

T.C. İSTANBUL KÜLTÜR ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

HAREKETLİ NESNE ANALİZİ İÇİN DERİN ÖĞRENME MİMARİSİ

YÜKSEK LİSANS TEZİ

Okay TÜRKYILMAZ

2000001488

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Bilgisayar Mühendisliği

Tez Danışmanı : Doç.Dr. Fatma PATLAR AKBULUT

HAZİRAN 2024

T.C. İSTANBUL KÜLTÜR ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

HAREKETLİ NESNE ANALİZİ İÇİN DERİN ÖĞRENME MİMARİSİ

YÜKSEK LİSANS TEZİ

Okay TÜRKYILMAZ

2000001488

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Bilgisayar Mühendisliği

Tez Danışmanı: Doç.Dr. Fatma PATLAR AKBULUT

Jüri Üyeleri: Doç.Dr. Akhan AKBULUT

Doç.Dr. Can EYÜPOĞLU

HAZİRAN 2024

İÇİNDEKİLER LİSTESİ

KISALTMALAR LİSTESİ	vii
ŞEKİLLER LİSTESİ	ix
TABLolar LİSTESİ	x
KISA ÖZET	xi
1. GİRİŞ	1
2. EVRİŞİMSEL SİNİR AĞLARI	5
2.1. Evrişimsel Sinir Ağının Basit Çalışma Prensibi.....	6
2.2. Evrişimsel Sinir Ağı Katmanları	6
2.2.1. Evrişimsel Katman	7
2.2.1.1. Evrişimde Kullanılan Filtreler	7
2.2.1.2. Evrişimsel Operasyon	8
2.2.1.3. Özellik Haritaları	10
2.2.1.4. Adımlama İşlemi.....	10
2.2.1.5. Dolgulama İşlemi.....	11
2.2.2. Aktivasyon Katmanı.....	11
2.2.3. Havuzlama Katmanı.....	12
2.2.4. Tamamen Bağlı Katman	13
2.3. Geri Yayılım Algoritması.....	17
2.4. Optimizasyon Algoritmaları	19
2.4.1. Toplu Gradyan İnişi	20
2.4.2. Stokastik Gradyan İnişi	20
2.4.3. Küçük Gruplu Gradyan İnişi	21
2.4.4. Momentum	21
2.4.5. RMSprop	22
2.4.6. Adam	22
2.4.7. Diğer Optimizasyon Algoritmaları	23
2.5. Aktivasyon Fonksiyonları	23
2.5.1. Doğrusal ve Adım Fonksiyonları	24
2.5.2. Sigmoid, Tanh, ArcTan, Softsign ve Softmax Fonksiyonları	25

2.5.3.	ReLU, Leakly ReLU, PReLU ve RLReLU	27
2.5.4.	Doğrusal Olmayan Diğer Aktivasyon Fonksiyonları.....	29
2.6.	Kayıp Hesaplama Fonksiyonları	31
2.6.1.	İkili Çapraz – Entropi Fonksiyonu	31
2.6.2.	Ağırlıklandırılmış İkili Çapraz-Entropi Fonksiyonu.....	32
2.6.3.	Kategorik Çapraz-Entropi Fonksiyonu	33
2.6.4.	Seyrek Kategorik Çapraz-Entropi Fonksiyonu	33
2.6.5.	Etiket Yumuşatma İle Çapraz-Entropi Fonksiyonu	34
2.6.6.	Odak Kayıp Fonksiyonu (Focal Loss)	35
2.6.7.	Hinge Kayıp Fonksiyonu	35
2.6.8.	Ortalama Yanlış Hata Kayıp Fonksiyonu	36
2.7.	Evrışim Sinir Ağlarında Eğitim Öncesi Süreç	36
2.7.1.	Veri Toplama, Etiketleme ve Verilerin Yeniden Boyutlandırılması ...	37
2.7.2.	Veri Setinin Bölünmesi	37
2.7.3.	Ağırlıkların Başlatılması	38
2.7.3.1.	Sıfır ve Bir ile Başlatma Tekniği	38
2.7.3.2.	Rastgele Başlatma Tekniği	39
2.7.3.3.	LeCun Başlatma Tekniği	39
2.7.3.4.	Xavier Başlatma Tekniği(Glorot Başlatma)	39
2.7.3.5.	He Başlatma Tekniği (Kaiming Başlatma).....	39
2.8.	CNN Performansının İyileştirilmesi İçin Geliştirilen Teknikler	40
2.8.1.	Unutturma Teknikleri.....	40
2.8.1.1.	Nöronları Unutturma(Dropout) Tekniği	40
2.8.1.2.	Maksimum Unutturma(MaxDropout) Tekniği	41
2.8.1.3.	Ağırlıkları Unutturma(DropConnect) Tekniği.....	41
2.8.1.4.	Blokları Unutturma(DropBlock) Tekniği	42
2.8.1.5.	Diğer Unutturma Teknikleri	42
2.8.2.	Normalizasyon Teknikleri.....	42
2.8.2.1.	Toplu Normalleştirme.....	43
2.8.2.2.	Ağırlıkların Normalize Edilmesi	44
2.8.2.3.	Diğer Normalizasyon Teknikleri	45
2.8.3.	Kayıp Fonksiyonunu Düzenleme Teknikleri	46
2.8.3.1.	L2 Düzenlemesi (Ridge).....	47

2.8.3.2.	L1 Düzenlemesi (Lasso)	47
2.8.3.3.	Elastik Ağ Düzenlemesi.....	48
2.8.4.	Etiket Yumuşatma Teknikleri	48
2.8.4.1.	Tek Tip Etiket Yumuşatma Tekniği	48
2.8.4.2.	Uyarlayıcı Etiket Yumuşatma Tekniği	49
2.8.5.	Veri Artırma Teknikleri	49
2.8.6.	Erken Durdurma.....	50
2.8.7.	Öğrenme Transferi	51
2.8.8.	Sınıf Dengesizliği Çözüm Teknikleri.....	52
2.8.9.	Sinir Ağı Eğitilirken Dikkat Edilmesi Gereken Konular	53
2.9.	CNN Değerlendirme Metrikleri	55
2.9.1.	Hata Matrisi.....	55
2.9.2.	Doğruluk (Accuracy)	56
2.9.3.	Kesinlik (Precision).....	56
2.9.4.	Duyarlılık (Recall)	56
2.9.5.	F1 Skoru	57
2.9.6.	F2 skoru.....	57
2.9.7.	Gerçek Negatif Oran(TNR).....	57
2.9.8.	Yanlış Pozitif Oran (FPR).....	57
2.9.9.	Negatif Tahmin Değeri(NPV).....	57
2.9.10.	Yanlış Keşif Oranı(FDR)	58
3.	NESNE TESPİTİ	58
3.1.	Geleneksel Yöntemlerle Nesne Tespitinin Yapılması.....	59
3.1.1.	Yönlü Gradyan Histogramı	59
3.1.2.	Haar Benzeri Özellikler.....	59
3.1.3.	Özellik Eşleştirme	59
3.1.4.	İkili Sağlam Bağımsız Temel Özellikler.....	59
3.1.5.	Seçici Arama Algoritması	60
3.2.	Derin Öğrenme Yöntemleri İle Nesne Tespitinin Yapılması.....	60
3.2.1.	İki Aşamalı Nesne Algılama Metotları	61
3.2.1.1.	Bölgesel Bazlı Evrimsel Sinir Ağı (R-CNN).....	61
3.2.1.2.	Uzamsal Piramit Havuzlama Ağı (SPPNet)	62
3.2.1.3.	Hızlı R-CNN	62

