalı bilgileri içeren "data.yaml " dosyası kullanılmaktadır. Model değerlendirme süreci her epoch'tan hemen sonra yürütülür, bu nedenle model ayrıca data.yaml dosyasındaki yol aracılığıyla doğrulama dizinine erişecek ve o anda değerlendirme için içeriğini kullanacaktır.

Modelin eğitimi için değiştirilmesi gereken argümanlardan bir tanesi de "cfg" argümanıdır. Bu argüman modeli yapılandırma yolumuzu belirlememizi sağlar. Daha önce model yaml dosyasında tanımlanan mimariye dayanan bu komut satırı, train.py dosyasının bu mimariyi giriş görüntülerini eğitmek için derlemesine ve oluşturmasına olanak tanır. Model ağrlıkları "weights" parametresi ile ayarlanır. Bu parametre eğitim süresinden tasarruf etmek amacı ile önceden eğitilmiş bir ağırlığın bulunduğu yolu içerebilir. Kullandığımız bu modelde "weights" parametresi boş bırakılarak eğitimin otomatik olarak rastgele ağırlıklarla başlaması sağlanmıştır.

Değiştirilecek parametrelerden olana “name” parametresi ile eğitim sırasında gerçekleştirilen tüm sonuçları içeren bir dizin oluşturacaktır. Son olarak “cache”parametresi görüntüleri daha hızlı eğitim için önbelleğe almayı sağlar. Şekil 5.17 da modelin eğitim işleminin son aşaması görselleştirilmiştir.

```
      Epoch    gpu_mem       box       obj       cls     total    labels  img_size
    199/199      1.73G   0.02332  0.007587  0.005633   0.03654        13       416: 100% 438/438 [01:00<00:00,  7.23it/s]
                 Class      Images      Labels          P          R     mAP@.5  mAP@.5:.95: 100% 63/63 [00:11<00:00,  5.27it/s]
                   all        2000        2000      0.921      0.893      0.923      0.505
                     A        2000          70      0.977      0.843      0.937      0.454
                     B        2000          70      0.868          1      0.968      0.527
                     C        2000          70      0.882      0.957      0.987      0.649
                   CCC        2000          70      0.976      0.857      0.932      0.493
                     D        2000          70      0.836      0.726      0.794      0.518
                     E        2000          70      0.966        0.9       0.93      0.248
                     F        2000          70      0.946      0.843      0.914        0.6
                     G        2000          70      0.983      0.971      0.994      0.635
                   GGG        2000          70      0.976          1      0.995       0.55
                     H        2000          70      0.867      0.935       0.88      0.538
                     I        2000          60      0.925      0.983      0.969       0.48
                   III        2000          60      0.846      0.983      0.931      0.505
                     J        2000          60      0.731      0.867      0.703      0.351
                     K        2000          80      0.962      0.487      0.845       0.36
                     L        2000          60      0.999       0.95      0.993      0.574
                     M        2000          70      0.843      0.857       0.76      0.432
                     N        2000          70       0.99      0.857      0.935      0.625
                     O        2000          70      0.984      0.882      0.991      0.569
                   OOO        2000          70      0.994          1      0.995      0.536
                     P        2000          70       0.92          1      0.988      0.501
                     R        2000          70          1      0.948      0.985      0.329
                     S        2000          70      0.985      0.943      0.994      0.513
                   SSS        2000          70      0.877          1      0.994      0.571
                     T        2000          70          1      0.995      0.995      0.454
                     U        2000          70      0.752      0.986      0.909      0.578
                   UUU        2000          70          1      0.547      0.765      0.417
                     V        2000          70      0.813      0.957       0.97      0.638
                     Y        2000          70      0.955      0.914      0.912      0.527
                     Z        2000          70      0.861      0.714      0.814      0.461
200 epochs completed in 3.833 hours.

Optimizer stripped from runs/train/yolov5s_results15/weights/last.pt, 14.9MB
Optimizer stripped from runs/train/yolov5s_results15/weights/best.pt, 14.9MB
CPU times: user 2min 57s, sys: 22.1 s, total: 3min 19s
Wall time: 4h 43s
```

Şekil 5.17 Model Eğitiminin Tamamlanması

Gerçekleştirilen her Epoch, eğitim sürecini 438 grup için ortalama süre 1 dakika ve 63 grup üzerinde değerlendirme için 11 saniye işlem yapılmıştır. Eğitimin toplam süresi 4 saat 43 saniye olarak hesaplanmıştır. 200 Epoch sonunda 0.923 mAP değeri tespit edilmiştir. Tanımlanan her sınıf için hesaplanan doğruluk ve güven değerleri Şekil 5.6 da gösterilmiştir.