3.2.1.4.	Daha Hızlı R-CNN.....	63
3.2.1.5.	Bölge Tabanlı Tamamen Evrişimsel Ağ(R-FCN)	63
3.2.2.	Tek Aşamalı Nesne Algılama Metotları	64
3.2.2.1.	SSD	64
3.2.2.2.	YOLOv1	65
3.2.2.3.	YOLOv2 ve YOLOv3	67
3.2.2.4.	YOLOv4	68
3.2.2.5.	YOLOv5	68
3.2.2.6.	YOLOv7	69
3.2.2.7.	Diğer Nesne Algılama Dedektörleri	71
3.3.	Nesne Tespitinde Kullanılan Kayıp Fonksiyonları	71
3.3.1.	Düzgün L1 Kaybı(Huber Kaybı)	72
3.3.2.	L2 Kaybı.....	72
3.3.3.	L1 Kaybı.....	72
3.3.4.	Birleşim Üzerinde Kesişen Kayıp(IoU)	73
3.3.5.	Genelleştirilmiş IoU Kaybı(GIoU)	73
3.3.6.	Uzaklık-IoU Kaybı(DIoU).....	73
3.3.7.	Tam IoU Kayıp Fonksiyonu (CIoU).....	74
3.3.8.	Ölçeğe Duyarlı IoU Kayıp Fonksiyonu (SIoU)	75
3.3.9.	Odaksal Verimli IoU	76
3.3.10.	YOLO Modelinde Kullanılan Kayıp Fonksiyonları	77
3.4.	Nesne Algılama Değerlendirme Ölçümleri	77
3.4.1.	Birleşim Üzerinde Kesişme (IoU).....	78
3.4.2.	Ölçümde Kullanılan Parametreler.....	78
3.4.3.	Nesne Tespitinde Yaygın Olarak Kullanılan Eşik Değerleri	79
3.4.4.	Maksimum Olmayan Bastırma Tekniği(NMS).....	79
3.4.5.	Kesinlik (Precision).....	79
3.4.6.	Duyarlılık (Recall)	79
3.4.7.	F1 Skoru	80
3.4.8.	Ortalama Kesinlik(AP).....	80
3.4.9.	mAP, mAP 50 ve mAP 50-95 Ölçümleri.....	80
3.5.	Nesne Tanıma Çalışmalarının Amacı ve Önemi	81
3.5.1.	Otonom Sistemler	81

3.5.2.	Gözetleme ve Güvenlik.....	82
3.5.3.	Tıbbi Görüntü İşleme	82
3.5.4.	Nesne Tespitinin Endüstri Alanındaki Katkısı.....	82
3.5.5.	Nesne Tespitinin Çevresel İzlemeye Katkısı	83
3.6.	Nesne Algılama Algoritmalarında Karşılaşılan Zorluklar ve Çözümler.....	83
3.6.1.	Hesaplama Karmaşıklığı	83
3.6.2.	Hız ve Doğruluk Arasındaki Hassas Denge.....	83
3.6.3.	Farklı Ölçekli Nesneleri Algılama	84
3.6.4.	Farklı Açı ve Görünüşteki Nesnelerin Algılanmasındaki Zorluklar	84
3.6.5.	Birbirlerine Karışmış Olan Nesnelerin Algılanması	84
3.6.6.	Ortamlara Uyum Sağlamadaki Zorluklar	85
3.6.7.	Küçük Nesnelerin Tespitindeki Zorluklar.....	86
4.	NESNE TESPİTİ İLE İLGİLİ YAPILAN ÇALIŞMALAR.....	87
5.	GELİŞTİRİLEN MODEL	89
5.1.	Eğitim Öncesi Süreç	90
5.1.1.	Araçların Görüntü Üzerinde Tespit Edilmesi ve Sınıflandırılması.....	90
5.1.2.	Veri Toplama	91
5.1.3.	Model İçin Görüntü Etiketleme Metodu	92
5.1.4.	Veri Setini Eğitim, Doğrulama ve Test Seti Olarak Bölüm İşlemi.....	94
5.1.5.	Algoritma Seçimi	95
5.1.6.	Öğrenme Transferi	97
5.1.7.	İhtiyaç Duyulan Donanım ve Yazılımlar	98
5.1.8.	YOLOv7 Algoritmasında Kullanılan Hiper Parametreler	98
5.2.	Modelin Eğitilmesi	102
5.2.1.	Modelin Eğitiminde Kullanılan Parametreler	103
5.2.2.	Eğitilen Modellerin Performansının Analiz Edilmesi.....	106
5.2.3.	49 Sınıftan Oluşan Veri Setinin Eğitilmesi ve Değerlendirilmesi	107
5.2.4.	42 Sınıftan Oluşan Veri Setinin Eğitilmesi ve Değerlendirilmesi	109
5.2.5.	35 Sınıftan Oluşan Veri Setinin Eğitilmesi ve Değerlendirilmesi	111
5.2.5.1.	Kesinlik Eğrisinin Değerlendirilmesi	113
5.2.5.2.	Duyarlılık Eğrisinin Değerlendirilmesi	115
5.2.5.3.	Kesinlik – Duyarlılık Eğrisinin Değerlendirilmesi.....	116
5.2.5.4.	F1 Skor Eğrisinin Değerlendirilmesi	117

5.2.5.5.	Hata Matrisinin Değerlendirilmesi	118
5.3.	Modelin Test Verisi Üzerinde Test Edilmesi	120
5.3.1.	Modelin Değişik Güven Skorları İle Test Edilmesi.....	120
5.3.2.	Test Verilerinden Elde Edilen Sonuçların Değerlendirilmesi.....	124
6.	TARTIŞMA	124
7.	SONUÇ	127
8.	KAYNAKLAR	129
9.	EKLER	139
9.1.	EK A.1. Gradyanların Hesaplanması ve Ağırlıkların Güncellenmesi.....	139



KISALTMALAR LİSTESİ

Adaboost	: Ayarlanabilir Güçlendirme
ADAM	: Uyarlanabilir An Tahmini (Adaptive Moment Estimation)
AP	: Ortalama Kesinlik(Average Precision)
AUC	: Eğri Altında Kalan Alan (Area Under Curve)
BCE	: İkili Çapraz-Entropi (Binary Cross-Entropy)
BGD	: Toplu Gradyan İnişi(Batch Gradient Descent)
BRIEF	: İkili Sağlam Bağımsız Temel Özellikler(Binary Robust Independent Elementary Features)
CCE	: Kategorik Çapraz Entropi
CIoU	: Tam IoU(Complete IoU)
CNN	: Evrişimsel Sinir Ağları (Convolutional Neural Network)
DCNN	: Derin Evrişimsel Sinir Ağları
DIoU	: Uzaklık IoU(Distance IoU)
DNN	: Derin Sinir Ağı (Deep Neural Network)
DPM	: Deforme Olabilen Parça Tabanlı Model
dSiLU	: Türevlenebilir Sigmoid Doğrusal Birim Fonksiyonu
EIoU	: Verimli IoU
ELU	: Üstel Doğrusal Birim
EWC	: Elastik Ağırlık Konsolidasyonu(Elastic Wight Consolidation)
Fast R-CNN	: Hızlı Bölgesel Tabanlı Evrişimsel Sinir Ağı
Faster R-CNN	: Daha Hızlı Bölgesel Tabanlı Evrişimsel Sinir Ağı
FC	: Tamamen Bağlı Ağ (Fully Connected Network)
FDR	: Yanlış Keşif Oranı(False Discovery Rate)
FPR	: Yanlış Pozitif Oran(False Positive Rate)
FPS	: Saniyedeki Kare Sayısı (Frames Per Second)
GeLU	: Gauss Hata Doğrusal Birim Aktivasyon Fonksiyonu
GIoU	: Genelleştirilmiş IoU
HOG	: Yönlü Gradyanlar Histogramı (Histogram of Oriented Gradients)
IoU	: Birleşim Üzerinde Kesişim(Intersection Over Union)
ISRLU	: Ters Karekök Doğrusal Birim
LR	: Öğrenme Oranı(Learning Rate)
LReLU	: Sızdıran Düzeltilmiş Doğrusal Birim (Leakly Rectified Linear Unit)
MAE	: Ortalama Mutlak Hata(Mean Absolute Error)
MLP	: Çok Katmanlı Algılayıcı (Multi-Layer Perceptron)
MSE	: Ortalama Kare Hatası(Mean Square Error)
NMS	: Maksimal Olmayan Bastırma(Non-Maximum Suppression)
NPV	: Negatif Tahminleyici Değer(Negative Predictive Value)
PAA	: Olasılığa Dayalı Atama(Probabilistic Anchor Assignment)
PANet	: Yol Kümeleme Ağı(Path Aggregation Network)
PLU	: Parçalı Doğrusal Birim (Piecewise Linear Unit)
PReLU	: Parametrik Düzeltilmiş Doğrusal Birim
R-CNN	: Bölgesel Tabanlı Evrişimsel Sinir Ağı (Region Based Convolutional Neural Network)
ReLU	: Düzeltilmiş Doğrusal Birim (Rectified Linear Unit)
R-FCN	: Bölge Tabanlı Tamamen Evrişimsel Ağ(Region-Based Fully Convolutional Network)

RLReLU	: Rastgele Sızdıran Düzeltilmiş Doğrusal Birim
RMSProp	: Kök Ortalama Kare Yayılım (Root Mean Square Propagation)
RNN	: Tekrarlayan Sinir Ağları(Recurrent Neural Network)
RoI	: Bölge Önerileri(Region of Interest)
RPN	: Bölge Öneri Ağı (Region Proposal Network)
RvNN	: Özyinelemeli Sinir Ağları(Recursive Neural Network)
SeLU	: Ölçeklendirilebilir Doğrusal Birim
SIFT	: Ölçekle Değişmeyen Özellik Dönüşümü (Scale-Invariant Feature Transform)
SIoU	: Ölçeğe Duyarlı IoU
SiLU	: Sigmoid Doğrusal Birim Fonksiyonu
SMAE	: Düzgün Ortalama Mutlak Hata(Smooth Mean Absolute Error)
SPM	: Uzamsal Piramit Eşleştirme
SPP	: Uzamsal Piramit Havuzlama
SPPNet	: Uzamsal Piramit Havuzlama Ağı
SSD	: Tek Atış Dedektörü (Single Shot Dedector)
SURF	: Hızlandırılmış Sağlam Özellikler (Speeded-Up Robust Features)
SVM	: Destek Vektör Makinesi (Support Vector Machine)
Swish	: Kendiliğinden Geçmeli Aktivasyon Fonksiyonu
TDR	: Gerçek Keşif Oran(True Discovery Rate)
TL	: Öğrenme Transferi(Transfer Learning)
TNR	: Gerçek Negatif Oran(True Negative Rate)
TPR	: Gerçek Pozitif Oran(True Positive Rate)
WBCE	: Ağırlıklandırılmış İkili Çapraz - Entropi
WiG	: Ağırlıklı Sigmoid Kapı Birim Fonksiyonu
YOLO	: Sadece Bir Kez Bak (You Only Look Once)

ŞEKİLLER LİSTESİ

Şekil 2.1 : Evrişimsel sinir ağının mimari yapısı	5
Şekil 2.2 : En sık kullanılan havuzlama yöntemleri.....	13
Şekil 2.3 : Tamamen bağlı katman.....	14
Şekil 2.4 : Adım ve Doğrusal Fonksiyon	24
Şekil 2.5 : Aktivasyon fonksiyonları: (a)Sigmoid, (b) Tanh.....	26
Şekil 2.6 : Düzeltilmiş aktivasyon fonksiyonları: (a)ReLU, (b) Leakly ReLU	28
Şekil 2.7 : ReLU, ELU ve SeLU fonksiyonları	30
Şekil 2.8 : Unuturma tekniği	41
Şekil 2.9 : Eğitim sonuçlarını gösterir grafik	51
Şekil 2.10 : Hata matrisi.....	55
Şekil 3.1 : R-CNN mimarisi.....	61
Şekil 3.2 : Hızlı R-CNN Mimarisi	63
Şekil 3.3 : R-FCN Mimarisi	64
Şekil 3.4 : YOLOv1 modeli	67
Şekil 3.5 : YOLOv5 mimarisi	69
Şekil 3.6 : ELAN ve E-ELAN	70
Şekil 5.1 : Araçların Etiketlenmesi	92
Şekil 5.2: YOLOv7' nin Mimari Yapısı	96
Şekil 5.3 : (a)Eğitim ve doğrulama sonuçları (b)Hata matrisi	108
Şekil 5.4 : (a) Kesinlik eğrisi, (b) Duyarlılık eğrisi	108
Şekil 5.5 : Eğitim ve doğrulama sonuçları	110
Şekil 5.6 : (a) Kesinlik eğrisi, (b) Duyarlılık eğrisi	110
Şekil 5.7 : Eğitim ve doğrulama sonuçları	112
Şekil 5.8 : Kesinlik değerlerinin güven skoru ile karşılaştırılması	114
Şekil 5.9 : Duyarlılık değerlerinin güven skoru ile karşılaştırılması	115
Şekil 5.10 : Kesinlik ve duyarlılık değerlerinin karşılaştırılması.....	117
Şekil 5.11 : F1 skorunun güven skoru ile karşılaştırılması	118
Şekil 5.12 : Hata matrisi.....	119
Şekil 5.13 : 0.25 ve 0.50 güven skoru ile elde edilen kesinlik eğrilerinin güven skorları ile karşılaştırılması	120
Şekil 5.14 : 0.25 ve 0.50 güven skoru ile elde edilen duyarlılık eğrilerinin güven skorları ile karşılaştırılması	121
Şekil 5.15 : 0.25 ve 0.50 güven skoru ile elde edilen kesinlik ve duyarlılık karşılaştırma eğrileri.....	122
Şekil 5.16 : Trafikteki araçların tespit edilip sınıflandırılması	122

TABLÖLER LİSTESİ

Tablo 5.1 : YOLO algoritmasındaki nesnelerin koordinatları ve sınıfları	93
Tablo 5.2 : Veri setinin eğitim, doğrulama ve test seti olarak bölümü	94
Tablo 5.3 : Eğitim, doğrulama ve test setindeki etiket sayıları	95
Tablo 5.4 : Kullanılan donanımlar	98
Tablo 5.5 : Modelin eğitiminde kullanılan parametre değerleri	107
Tablo 5.6: Modelin eğitiminde kullanılan parametre değerleri	109
Tablo 5.7: Modelin eğitiminde kullanılan parametre değerleri	111
Tablo 5.8: Kategorilere göre kesinlik, duyarlılık ve mAP değerleri.....	113
Tablo 5.9 : Farklı sınıfta araçların tespit edilip etiketlenmesi.....	123



Enstitüsü : Lisansüstü Eğitim Enstitüsü
Dalı : Bilgisayar Mühendisliği Anabilim Dalı
Programı : Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Tez Danışmanı : Doç.Dr. Fatma PATLAR AKBULUT
Tez Türü ve Tarihi : Yüksek lisans – Haziran 2024

KISA ÖZET

HAREKETLİ NESNE ANALİZİ İÇİN DERİN ÖĞRENME MİMARİSİ

Okay TÜRKYILMAZ

Bu tez, çeşitli hareketli nesnelerin yüksek hassasiyetle ve gerçek zamanlı olarak tanımlanmasını amaçlamaktadır. Çalışmada, özellikle karayolları başta olmak üzere hava, deniz ve demiryolları üzerinde hareket halinde olan tüm araçlar, araç plakaları ve yayaları tanımaya odaklanılmaktadır. Geliştirilmiş olan model trafik kazası, yaya, araç plakası, bisiklet, üç tekerlekli bisiklet-tripörtör, motorsiklet, atv, kar aracı, jetski, araba, van, limuzin, otobüs-minibüs, kamyon, yarı römorklu kamyon(tır), tankerli kamyon, çöp kamyonu, beton mikserli kamyon, itfaiye aracı, yol süpürme kamyonu, iş makinesi, forklift, traktör, segway, at arabası, helikopter, uçak, gemi-tekne, tren, skuter, araba-polis, araba-taksi, van-ambulans, van-polis ve van-taksi olmak üzere toplam 35 sınıfı kapsamaktadır. Hareketli nesleri tanımanın yanı sıra farklı hava koşulları da modellemenin önemli bir bileşeni olarak belirlenmiştir. Model, dijital kameralardan elde edilen görüntüleri hızlı bir şekilde işleyebilen YOLOv7 algoritması ile geliştirilmiştir. YOLOv7 algoritması görüntü üzerindeki araçları gerçek zamanlı olarak tanımlayarak bu araçların sınıflandırılmasını mümkün kılar. Performansı artırmak ve eğitimi hızlandırmak için COCO veri seti ile önceden eğitilmiş bir YOLOv7 modelindeki öğrenilen bilgiler, özel araç sınıflarını içeren veri setimizin eğitiminde kullanılmıştır. Çalışma, tüm araç sınıflarında yüksek kesinlik ve geri çağırma hızlarına ulaşmayı hedeflemektedir. Çalışmanın YOLOv7 algoritması ile eğitilmesi sonucunda %89 mAP@0.50 ve %84 geri çağırma değerlerine ulaşılarak önemli oranda başarı elde edilmiştir.

Anahtar Kelimeler: Nesne Tespiti, Görüntü İşleme, YOLOv7, Öğrenme Transferi, Derin Öğrenme, Araç Tanıma, Plaka Tanıma, Trafik Kazası Tespit Etme

University : İstanbul Kültür University
Institute : Institute of Graduate Studies
Department : Computer Engineering Master's Program With Thesis
Programme : Computer Engineering Master's Program with Thesis
Supervisor : Assoc.Prof. Fatma PATLAR AKBULUT
Degree Awarded and Date : MA – June 2024

ABSTRACT

DEEP LEARNING ARCHITECTURE FOR MOVING OBJECT ANALYSIS

Okay TÜRKYILMAZ

This thesis aims to identify various moving objects with high precision and in real time. The study focuses on recognizing all vehicles especially on highways, air, sea and railways, vehicle license plates and pedestrians in motion. The developed model consists of a total of 35 classes including traffic accidents, pedestrians, license plates, bicycles, tricycles, motorcycles, ATVs, snowmobiles, jet skis, cars, vans, limousines, buses, minibuses, trucks, semi-trailer trucks, tanker trucks, garbage trucks, concrete mixer trucks, fire trucks, road sweeping trucks, engineering vehicles, forklifts, tractors, segways, carts, helicopters, airplanes, ships-boats, trains, scooters, car-police, car-taxi, van-ambulance, van-police and van-taxi. In addition to recognizing moving objects, different weather conditions have also been identified as an important component of modeling. The model was developed with the YOLOv7 algorithm, which can quickly process images obtained from digital cameras. The YOLOv7 algorithm makes it possible to identify and classify vehicles on the image in real time. To improve performance and speed up training, the learned information from a YOLOv7 model pre-trained with the COCO dataset was used to train our dataset containing custom vehicle classes. The study aims to achieve high accuracy and recall rates in all vehicle classes. As a result of training the study with the YOLOv7 algorithm, significant success was achieved by reaching 89% mAP@0.50 and 84% recall values.