5.5 Model Değerlendirilmesinde Kullanilan Yöntemler

Her modelin eğitim süreci ve sonunda elde ettiği başarı ya da performansı ölçütlere uygun olarak değerlendirilmesi gerekmektedir. Nesne tespiti ve sınıflandırması için geliştirilen modellerde sınıf tespiti için dört olasılık mevcuttur. Bu olasılıklar True Positive, True Negative, False Positive ve False Negative olarak adlandırılmaktadır. Gerçek değeri doğru olan bir nesneyi doğru olarak sınıflandırmak True Positive, gerçek değeri yalnış olan bir nesneyi yalnış olarak sınıflandırmak True Negative, gerçek değeri doğru olan bir nesneyi yalnış olarak sınıflandırmak False Positive ve gerçek değeri yalnış olan bir nesneyi doğru olarak sınıflandırmak False Negative olarak tanımlamak mümkündür. Parametrelerinin anlaşılabilmesi için tahmini yapılan herhangi bir harfle ilgili bilgilerle açıklanmıştır. Örneğin gerçek değeri "M" harfi olarak bilinen bir görüntünün model tarafından yapılan tahmini sonucunda "M" harfi olarak tanınması True Positive, "M" harfinin dışında yanlış olarak tanınan herhangi bir harf değeri False Positive olarak değerlendirilmektedir. "M" harfi dışında başka bir hareketi içeren yalnış bir giriş görüntüsünün uygulamaya verilmesi sonucunda uygulamanın "M" harfini tespit edememesi yani yanlışa yanlış demesi True Negative, verilen aynı yanlış görüntünün "M" olarak tanınması ise False Negative olarak değerlendirilmektedir. Olasılıkarı içeren matris Karmaşıklık matrisi olarak tanımlanır ve Karmaşıklık matrisindeki değerler aşağıdaki Tablo 5.4 de belirtilmiştir.

Tablo 5.4 Karmaşıklık Matrisi

Karmaşıklık Matrisi	Tahmin Edilen değer	
Gerçek Değer	**True Positive** Gerçekte Doğru olup Doğru Tahmin edilen	**False Negative** Gerçekte Doğru olup Yalnış Tahmin edilen
	False Positive Gerçekte Yalnış olup Doğru Tahmin edilen	**True Negative** Gerçekte Yalnış olup Yalnış Tahmin edilen

Bu değerler kullanılarak Precision(Kesinlik), Recall(Duyarlılık) ve Mean Average Precision (Ortalama Kesinlik) değerleri hesaplanmaktadır.

5.5.1 Precision (Kesinlik)

Nesne tespiti ve sınıflandırması problemlerinde girdi resmi içeriğindeki tahmin edilen True Positive sonuçların, tahmin edilen tüm Positive değerlerin toplamına bölünmesi ile elde edilen değer Precision(Kesinlik) değeri olarak hesaplanır. Precision(Kesinlik) değerinin hesaplanmasında kullanılan formül Denklem 5.1 de verilmiştir.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{5.1}$$

5.5.2 Recall (Duyarlılık)

Nesne tespiti ve sınıflandırması problemlerinde girdi resmi içeriğindeki tahmin edilen True Positive sonuçların, olması gereken tüm Positive değerlerin toplamına bölünmesi ile elde edilen değer Recall (Duyarlılık) değeri olarak hesaplanır. Recall (Duyarlılık) değerinin hesaplanmasında kullanılan formül Denklem 5.2 de verilmiştir.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{5.2}$$

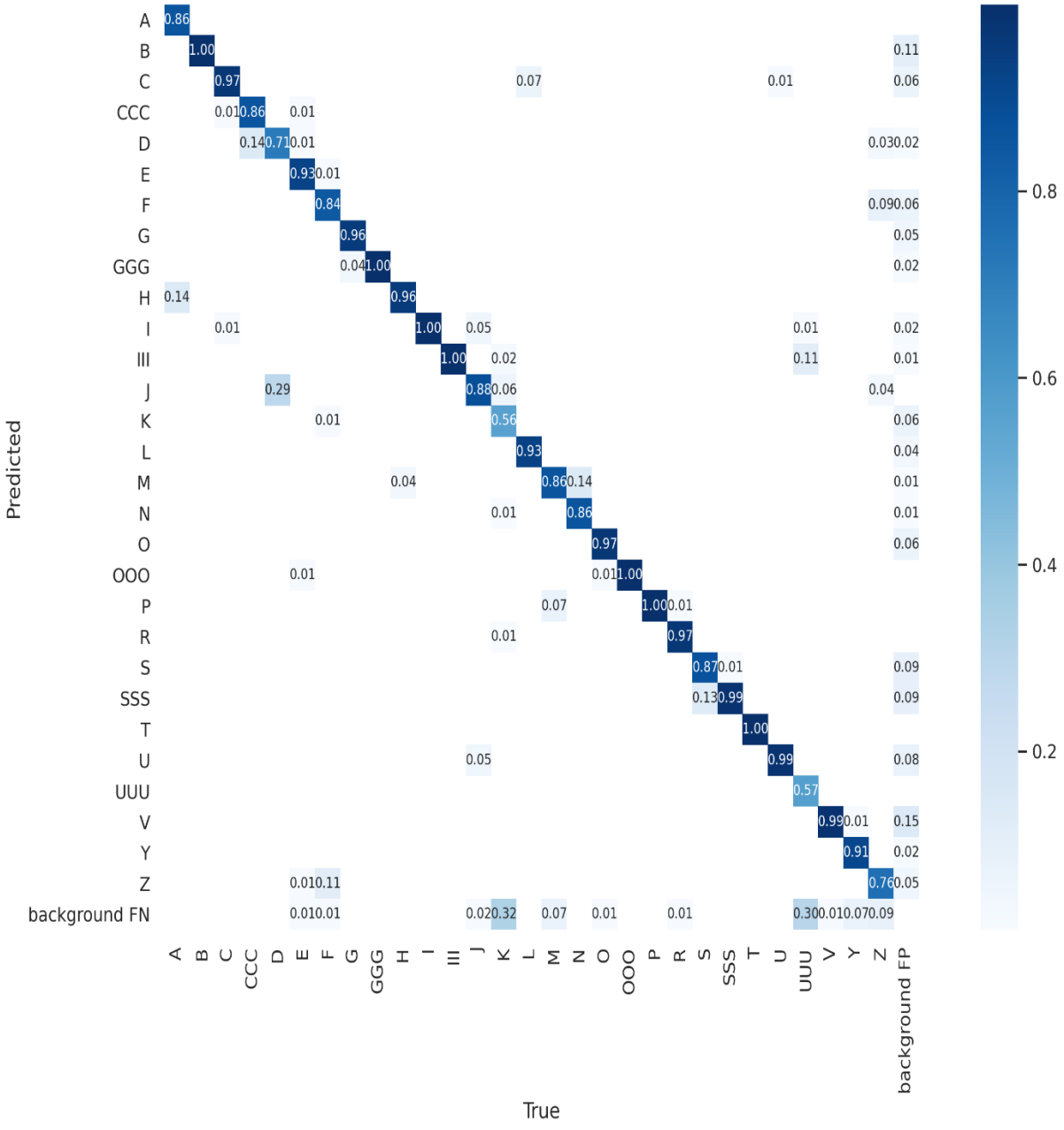
5.5.3 mAP (Ortalama Averaj Kesinliği)