Keywords: Object Detection, Image Processing, YOLOv7, Learning Transfer, Deep Learning, Vehicle Dedection, License Plate Dedection, Traffic Accident Detection

1. GİRİŞ

Son yıllarda, birçok alanda makine öğrenimi kullanımı yaygınlaşarak makine öğrenimi tabanlı uygulamalar popüler hale gelmiştir. Özellikle veri madenciliği, görüntü analizi, doğal dil işleme, diller arası çevrim, konuşma tanıma gibi alanlarda makine öğrenimi tabanlı çalışmalar giderek artmış, bu alanlarda birçok uygulama geliştirilmiştir [1]. Makine öğreniminde birçok sinir ağı türleri geliştirilmiş olup bunlardan özyinelemeli sinir ağları (RvNN), tekrarlayan sinir ağları (RNN) ve evrişimsel sinir ağları (CNN) en popüler olanlarıdır. RvNN doğal dil işleme alanında oldukça etkili olup tahminleri hiyerarşik bir yapıda gerçekleştirir ve bileşimsel vektörleri kullanarak çıktıları sınıflandırabilir. RvNN modeli, alt düğümlerden gelen bilgileri birleştirerek girdi verilerini özyinelemeli olarak işler. RNN de doğal dil işleme, diller arası çevrim, konuşma tanıma gibi alanlarda kullanılan sinir ağı türü olup bir sonraki adımı takip edebilmek için girdiler arasında ilişki kurarlar ve eğitilirken tüm ilişkilerini hatırlarlar. Son olarak CNN ise görüntü analizlerinde, görüntülerin sınıflandırılması ve görüntü üzerindeki nesnelerin tespitinde kullanılan sinir ağı türüdür.

Mimari yapısını canlıların görsel algısından[2] ilham alan evrişimli sinir ağının temeli, 1959 yılında Hubel ve Wiesel'in keşfiyle başlamıştır [3]. 1980 yılında bu çalışmadan esinlenerek, K. Fukusima [4] tarafından hiyerarşik çok katmanlı bir yapay sinir ağı türü önerilmiş, bu çalışma evrişimsel sinir ağı türlerinin gelişiminde ilham kaynağı olmuştur. Bu model CNN' in ilk teorik modeli olarak kabul edilir. 1998 yılında, LeCun v.d. tarafından el yazılı rakamları tanıyan LeNet-5 [5] adlı ilk modern CNN modeli geliştirilmiştir. 2012 yılında AlexNet[6]' in rekor performansından sonra CNN popüler olmaya başlamıştır. Evrişimsel sinir ağı modelinin gelişmesiyle görüntü analizi alanlarındaki çalışmalar son 10 yılda hız kazanmış ve bu alanda çok önemli başarılar elde edilmiştir[7].

Nesne algılama, görüntü veya videodaki nesnelerin tespit edilip görüntü üzerinde gösterilmesi için kullanılan bilgisayarlı görme tekniğidir [8]. Görüntülerin sınıflandırılması yaklaşımından farklı bir teknik olan nesne algılama tekniğinde hem yerleştirme hem de sınıflandırma teknikleri kullanılır. Görüntü üzerindeki

nesnelerin tespiti görüntülerin basit bir şekilde sınıflandırma işleminden daha karmaşık bir problemdir. CNN algoritmaları girdi görüntülerinin tahmin edilip kategorilere ayrılması işlemini gerçekleştirirken, nesne tespit algoritmaları ise yerelleştirme ve sınıflandırma yaklaşımlarını kullanarak görüntü üzerindeki nesnelerin tespit edilmesini sağlar. Yerelleştirme tekniğinde, görüntü üzerindeki nesneler tespit edilir ve nesnelerin etrafı sınırlayıcı kutular içerisine alınır. Sınıflandırma işleminde ise, tespit edilen nesneler sınıflandırılarak sınırlayıcı kutuya etiket ataması yapılır.

Temel nesne algılama metotları geleneksel ve derin öğrenme metotları şeklinde iki ana guruba ayrılmaktadır. Geleneksel nesne algılama metotları üç adımda gerçekleştirilir[7-9]. İlk adımda, görüntünün tamamı incelenip görüntüden bölge önerileri üretilmektedir. Bölge önerilerini üretmek için seçici arama algoritması [10] gibi algoritmalar kullanılmaktadır. İkinci adımda ise, özellik çıkarımı işlemidir. Üretilen bölge önerileri analiz edilerek HOG[11], SIFT[12] ve SURF[13] gibi algoritmalar yardımıyla özellik çıkarımı veya görüntü kalıpları elde edilir. Son adımda da, bir nesne içerip içermediğine bakılmaksızın bu görüntü kalıpları SVM[14] gibi sınıflandırma algoritmalar yardımıyla kategorilere ayrılması sağlanır. Geleneksel tekniklerle nesne tespiti yapılması işleminde bütün görüntü taranır, değişken olan görüntülerde nesneleri görüntü üzerinde yerelleştirmek hesaplama maliyeti açısından oldukça masraflı bir yöntemdir.

Nesne algılamada son yıllarda yaygın olarak kullanılan diğer bir yöntem ise derin öğrenme teknikleri ile gerçekleştirilen metotlardır. Derin öğrenme teknikleriyle geliştirilen algoritmalar ile nesne tespitinin yapılmasında, geleneksel yöntemler ile başarısız sonuçlar elde edilmiş, özellikle CNN ile ilgili çalışmaların yaygınlaşmasından sonra CNN tabanlı birçok nesne algılama algoritmaları geliştirilmiştir. Derin öğrenme tabanlı algoritmalar iki aşamalı ve tek aşamalı olmak üzere iki bölüme ayrılır. İki aşamalı algoritmalar bölgesel tabanlı algoritmalar olup bunlara R-CNN [15], Fast R-CNN [16] ve Faster R-CNN [17] gibi algoritmalar örnek verilebilir. Bölgesel tabanlı algoritmalarda önce bölge önerileri üretilir, CNN ile özellik çıkarımı yapılır, daha sonra elde edilen bu özellikler SVM algoritmaları yardımı ile sınıflandırılır. Bölgesel tabanlı algoritmalarda üretilen bu bölge önerilerinin sayısı yaklaşık 2000 adet olabilmektedir. Çok sayıda bölge önerisinden CNN ile özellik çıkarımının yapılması ve SVM ile kategorilere ayrılması işlemi, hesaplama maliyeti açısından çok pahalı bir yöntem olup gerçek zamanlı

uygulamalarda çok verimli olamamaktadır. Tek aşamalı algoritmalarda ise görüntü ızgaralara bölünür ve sonra her bir ızgara hücresinde CNN yardımı ile bir nesne içerip içermediğinde bakılır. Daha sonra IoU metotları ile bu ızgara hücreler birleştirilerek görüntü üzerindeki nesnelerin tespiti yapılmaktadır. Özellikle YOLO vb. algoritmalar tek aşamalı olarak farklı bir yaklaşımla görüntüyü analiz ederek görüntüdeki nesneleri çok hızlı bir şekilde tespit etmesi nedeni ile son yıllarda gerçek zamanlı uygulamalarda tercih edilmeye başlanmıştır. YOLO algoritmasının [18] ilk sürümü 2016 yılında J.Redmon v.d. tarafından yayımlanan bir makale ile duyurulmuştur. 2016 yılından sonra YOLO algoritmasının birçok sürümü çıkarılmış, özellikle YOLOv5' den sonra farklı ölçeklerde modeller çıkarılmıştır. Küçük ölçekli modellerde eğitim süresi daha hızlı ve doğruluk oranı sınırlıdır. Model boyutu arttıkça eğitim süresi ve nesne tespit süresi artmakta ancak doğruluk oranındaki performans da yükselmektedir.

Tek aşamalı olan YOLO algoritmasının son sürümleri (YOLOv5, YOLOv7, YOLOv8, YOLOv9), ilk versiyona göre bir hayli gelişmiş olduğu, geliştirilen versiyonlarda önemli iyileştirmeler yapıldığı, tespit edilen birçok problemin üstesinden gelinerek görüntü üzerindeki nesneleri tespit etmede yüksek performanslar elde edilmeye başlanmıştır. YOLOv5' den sonra algoritma içerisinde veri artırma teknikleri kullanılarak veri seti zenginleştirilip hem modelin yetersiz öğrenmesine hem de modelin aşırı öğrenmesinin önüne büyük ölçüde geçilerek modelin optimum bir sonuca ulaşmasına yardımcı olmaya başlamıştır. Algoritmada yer alan veri artırma teknikleri sayesinde modelin gerçek dünya ile testinde yüksek performanslar da elde edilmiştir.

Son yıllarda nesne tespiti algoritmaların yaygınlaşmasına bağlı olarak bu algoritmalarla faydalanılmış, özellikle sağlık, güvenlik, gözetleme, endüstri ve trafikteki araçları izleme gibi birçok alanda uygulamalar geliştirilerek bu alanlardaki hizmetlere büyük oranda yardımcı olmaya başlamıştır. Biz de geliştirmiş olduğumuz model de hem trafikteki araçları tespit edip hem de tespit ettiğimiz bu araçları sınıflandırdık. Çalışmamızda hareket halinde olan araç, araç plakası ve yaya gibi nesneleri tespit edip sınıflandırmaya çalıştık. Modelimiz araç plakası, bisiklet, üç tekerlekli bisiklet-tripörtör, motorsiklet, atv, kar aracı, jetski, araba, van, limuzin, otobüs-minibüs, kamyon, yarı römorklu kamyon(tır), tankerli kamyon, çöp kamyonu, beton mikserli kamyon, itfaiye aracı, yol süpürme kamyonu, iş makinesi, forklift, traktör, segway, at arabası, helikopter, uçak, gemi-tekne, tren, yaya, trafik kazası,

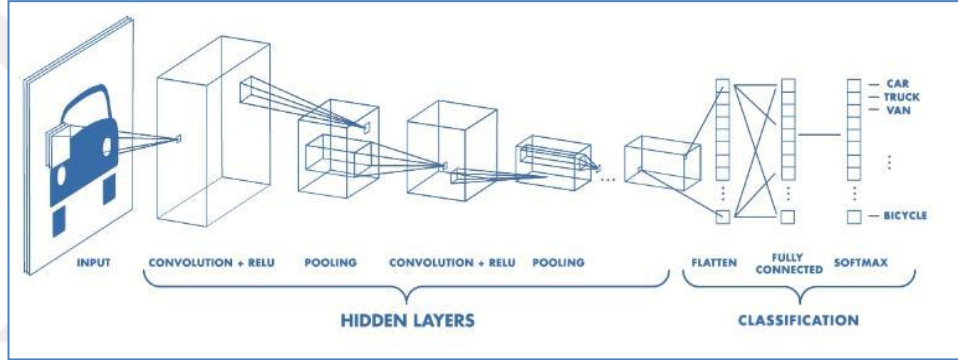
skuter, araba-polis, araba-taksi, van-ambulans, van-polis, van-taksi olmak üzere toplam 35 sınıftan oluşmaktadır. Geliştirmiş olduğumuz çalışmaya benzer çalışmalar yapılmış, ancak biz bu çalışmaları biraz daha genişleterek araçları kendi aralarında birçok sınıfa ayırmayı başardık. Ayrıca çalışmamızda araçların plaka numaralarını da sınıf kategorisine ekleyerek araçların plakalarını tespit etmeyi de hedefledik.

Çalışmamızda, trafikte hareket halinde olan araçları toplam 35 sınıfa ayırdık. Kameralardan elde edilen görüntüleri YOLO algoritması sayesinde gerçek zamanlı olarak analiz ederek istenilen sınıftaki araçları tespit etme olanağı vermiş olduk. Örneğin, tankerli araçların yasaklı tünellerden geçişini önlemek için bu araçları zamanında tespit edip gerekli olan önlemler zamanında alınabilir. Başka bir örnek vermek gerekirse, kalabalık olan özellikle İstanbul, Ankara, İzmir gibi şehirlerde karayolundaki trafik yoğunluğunu azaltmak ve trafik akışını hızlandırmak için belirli saatlerde tır, kamyon vb. araçların girmesi yasaklanabilir. Dolayısıyla, kara yollarını gözetleyen kameralardan elde edilen görüntüler bu modelimiz sayesinde işlenerek istenilen sınıftaki araçlar zamanında tespit edilip ilgili yollara yönlendirilebilir veya belirli saatlerde şehre girişi engellenebilir. Yapmış olduğumuz çalışma sayesinde, araçları birçok gruba ayırarak yukarıdaki örneklere benzer akla gelmeyen birçok farklı amaçlarla da kullanılabilir.

Modelimizi geliştirmeden önce öncelikle evrimsel sinir ağı hakkında çok geniş literatür taraması yapılarak bu alanlarda yapılan çalışmalar, sinir ağında kullanılan teknikler, sinir ağının performansını etki eden problemler, bu problemlerin iyileştirilmesinde kullanılan teknikler çok yönlü ve kapsamlı bir şekilde araştırılmıştır. Daha sonra nesne algılama algoritmalarının geliştirilmesindeki süreç, karşılaşılan zorluklar, kullanılan teknikler, modelin performansını etki eden problemler ve bu problemlere getirilen çözümler detaylı bir şekilde incelenmiştir. Bu çalışmada modeli iyi bir şekilde eğitebilmek, modelin performansını olabildiğince artırabilmek, eğitim aşamasında yaşanacak olan problemlerin nedenini anlayabilmek, bu problemlere çözüm getirebilmek için hem evrimsel sinir ağının hem de nesne tespit algoritmalarının çalışma prensibinin çok iyi bir şekilde kavranılması gerekmektedir.

2. EVRİŞİMSEL SİNİR AĞLARI

Mimari yapısını canlıların görsel algısından ilham alan evrişimsel sinir ağları, özellikle görüntülerin tanınmasında ve sınıflandırılmasında yaygın olarak kullanılan çok katmanlı bir sinir ağı türüdür[2]. Evrişimli sinir ağının temeli 1959 yılında Hubel ve Wiesel' in keşfiyle başlamış[3], 1980 yılında da bu çalışmadan esinlenilerek K. Fukusima [4] tarafından evrişimsel sinir ağı türlerine ilham kaynağı olan ve ilk teorik model olarak kabul edilen hiyerarşik çok katmanlı bir yapay sinir ağı türü önerilmiştir. 1998 yılında, LeCun v.d. tarafından el yazılı rakamları tanıyan LeNet-5 [5] adlı ilk modern CNN modeli geliştirilmiş, 2012 yılında da A.Krizhevsky v.d. tarafından görüntülerin sınıflandırılması için AlexNet[6]' in tanıtılmasıyla CNN popüler olmaya ve birçok alanda yaygın olarak kullanılmaya başlamıştır [1, 19-22].



Şekil 2.1 : Evrişimsel sinir ağının mimari yapısı

Çok katmanlı algılayıcıya(MLP) benzeyen ve yaygın olarak kullanılan CNN evrişimsel, aktivasyon, havuzlama ve tam bağlantılı katman olmak üzere çok sayıda katmandan oluşabilmektedir. CNN mimarisinin yapısını gösteren bir örnek şekil 2.1’de gösterilmektedir. Evrişimsel katmanda özellik haritaları(feature maps) elde etmek için girdi görüntüleri filtreler yardımı ile evrimleştirilirken, çıkan değerler aktivasyon katmanında normalize edilir. Havuzlama katmanında ise önceki katmanlardan gelen özellik haritalarından alt örneklemeler yapılır, önemli bilgilerin korunması ve bu bilgilerin baskın olması sağlanır. Tam bağlantılı katmanda, havuzlama katmanından özellik haritaları tek boyutlu vektörlere dönüştürülür, softmax ve sigmoid gibi aktivasyon fonksiyonları yardımı ile modelde belirlenmiş olan her bir sınıf için sınıflandırma puanları hesaplanarak üretilir. Üretilen sınıflandırma puanlarından en yüksek olanı modelin tahmin ettiği sınıftır.

CNN modeli içerisinde girdi görüntüleri yükseklik(m), genişlik(m) ve derinlik(r) olmak üzere üç boyuta ($m \times m \times r$) sahiptir. Derinlik kanal numarası

olarak da adlandırılabilir. RGB özelliğine sahip girdi görüntüsünde kırmızı, yeşil ve mavi olmak üzere üç kanal vardır. Girdi görüntüsüne benzer şekilde evrişim katmanında girdi görüntüsüne uygulanan filtrelerde yükseklik(n), genişlik(n) ve derinlik(k) olmak üzere üç boyuta($n \times n \times k$) sahiptir. Katmanda görüntüye uygulanan filtrelerin derinliği(k) girdi görüntünün derinliği(r) ile aynı boyuta sahiptir.