Kesinlik ve Duyarlılık değerlerini birlikte kullanarak tek bir değer hesaplanmıştır. Ortalama kesinlik olarak tanımlanan bu değerin formülü aşağıdaki Denklem 5.3'te gösterilmektedir.

$$mAP = \int_0^1 P(R)dR \tag{5.3}$$

5.5.4 Modele Ait Veriler

Türk İşaret Dili alfabesindeki harflerin tanınması amacı ile Yolov5s modeli kullanılarak eğitimi tamamlanan modele ait oluşturulan karmaşıklık matrisi Şekil 5.18. de görselleştirilmiştir. Karmaşıklık matrisi Sol tarafta bulunan Predicted(Tahmin Edilen Değer) ve alt kısımda bulunan True(Gerçek Değer) kısımlarından meydana gelmektedir.



Şekil 5.18 Karmaşıklık Matrisi Değerleri

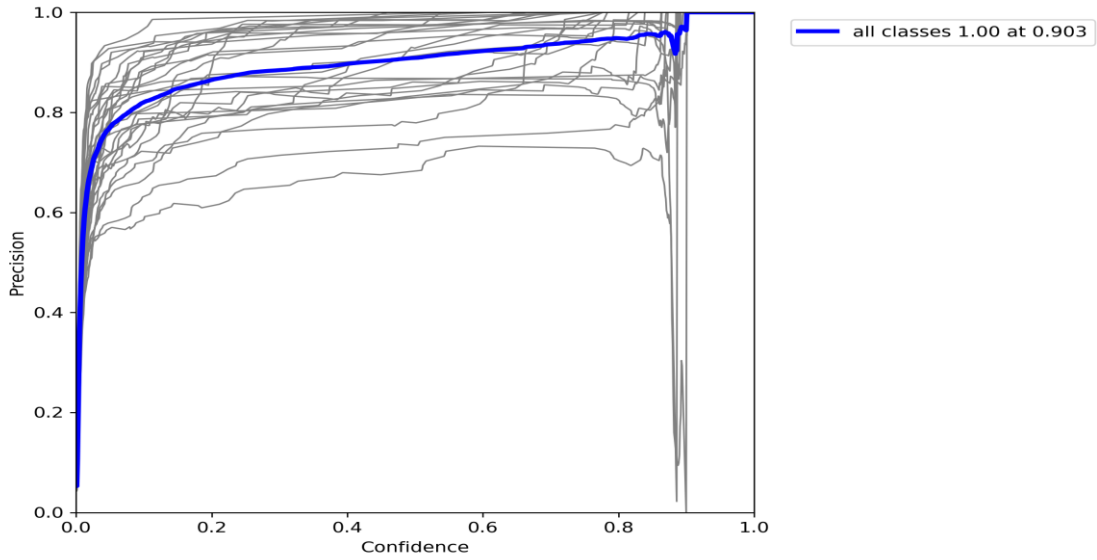
Eğitim işlemi 200 epoch olacak şekilde gerçekleştirilmiştir. Eğitim sonucunda hem toplam olarak hemde her sınıf için ayrı ayrı Precision(Kesinlik),Recall(Duyarlılık) ve Mean Average Precision (Ortalama Kesinlik) değerleri hesaplanmaktadır. Yapılan eğitim sonrasındaki **Precision Değeri 0.921, Recall Değeri 0.89, mAP@.5 Değeri 0.923 ve mAP@.5:.95: Değeri 0.505** toplam değerler olarak hesaplanmıştır.