2.1. Evrişimsel Sinir Ağının Basit Çalışma Prensipleri

Evrişimsel sinir ağında bir modelin eğitim sürecinde ileri ve geri yayılım algoritmaları kullanılır. Girdi görüntüleri ilk önce değerleri rastgele başlatılmış olan ağırlık filtreleri ile evrimleştirilir, bu evrimleştirme sonucu özellik haritaları elde edilir, özellik matrislerindeki değerler aktivasyon fonksiyonları ile normalize edilir ve daha sonra havuzlama katmanında bu özellikler baskın hale getirilir. Elde edilen çok sayıda çıktı özellikleri tamamen bağlı katmanda düzleştirilir, bilgiler ağırlık matrisleri ile çarpılarak nöronlar elde edilir. Bu nöronların sayısı ağırlık matrisinin boyutuna bağlıdır. Bu sürecin sonunda sınıflandırma işlemi gerçekleştirilerek sınıflandırma puanları üretilir. En yüksek sınıflandırma puanı modelin tahmin ettiği sınıftır. Girdi katmanından başlayıp çıktı katmanına doğru olan bu sürece ileri yayılım algoritması olarak adlandırılır. İleri yayılım algoritmasının sonucunda gerçek görüntü ile modelin tahmin ettiği görüntü verileri karşılaştırılır, kayıplar ve bu kayıpların gradyanları hesaplanarak geriye doğru bütün ağırlık matrisleri tekrar güncellenir. Çıktı katmanından girdi katmanına doğru gerçekleştirilen bu ağırlık güncellemeleri geri yayılım ve optimizasyon algoritmaları ile gerçekleştirilir. Geri yayılım sırasında, kayıplar en aza indirgenecek şekilde calculus zincir kuralı ile hesaplanır, ağırlıklar ve sapmalar optimizasyon algoritmaları ile tekrar güncellenir. Çıktı katmanından girdi katmanına doğru olan bu süreç geri yayılım algoritması olarak adlandırılır [23]. İleri yayılım ve geri yayılım algoritmaları modelin eğitimi sırasında birçok kez tekrarlanır. Bu süreç eğitim döngüsüdür ve eğitim döngüsü hatalar en aza indirilene ve model iyi bir performans gösterene kadar devam edebilir.

2.2. Evrişimsel Sinir Ağı Katmanları

CNN mimarisi, evrişim, aktivasyon, havuzlama ve tamamen bağlı katman gibi çok sayıda katmana sahip olup her bir katman kendine özgü görevlere sahiptir.

Evriřim katmanında girdi görüntüleri belirlenmiř filtreler aracılıęı ile evrimleřtirilerek görüntülerden özellik haritaları çıkartılır. Bu özellik haritaları aktivasyon katmanından geçirilerek veriler normalize edilir. Havuzlama katmanında alt örneklemeler yapıldıktan sonra bu veriler tamamen baęlı katmana iletilir. Tamamen baęlı katmanda ise veriler tek vektör boyutuna çevrilir ve aktivasyon fonksiyonları yardımı ile görüntünün sınıflandırma iřlemi gerekleřirilir.

2.2.1. Evriřimsel Katman

Evriřim konsepti özellikle görüntü iřleme alanlarında kullanılan matematiksel bir iřlem olup temeli LeCun tarafından ortaya atılmıřtır[24]. Evriřimsel katman, evriřimsel sinir aęının en önemli katmanı olup ekirdek olarak adlandırılan bir dizi filtreye sahiptir. Bu katmanda, evriřimsel bir filtre kullanılarak her bir nöron iin girdilerin aęırlıklı toplamı üretilir[25, 26]. Bir CNN mimarisi birden fazla evriřimsel katmana sahip olup her bir katmanda birden fazla farklı filtre kullanılır[1, 2]. Bu filtreler ile görüntüler evrimleřtirilerek görüntülerden özellik haritaları elde edilir. Gizli katman ierisinde yer alan her bir evriřimsel katmanda farklı “n” sayıda filtre kullanılarak girdi görüntüsü evrimleřtirilir ve görüntüden “n” sayıda özellik haritası oluřturulabilir[27, 28]. Kenar filtresi ile yapılan evriřim sonrasında görüntüdeki kenarlar tespit edilirken bulanıklařtırma filtresiyle yapılan evriřimde ise girdi görüntülerindeki bulanık veriler elde edilir. Evrimleřtirmede kullanılan bu filtreler sayesinde girdi görüntüsünün köřeleri, lekeleri, kenarları, eęrileri, řekilleri, yapıları, dokuları, yoğunluk kalıpları, evrensel anlamda soyut ve karmařık ayrıntıları tespit edilir. Ayrıca, modelin öęrenebilmesini ve genelleme yapabilmesini saęlamak iin anlamlı bilgileri tutmasını da saęlar.

2.2.1.1. Evriřimde Kullanılan Filtreler

Filtreler, girdi görüntülerinden belirli özellik haritalarını elde etmek iin kullanılan küçük matrislerden oluřur. Ü boyutlu RGB girdi görüntülerinden özellik haritaları elde etmek iin üç boyutlu ($n \times n \times k$) filtreler kullanılır. Evriřim iřlemlerinde kullanılan bu filtreler He[29] ve Xavier[30] gibi teknikler ile rastgele deęerler ile bařlatılır ve eęitim sırasında geri yayılım algoritması sayesinde tekrar güncellenerek modelin girdi görüntüsündeki anlamlı bilgileri öęrenmesi saęlanır.

Girdi görüntülerinden anlamlı bilgiler elde etmek iin kullanılan küçük boyutlu bu filtrelerin her bir elemanı aęırlık olarak adlandırılır. İki boyutlu 3×3 filtre matrisi 9 aęırlıęa sahip iken, üç boyutlu $3 \times 3 \times 3$ filtre matrisi 27 aęırlıęa

sahiptir. Bu ağırlıklar bazı teknikler kullanılarak rastgele değerler ile başlatılır, modelin eğitim süreci içerisinde geri yayılım algoritması ile tekrarlı bir şekilde güncellenir.

RGB görüntüleri $x = x_1 + x_2 + x_3$ şeklinde üç kanala sahiptir. Girdi görüntüsünün her bir kanalı farklı ağırlıklar ile evrimleştirilir. Tek bir evrişim katmanında çok sayıda ($n \times n \times k$) evrişim filtresi kullanılır. Her bir evrişim filtresi ile evrimleştirilen girdi görüntüsünden filtre sayısı kadar özellik haritaları elde edilir. Üç boyutlu bir RGB görüntüsünde $m \times (n \times n \times 3)$ filtre kullanılır ise girdi görüntüsünden “m” sayısı kadar özellik haritası çıktısı elde edilir. Bir katmanda çok sayıda filtre kullanılması modelin karmaşıklık yapısını artırır ve modelin veri setindeki görüntüler üzerinden çok detaylı bilgiler öğrenmesi sağlanır. Ancak modelin bu karmaşıklığı modelin aşırı uyum göstermesine de neden olabileceğinden kullanılan filtrelerin ve ayrıca katman sayısının iyi ayarlanması gerekir. Az sayıda filtre ve katman kullanılması da modelin yetersiz öğrenmesine de neden olabilir. Aslında modelin karmaşıklık yapısı veri setinin büyüklüğü ve görüntülerin çözünürlüklerin kalitesi ile orantılı olmalıdır.