Hesaplanan bu değerlere ilişkin bilgiler Şekil 5.19 da belirtilmiştir.

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95:
all	2000	2000	0.921	0.893	0.923	0.505
A	2000	70	0.977	0.843	0.937	0.454
B	2000	70	0.868	1	0.968	0.527
C	2000	70	0.882	0.957	0.987	0.649
CCC	2000	70	0.976	0.857	0.932	0.493
D	2000	70	0.836	0.726	0.794	0.518
E	2000	70	0.966	0.9	0.93	0.248
F	2000	70	0.946	0.843	0.914	0.6
G	2000	70	0.983	0.971	0.994	0.635
GGG	2000	70	0.976	1	0.995	0.55
H	2000	70	0.867	0.935	0.88	0.538
I	2000	60	0.925	0.983	0.969	0.48
III	2000	60	0.846	0.983	0.931	0.505
J	2000	60	0.731	0.867	0.703	0.351
K	2000	80	0.962	0.487	0.845	0.36
L	2000	60	0.999	0.95	0.993	0.574
M	2000	70	0.843	0.857	0.76	0.432
N	2000	70	0.99	0.857	0.935	0.625
O	2000	70	0.984	0.882	0.991	0.569
OOO	2000	70	0.994	1	0.995	0.536
P	2000	70	0.92	1	0.988	0.501
R	2000	70	1	0.948	0.985	0.329
S	2000	70	0.985	0.943	0.994	0.513
SSS	2000	70	0.877	1	0.994	0.571
T	2000	70	1	0.995	0.995	0.454
U	2000	70	0.752	0.986	0.909	0.578
UUU	2000	70	1	0.547	0.765	0.417
V	2000	70	0.813	0.957	0.97	0.638
Y	2000	70	0.955	0.914	0.912	0.527
Z	2000	70	0.861	0.714	0.814	0.461

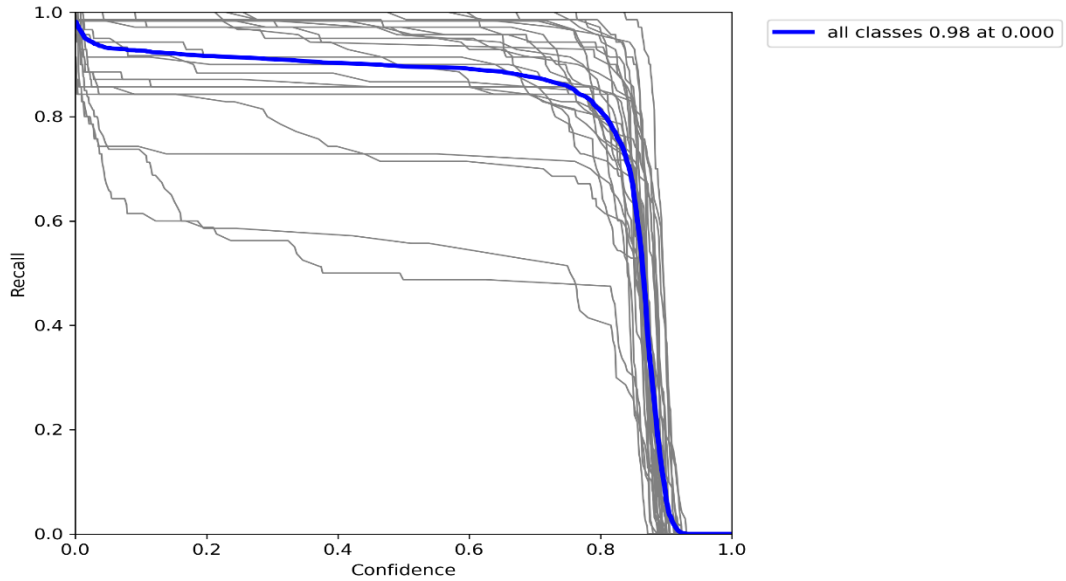
Şekil 5.19 Hesaplanan Değerler

Kullanılan YOLOv5s modelinin Precision (Kesinlik) grafiği Şekil 5.20 de gösterilmiştir.



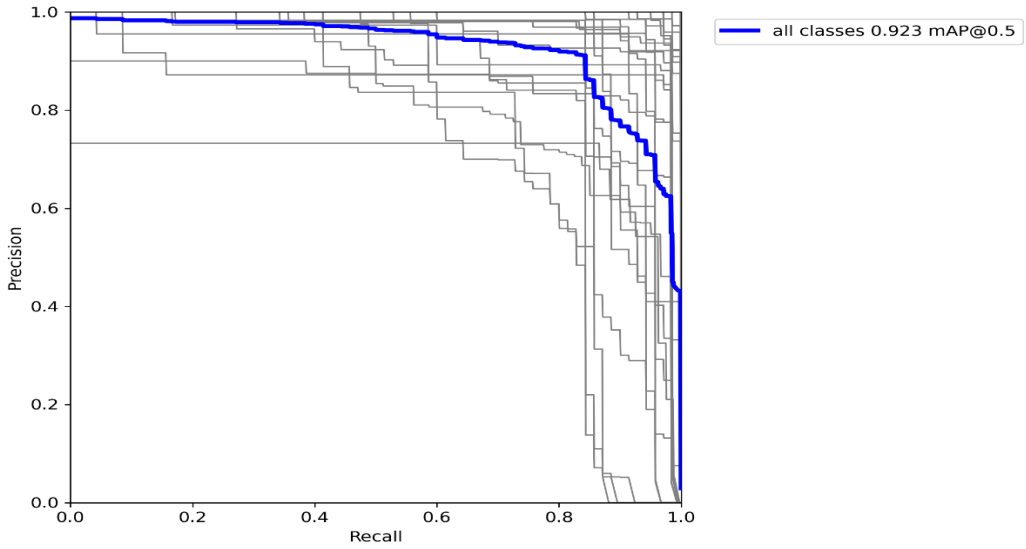
Şekil 5.20 Precision (Kesinlik) Grafiği

Kullanılan YOLOv5s modelinin Recall (Duyarlılık) grafiği Şekil 5.21 de gösterilmiştir.



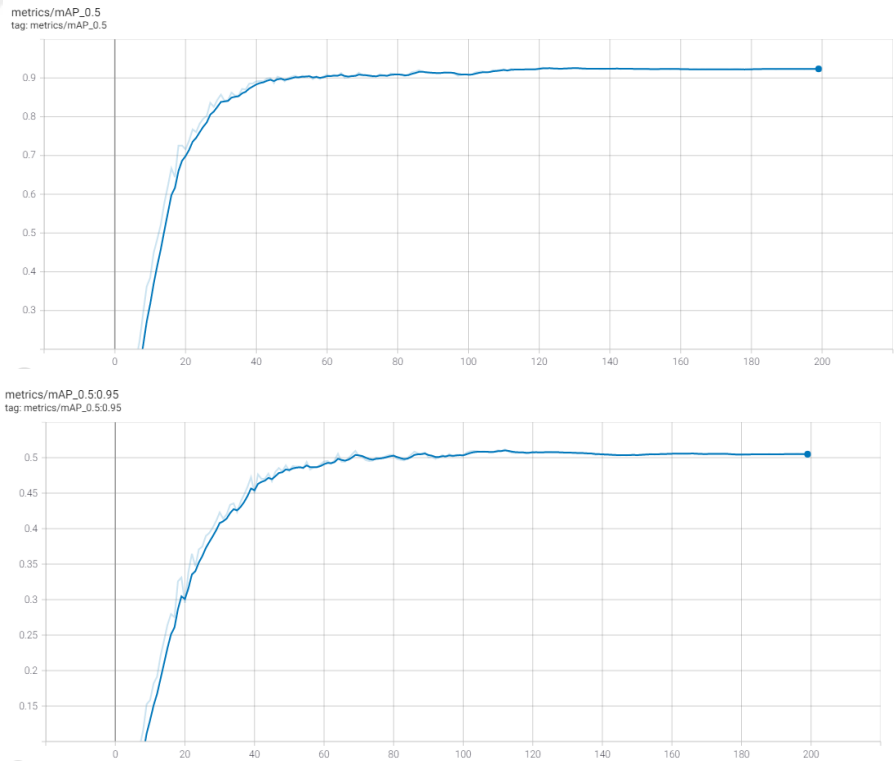
Şekil 5.21 Recall (Duyarlılık) Grafiği

Kullanılan YOLOv5s modelinin Precision (Kesinlik)-Recall (Duyarlılık) grafiği Şekil 5.22 de gösterilmiştir.



Şekil 5.22 Precision (Kesinlik)-Recall (Duyarlılık) Grafiği

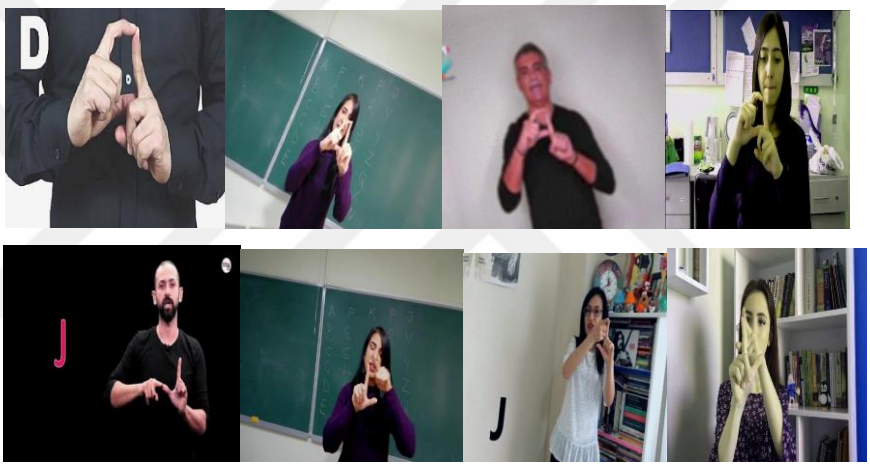
Kullanılan YOLOv5s modelinin mAP (Ortalama Averaj Kesinliği) değerlerinden mAP 0.5 ve mAP 0.5:0.95 grafiği Şekil 5.23 de gösterilmiştir.