2.2.1.2. Evrişimsel Operasyon

Evrişimsel operasyon girdi görüntülerinden özellik haritaları elde etmek için yapılan bir işlemdir. Gri tonlamalı görüntü formatı tek kanal içerirken, RGB formatındaki görüntüler üç kanal içerir. Evrişim işlemini anlamak için ağırlıkları rastgele başlatılmış 3×3 filtre ile 6×6 gri tonlamalı görüntü örneğini ele alalım.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} \\ x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \quad (2.1a)$$

$$y_{11} = x_{11} \cdot w_{11} + x_{12} \cdot w_{12} + x_{13} \cdot w_{13} + x_{21} \cdot w_{21} + x_{22} \cdot w_{22} + x_{23} \cdot w_{23} + x_{31} \cdot w_{31} + x_{32} \cdot w_{32} + x_{33} \cdot w_{33} \quad (2.1b)$$

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix} \quad (2.1c)$$

Denklem 2.1a’ da belirtilen X matrisi 6×6 boyutuna sahip girdi görüntüsü ve W matrisi de 3×3 boyutuna sahip ağırlıkları rastgele başlatılmış evrimleştirme filtresidir. Denklem 2.1b’ de evrimleştirme işlemindeki hesaplama örneği

gösterilmiştir. Denklem 2.1b' deki evrimleştirme işlemi, filtre soldan sağa ve daha sonra yukarıdan aşağı bir adım kaydırılarak görüntünün tamamı için uygulanır ve elde edilen skalar sonuç değeri denklem 2.1c' de görülen “Y” matrisinde karşılık gelen kutucuklara yazılır. Denklem 2.1c' de görüldüğü üzere 6×6 girdi görüntüsü, 3×3 evrimleştirme filtresi ile evrimleştirildikten sonra elde edilen özellik haritası matrisinin boyutu küçülmüştür. Evrimleştirme işleminde adım sayısı 1 olarak belirlenmiştir. Özellik haritası matrisinin boyutu filtrenin boyutuna ve kullanılan adımlama tekniğine göre değişiklik gösterebilir. Evrimleştirme işlemi yapılırken adım değeri arttırıldığında elde edilen özellik haritası matrisinin boyutu küçülürken, adım sayısının değeri azaltıldığında da özellik haritası matrisinin boyutu artmaktadır. Aynı şekilde evrimleştirme filtresinin boyutu arttıkça da elde edilen özellik haritası matrisinin boyutu küçülmekte, filtrenin boyutu küçüldükçe özellik haritası matrisinin boyutu artmaktadır. Birden fazla evrimleştirme katmanına sahip olan CNN mimarisinde her katmanda yapılan evrimleştirme işlemi, elde edilen özellik haritası matrislerinin küçülmesine ve bu matrisler sonraki evrimleştirme katmanında da yapılan evrimleştirme işlemi sonucunda matrisin boyutunun daha da küçülmesine sebebiyet verir. Dolayısıyla girdi görüntülerinden elde edilen bilgilerin kaybolmasına yol açar. Evrimleştirme sonucu elde edilen özellik haritası matrisinin girdi görüntüsü ile aynı olabilmesi için dolgulama gibi tekniklerden faydalanılır. Özellik haritası matrisine uygulanan dolgulama tekniği sayesinde özellik haritalarının boyutu korunmuş olur.

Evrimeleştirme işlemini üç kanallı bir görüntü üzerinde yapıldığında, görüntünün her bir kanalı için ayrı filtreler kullanılarak evrimleştirme işlemi gerçekleştirilir. Farklı filtrelerle üç kanallı görüntünün evrimleştirme işlemi toplanır ve tek bir çıktı elde edilir. Bu filtreler eğitilebilir parametrelerdir, elamanları ağırlık olarak adlandırılırlar ve geri yayılım sırasında devamlı güncellenirler. Ayrıca modele daha fazla esneklik katmak ve modelin daha iyi öğrenebilmesini sağlamak için evrimleştirme işleminin sonucuna bir sapma katsayısı eklenir. Modelde birden fazla katman kullanılır ise her bir katmanda aynı şekilde birden fazla $n \times n \times 3$ boyutlu farklı filtreler kullanılarak istenildiği kadar özellik haritaları oluşturulabilir. Evrimleştirme işleminin genelleştirilmiş hali denklem 2.2' de tanımlanmıştır.

$$Y_i = B_i + \sum_{j=1}^c X_j * W_{ij} \quad , \quad i = 1 \dots d \quad (2.2)$$

Burada c görüntüdeki kanal sayısını, d çıktı sayısını, Y_i çıktı matrisini, B_i sapma matrisini, X_j girdi matrisinin bir kanalını, W_{ij} girdi görüntüsünün kanal matrisinde kullanılacak ağırlık matrisini temsil etmektedir. Evrişim işleminde, $d \times c$ sayısı kadar filtre kullanılır. Bir görüntüden “ d ” sayısı kadar özellik haritaları bu şekilde oluşturulabilir. Bu işlem ayrı katmanlarda gerçekleştirilir. Her bir katmanın çıktısı, kullanılan filtrelerin farklılık göstermesi sebebi ile farklı olacaktır. Evrişim işlemi tamamlandıktan sonra elde edilen bu matris özellik haritası olarak adlandırılır. Evrişimden çıkan bu matrisin değerleri aktivasyon işleminden geçmesi için aktivasyon katmanına aktarılır.

2.2.1.3. Özellik Haritaları

Evrileştirme katmanında girdi görüntüsü, evrileştirme filtresi ile evrileştirildikten sonra özellik haritaları(feature maps) elde edilir. Özellik haritaları, görüntülerden elde edilen kenarlar, yapılar gibi anlamlı bilgileri içerir. Bu bilgiler, görüntünün sınıflandırılmasında ve tanınmasında önemli rol oynar. Evrileştirme işlemi sonucunda elde edilen özellik haritası matrisi, kaydırma ve filtre boyutuna göre değişiklik gösterir. Eğer, evrileştirmeden sonra özellik matrisi küçülürse sonraki evrileştirme işlemine iletilmeden önce bu matrise dolgulama işlemi yapılarak matrisin sonraki katmanlarda daha da küçülmesi engel olunur ve matristeki bilgilerin korunması sağlanır.

2.2.1.4. Adımlama İşlemi

Evrişim filtresinin giriş görüntüsü üzerinde nasıl hareket edeceğine adım ile belirlenir. Adım, filtrenin görüntü üzerinde kaç konum kaydırılacağını belirtir [31]. Adımın varsayılan değeri 1 olarak kabul edilir. Adım ne kadar büyük olursa, özellik haritası o kadar küçülmektedir ve bilgi kayıplarına neden olmaktadır ancak hesaplama maliyeti de düşmektedir. Adımın boyutu küçüldüğünde ise, özellik haritası matrisinin boyutu artar, ayrıntılar detaylı bir şekilde yakalanır fakat hesaplama maliyetinin de artmasına neden olur [25]. 1 ve 2 boyutlu adımların kullanılması daha yaygındır. 3 boyutlu adımlar daha çok alt örnekleme ihtiyacı duyulduğu zaman kullanılmaktadır. Çeşitli adım boyutları farklı katmanlarda da kullanılabilir.

2.2.1.5. Dolgulama İşlemi

Evrimleştirme katmanında yapılan evrimleştirme operasyonun sonucunda özellik haritası matrisinin boyutu küçülür. Küçülen matris bir sonraki katmana iletilmeden önce matrisin boyutunun aynı kalmasını sağlamak için matrise dolgulama işlemi gerçekleştirilir. Aksi takdirde, evrişim işleminin her adımında özellik haritası matrisinin küçülmesi, daha derin katmanlarda bilgi kaybına neden olabilir. Dolgulama, geçerli dolgulama, sıfır dolgulama, tam dolgulama, dolgu yansıtma, dolgu çoğaltma ve dairesel dolgulama gibi tekniklerden oluşur [25, 32, 33].

Sıfır dolgulama işleminde, evrişim işlemi sonucu boyutu küçülen özellik matrisinin etrafı sıfırlar ile dolgulama yapılarak matrisin boyutu girdi görüntüsünün boyutu ile aynı kalması sağlanır. Denklem 2.3a' da özellik matrisi, denklem 2.3b' de sıfır dolgulama işlemi sonucu elde edilen özellik matrisi görülmektedir.

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix} \quad (2.3a)$$

$$Y' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_{11} & y_{12} & y_{13} & y_{14} & 0 & 0 \\ 0 & 0 & y_{21} & y_{22} & y_{23} & y_{24} & 0 & 0 \\ 0 & 0 & y_{31} & y_{32} & y_{33} & y_{34} & 0 & 0 \\ 0 & 0 & y_{41} & y_{42} & y_{43} & y_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.3b)$$

Dolgulama yapıldıktan sonra, mekânsal boyutlar korunur, kenarlardaki özellikler yakalandığı için bilgi kaybı önlenir. Ancak, hesaplama maliyeti artar, özellik haritalarının kenarlarında sahte yapılar görünür.

Tam dolgulama işleminde, evrişim filtresi girdi görüntünün kenarları ve köşeleri dâhil olacak şekilde uygulanarak çıktı matrisinin aynı kalması sağlanır. Dolgu yansıtma işleminde ise, çıktı matrisinin kenar değerleri matrisin kenarlarına yansıtılarak genişletilir. Bu sayede, kenar etkileri minimize edilir ve orijinal görüntü korunmuş olur. Dolgu çoğaltma tekniğinde, özellik matrisinin kenar değerleri matrisin dışına doğru eklenerek çoğaltma işlemi gerçekleştirilir.