Şekil 5.23 mAP 0.5 ve mAP 0.5:0.95 grafiği

Yapılan eğitim sonrasında tek tek her harf için elde edilen değerler kontrol edildiğinde mAP@.5 değeri 0.9 altında olan harfler olduğu tespit edilmiştir. Tespit edilen bu harflerin D,H,J,K,M,Ü ve Z harfleri olduğu görülmektedir. Türk işarret dilindeki bu harflerin sistem tarafından en çok hangi harflerle karıştırıldığına dair bilgiler hata matrisi kontrol edilerek elde edilebilmektedir. Hata matrisi kontrol edilerek tahmini karıştırılan harflerle ilgili bilgilere yer verilecektir.

"D" harfinin modeldeki mAP@.5 değeri 0.794 olarak hesaplanmış ve "J" harfinin modeldeki mAP@.5 değeri 0.703 olarak hesaplanmıştır. Bu değerlerin hesaplanmasındaki en büyük etmen, "J" olarak tahmin edilen harflerin gerçekte "D" harfi olmasıdır. Her iki harfi oluşturan hareketin iki el kullanılark yapılıyor olması ve parmakların pozisyonları harflerin karıştırılmasına sebep olduğu düşünülmektedir. Model tarafından tanınmasında zorluk çekilen ve karıştırılan "D" ve "J" harflerine ait veri setinde bulunan görsellerden bir kısmı aşağıdaki Şekil 5.24 de gösterilmiştir.



Şekil 5.24 "D" ve "J" Harflerine Ait Görseller

“H” harfinin modeldeki mAP@.5 değeri 0.88 olarak hesaplanmıştır. Bu değerin hesaplanmasındaki en büyük etmen, “H” olarak tahmin edilen harflerin gerçekte “A” harfi olmasıdır. Her iki harfi oluşturan hareketin iki el kullanılark yapılıyor olması ve parmakların pozisyonları harflerin karıştırılmasına sebep olduğu düşünülmektedir. Model tarafından tanınmasında zorluk çekilen ve karıştırılan “H” ve “A” harflerine ait veri setinde bulunan görsellerden bir kısmı aşağıdaki Şekil 5.25 de gösterilmiştir.

Şekil 5.25 “H” ve “A” Harflerine Ait Görseller

“K” harfinin modeldeki mAP@.5 değeri 0.845, “Ü” harfinin modeldeki mAP@.5 değeri 0.765 ve “Z” harfinin modeldeki mAP@.5 değeri 0.814 olarak hesaplanmıştır. Hesaplanan bu değerler kontrol edildiğinde her üç harfinde gerçek değerleri yerine model tarafından Backgraoun FN olarak tahmin edilmesi sonucunda hesaplandığı anlaşılmaktadır. Tahmin değerleri düşük çıkan bu harflerin karmaşık el işaretlerinden oluşması nedeniyle model tarafından yeterli derecede tanımlanamadığı ve arka plan olarak algılandığı düşünülmektedir. Model tarafından tanınmasında zorluk çekilen ve arkaplan olarak tanımlanan “K” harfine ait görseller Şekil 5.26 de, “Ü” harfine ait görseller Şekil 5.27 de ve “Z” harfine ait görsellerden bir kısmı aşağıdaki Şekil 5.28 de gösterilmiştir.

Şekil 5.26 “K” Harflerine Ait Görseller

Şekil 5.27. “Ü” harfine ait görseller

Şekil 5.278 “Z” Harflerine Ait Görseller

“M” harfinin modeldeki mAP@.5 değeri 0.76 olarak hesaplanmıştır. Bu değerin hesaplanmasındaki en büyük etmen, “M” olarak tahmin edilen harflerin gerçekte “N” harfi olmasıdır. Her iki harfi oluşturan hareketin iki el kullanılark yapılıyor olması ve her iki harfi oluşturan haraketlerin birbirine benzer olmasının sebep olduğu düşünülmektedir. Model tarafından tanınmasında zorluk çekilen ve karıştırılan “M” ve “N” harflerine ait veri setinde bulunan görsellerden bir kısmı aşağıdaki Şekil 5.29 de gösterilmiştir.

Şekil 5.289 “M” ve “N” Harflerine Ait Görseller

6. SONUÇLAR

Gerçekleştirilen yüksek lisans çalışması kapsamında Türk İşaret Dili alfabesinde bulunan harflerin hızlı ve doğru bir şekilde tanınması hedeflenmiştir. Bu hedef doğrultusunda gerçekleştirilecek uygulamaya uygun herkese açık olarak paylaşılan bir veri setinin olmadığı anlaşılmıştır. Bunun üzerine Türk İşaret Dilindeki harflerin öğrenilmesi amacıyla onbeş ayrı kişi tarafından herkese açık olarak paylaşılan videolar veri seti oluşturmak amacı ile kullanılmmıştır. Videolardan elde edilen görüntülerin yeterli olmadığı durumlarda yeni videolar hazırlanmıştır. Hazırlanan tüm videolar kullanılarak alfabemizde bulunan 29 harfi içeren toplamda 10000 adet görüntü ile yeni ve özgün bir veri seti oluşturulmuştur. Yaplılan litaratür taraması sonucunda nesne tanıma ve nesne takibi alanlarında 140 FPS hıza erişebilecek kapasitedeki YOLO modelinin kullanılması uygun bulunmuştur. Oluşturulan veri seti YOLO modelinin en son sürümü olan YOLOv5 sürümü ile Google-Colab ortamında eğitime tabi tutularak % 92,3 başarım sağlanmıştır.

Gerçekleştirilen uygulama ile her ne kadar %92,3 başarı elde edilmişse de bu başarı oranı etkileyen en büyük etmenlerden biri oluşturulan veri setidir. Uygulamada kullanılan veri seti birçok farklı yaş grubundaki farklı cinsiyetlerdeki insanlar tarafından oluşturulsa da Türk İşaret Dili alfabesini oluşturan hareketlerin büyük bir bölümünün her iki el kullanılarak yapılan hareketlerden olması ve bazı harfleri oluşturan hareketlerin sabit olmaması nedeniyle veri setinde yapılacak olan iyileştirilmelerin modelin başarımını artırılabileceği düşünülmektedir. Harflerle ilgli yapılan çalışmaların başarı yüzdesinin artması ileriye dönük olarak Türk İşaret Dilini oluşturan kelimelerin hatta cümlelerin tanınmasına yönelik olarak yapılacak çalışmalara motivasyon oluşturacaktır. Ayrıca Türk işaret dilinin tanınmasına yönelik olarak yapılacak her türlü çalışmanın bilgisayar bilimlerine sağlayacağı katkıların yanında toplumumuzdaki engelli bireylerin günlük yaşantılarına sağlayacağı kolaylıklar açısından sağlayacağı faydalar bakımından değerli olduğu düşünülmektedir.

KAYNAKLAR

[1] Charayaphan, C., & Marble, A. E. (1992). Image processing system for interpreting motion in American Sign Language. Journal of Biomedical Engineering, 14(5), 419-425.

[2] Takahashi, T., & Kishino, F. (1991). Hand gesture coding based on experiments using a hand gesture interface device. Acm Sigchi Bulletin, 23(2), 67-74.

[3] Kramer, J., & Leifer, L. J. (1990). A 'talking glove'for nonverbal deaf individuals. SIMA Tech Report.

[4] Kramer, J. F. (1996). The TalkingGlove®: hand-gesture-to-speech using an instrumented glove and a tree-structured neural classifying vector quantizer (Doctoral dissertation, Stanford University).

[5] Murakami, K., & Taguchi, H. (1991, March). Gesture recognition using recurrent neural networks. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 237-242).

[6] Waldron, M. B., & Kim, S. (1995). Isolated ASL sign recognition system for deaf persons. IEEE Transactions on rehabilitation engineering, 3(3), 261-271.

[7] Wysoski, S. G., Lamar, M. V., Kuroyanagi, S., & Iwata, A. (2002, November). A rotation invariant approach on static-gesture recognition using boundary histograms and neural networks. In Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02. (Vol. 4, pp. 2137-2141). IEEE.

[8] Allen, J. M., Asselin, P. K., & Foulds, R. (2003, March). American Sign Language finger spelling recognition system. In 2003 IEEE 29th Annual Proceedings of Bioengineering Conference (pp. 285-286). IEEE.

[9] Wang, H. G., Sarawate, N. N., & Leu, M. C. (2004, July). Recognition of American sign language gestures with a sensory glove. In Japan USA Symposium on Flexible Automation, Denver, CO (pp. 102-109).

[10] Oz, C., & Leu, M. C. (2011). American Sign Language word recognition with a sensory glove using artificial neural networks. Engineering Applications of Artificial Intelligence, 24(7), 1204-1213.

[11] Bowden, R., & Sarhadi, M. (2002). A non-linear model of shape and motion for tracking finger spelt american sign language. Image and Vision Computing, 20(9-10), 597-607.

[12] Kane, L., & Khanna, P. (2015). A framework for live and cross platform fingerspelling recognition using modified shape matrix variants on depth silhouettes. Computer Vision and Image Understanding, 141, 138-151.

[13] Kumar, E. K., Kishore, P. V. V., Sastry, A. S. C. S., Kumar, M. T. K., & Kumar, D.

[14] A. (2018). Training CNNs for 3-D sign language recognition with color texture coded joint angular displacement maps. IEEE Signal Processing Letters, 25(5), 645-649.

[15] Kim, D., Rodriguez, S. A., Matson, E. T., & Kim, G. J. (2018). Special issue on smart interactions in cyber-physical systems: Humans, agents, robots, machines, and sensors.

[16] Vassilia, P. N., & Konstantinos, M. G. (2006, May). Multimodal continuous recognition system for greek sign language using various grammars. In Hellenic Conference on Artificial Intelligence (pp. 584-587). Springer, Berlin, Heidelberg.

[17] Malik, M. S. A., Kousar, N., Abdullah, T., Ahmed, M., Rasheed, F., & Awais, M. (2018). Pakistan Sign Language Detection using PCA and KNN. International Journal of Advanced Computer Science and Applications, 9(54), 78-81.

[18] Vo, D. H., Huynh, H. H., Doan, P. M., & Meunier, J. (2017). Dynamic gesture classification for Vietnamese sign language recognition. Int. J. Adv. Comput. Sci. Appl, 8(3), 412

[19] Haberdar, H., & Albayrak, S. (2005, October). Real time isolated turkish sign language recognition from video using hidden markov models with global features. In International Symposium on Computer and Information Sciences (pp. 677-687). Springer, Berlin, Heidelberg.

[20] Haberdar, H., & Albayrak, S. (2006, September). A two-stage visual turkish sign language recognition system based on global and local features. In International Symposium on Methodologies for Intelligent Systems (pp. 29-37). Springer, Berlin, Heidelberg.

[21] Memiş, A., & Albayrak, S. (2013, April). Turkish Sign Language recognition using spatio-temporal features on Kinect RGB video sequences and depth maps. In 2013 21st Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

[22] Ceber, Y. E., Karacaoğlan, E., Uysaf, F., & Tokmakçi, M. (2017, October). The design of glove that can translate sign language to Turkish language. In 2017 Medical Technologies National Congress (TIPTEKNO) (pp. 1-4). IEEE.

[23] Çelik, Ö , Odabas, A . (2020). Sign2Text: Konvolüsyonel Sinir Ağları Kullanarak Türk İşaret Dili Tanıma . Avrupa Bilim ve Teknoloji Dergisi , (19) , 923-934 .

[24] Türk İşaret Dili Sözlüğü, https://orgm.meb.gov.tr [Erişildi: 01/06/2020]

[25] ZESHAN Ulrike, Aspects of Türk İşaret Dili (Turkısh Sign Language), Sign Language & Linguistics, vol.6, no.1, s.43-75., 2003.

[26] MILES M., Signing in the Seraglio: Mutes, dwarfs and gestures at the Ottoman Court 1500-1700, Disability & Society, vol. 15, no. 1, 115-134, 2000.

[27] TÜİK, Türkiye Engelliler Araştırması, 2002, http://www.tuik.gov.tr [Erişildi: 01/06/2020]

[28] Başkent Üniversitesi Fen Bilimleri Enstitüsü Türk İşaret Dili Alfabesinin Derin Öğrenme Yöntemi İle Sınıflandırılması Zeren Berna KIN 2019 Yüksek Lisans Tezi

[29] Yapay Zeka Ders Notları https://medium.com/@yasinguzel/yapay-zeka-ders-notlar%C4%B1-03-biyolojik-sinir-sistemi-ve-yapay-sinir-a%C4%9F%C4%B1-h%C3%BCcresi-6555add68d80 [Erişildi: 10/05/2021]

[30] HAYKIN, Simon, Neural Networks A Comprehensive Foundation, Second Edition, Pearson Education, 1999.

[31] ÖZTEMEL, Ercan, Yapay Sinir Ağları, 3. Basım, Papatya Yayıncılık, 2012.

[32] Dzone-articles Comparison Between Deep Learning and Machine Learning https://dzone.com/articles/comparison-between [Erişildi: 10/05/2021]

[33] Girshick, R., Donahue, J., Darrell, T., & Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

[34] Girshick, R. 2015. Fast r-cnn. arXiv preprint arXiv:1504.08083.

[35] http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf [Erişildi: 25/05/2021]

[36] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. 2016. Ssd: Single shot multibox detector. Paper presented at the European conference on computer vision.

[37] M. Tan ve A. Yu, «EfficientDet: Towards Scalable and Efficient Object Detection,» 15 4 2020. [Çevrimiçi]. Available: https://ai.googleblog.com/2020/04/efficientdet-towardsscalable-and.html. [Erişildi: 25/05/2021]

[38] J. Redmon, S. Divvala, R. Girshick ve A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection» 2016.

[39] J. Redmon ve A. Farhadi, «YOLO9000: Better, Faster, Stronger,» 2016.

[40] J. Redmon ve A. Farhadi, «YOLOv3: An Incremental Improvement,» Washington, 2018.

[41] R. Geng, Y. Ma ve W. Huang, «An improved helmet detection method for YOLOv3 on an unbalanced dataset,» Dalian, 2020

[42] J. Hui, «YOLOv4,» 4 5 2020. [Çevrimiçi]. Available: https://jonathanhui.medium.com/yolov4-c9901eaa8e61. [Erişildi: 22/05/2021]

[43] Q. Shi and J. Li, "Objects Detection of UAV for Anti-UAV Based on YOLOv4," 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology

ÖZGEÇMİŞ

Fatih BANKUR

[redacted]