2.2.2. Aktivasyon Katmanı

Bu katmanda, evrişim katmanından gelen özellik haritalarının değerleri aktivasyon işleminden geçirilerek değerler normalize edilir. Bu işlem modele doğrusal olmama durumu katarak modelin öğrenebilmesini sağlar. Evrişimsel sinir

ağlarının gizli katmanlarında genellikle daha basit olan ReLU ve sızdıran ReLU gibi düzleştirici aktivasyon fonksiyonları kullanılır. Bu fonksiyonların basitliği hesaplama maliyetini azaltır. Çünkü bir evrişimsel sinir ağı çok sayıda evrişimsel katmana sahiptir. Her evrişim sonrası elde edilen özellik haritalarının karmaşık matematiksel işlemlerden geçirilmesi ağıın eğitim sürecini ciddi oranda uzatacaktır. Dolayısıyla gizli katmanlardaki aktivasyon işlemlerinde ReLU gibi düzleştirici fonksiyonların kullanılması, modelin eğitim sürecine olumlu yönde ciddi oranda katkı sağlayacaktır. Denklem 2.4a bir önceki evrişim katmanından gelen çıktı matrisini temsil eder. Bu çıktı matrisi aktivasyon katmanına girdi matrisi olarak aktarılacak her bir elemanı aktivasyon işleminden geçer. Denklem 2.4b’ de aktivasyon işlemi ve ReLU fonksiyonu tanımlanmıştır.

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & y_{16} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} & y_{26} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} & y_{36} \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45} & y_{46} \\ y_{51} & y_{52} & y_{53} & y_{54} & y_{55} & y_{56} \\ y_{61} & y_{62} & y_{63} & y_{64} & y_{65} & y_{66} \end{bmatrix} \quad (2.4a)$$

$$a_i = \sigma(y_i), \quad a_i = ReLU(y_i) = (0, y_i) \quad (2.4b)$$

Burada, y_i Y girdi matrisindeki elemanı, σ aktivasyon fonksiyonunu, a_i ise aktive edilmiş değeri temsil eder. Y girdi matrisinin bütün elemanları aktivasyon işleminden geçtikten sonra elemanları aktive edilmiş A aktivasyon matrisi elde edilir. Denklem 2.4c’ de aktivasyon işleminden geçen A matrisi gösterilmektedir.

$$A = \begin{bmatrix} \sigma(y_{11}) & \sigma(y_{12}) & \sigma(y_{13}) & \sigma(y_{14}) & \sigma(y_{15}) & \sigma(y_{16}) \\ \sigma(y_{21}) & \sigma(y_{22}) & \sigma(y_{23}) & \sigma(y_{24}) & \sigma(y_{25}) & \sigma(y_{26}) \\ \sigma(y_{31}) & \sigma(y_{32}) & \sigma(y_{33}) & \sigma(y_{34}) & \sigma(y_{35}) & \sigma(y_{36}) \\ \sigma(y_{41}) & \sigma(y_{42}) & \sigma(y_{43}) & \sigma(y_{44}) & \sigma(y_{45}) & \sigma(y_{46}) \\ \sigma(y_{51}) & \sigma(y_{52}) & \sigma(y_{53}) & \sigma(y_{54}) & \sigma(y_{55}) & \sigma(y_{56}) \\ \sigma(y_{61}) & \sigma(y_{62}) & \sigma(y_{63}) & \sigma(y_{64}) & \sigma(y_{65}) & \sigma(y_{66}) \end{bmatrix} \quad (2.4c)$$

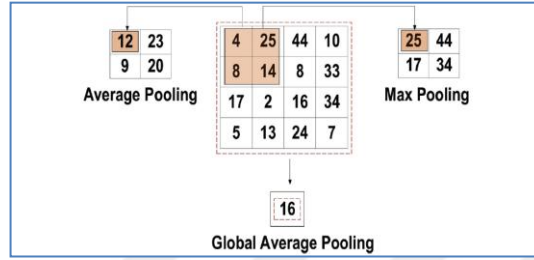
Aktivasyon işleminden sonra elde edilen A matrisi bir sonraki katman olan Havuzlama katmanına aktarılır.

2.2.3. Havuzlama Katmanı

Evrişim işlemi ile elde edilen özellik haritaları aktivasyon işleminden sonra havuzlama katmanına aktarılır. Havuzlama katmanında, bir önceki katmandan gelen çıktı verilerinden alt örnekleme yapılır[1]. Parametre sayısı azaldığından matrisin boyutu küçülür. Boyutu küçülen matrisin hesaplama karmaşıklığı ve maliyeti de azalmış olur [34]. Havuzlama aşamasının her adımında baskın bilgilerin çoğunluğu

korunur. Evrişim işlemine benzer şekilde, havuzlama işleminden önce hem adımın hem de filtrenin boyutu, başlangıçta atanır.

Havuzlama katmanında ağaç, kapılı, ortalama, minimum, maksimum, küresel ortalama ve küresel maksimum havuzlama gibi farklı türde havuzlama yöntemleri mevcuttur. Maksimum, minimum ve küresel ortalama havuzlama en bilinen ve sıklıkla kullanılan havuzlama yöntemleri arasında yer alır. Şekil 2.2'de sıklıkla kullanılan havuzlama yöntemleri gösterilmektedir.



Şekil 2.2 : En sık kullanılan havuzlama yöntemleri

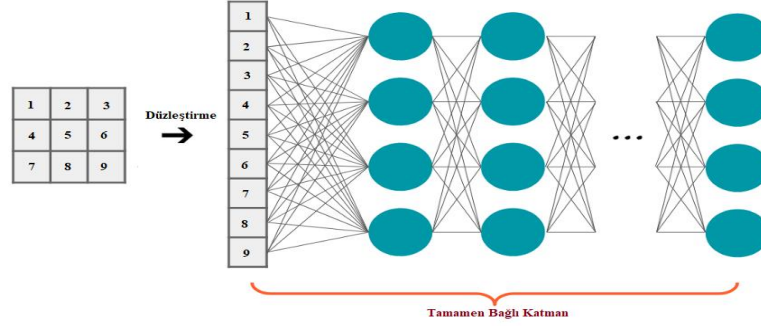
CNN mimarisindeki havuzlama katmanının eksikliğine bağlı olarak CNN modelinin performansının düşmesine neden olabilir. Havuzlama katmanı, modelin aşırı öğrenmesine engel olurken ağırlıkların hesaplanmasının azaltılmasına da yardımcı olur[19]. Denklem 2.5’ de bir önceki katmandan gelen 6×6 çıktı matrisi görülmektedir. Bu matrise 2×2 maksimum havuzlama filtresi ve adım sayısını “1” olarak tanımlar ise havuzlama işleminden sonra yeni çıktı matrisi küçülerek 5×5 boyutunda olur. Daha büyük girdi görüntülerinde daha büyük boyutta havuzlama filtreleri ve adım sayısı kullanılır ve girdi verisi daha da küçülür.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix}, \quad P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{15} \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} \\ p_{41} & p_{42} & p_{43} & p_{44} & p_{45} \\ p_{51} & p_{52} & p_{53} & p_{54} & p_{55} \end{bmatrix} \quad (2.5)$$

Havuzlama işleminden sonra elde edilen P matrisinde, girdi görüntüsündeki baskın bilgiler korunmuştur. Eğer ağın mimarisinde çok sayıda evrişim, aktivasyon ve havuzlama katmanından oluşursa bu işlemler tekrarlanır. Son katmandan çıkan özellik matrisi bir sonraki katman olan tamamen bağlı katmanına aktarılır.

2.2.4. Tamamen Bağlı Katman

Bir önceki katmandan gelen çok boyutlu çıktı verileri tek boyuta dönüştürülerek tamamen bağlı katmana girdi olarak aktarılarak nöronlar elde edilir[35]. Şekil 2.3.’ de tamamen bağlı katman gösterilmektedir.



Şekil 2.3 : Tamamen bağlı katman

Tamamen bağlı katmandaki nöron sayısı bir parametre ile belirlenir. Tamamen bağlı katmandaki her bir nöron, önceki katmandaki bütün nöronlara bağlantılıdır. Bu her bir bağlantı, ağırlık değerine sahiptir. Ayrıca tamamen bağlı katmandaki her bir nöron bir sapma değerine de sahiptir. Şekil 2.3’ deki tamamen bağlı katmanın ilk kısmında 4 nöron ve kendisinden önceki katmanda 9 düğüm mevcuttur. Oluşturulacak olan ağırlık matrisi 4×9 , sapma matrisi ise 4×1 boyutlu olur. Denklem 2.6’ da önceki katmandan gelen X matrisi, tamamen bağlı katman için oluşturulan W ağırlık matrisi ve B sapma matrisi görülmektedir.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} \quad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \\ w_{61} & w_{62} & w_{63} & w_{64} \\ w_{71} & w_{72} & w_{73} & w_{74} \\ w_{81} & w_{82} & w_{83} & w_{84} \\ w_{91} & w_{92} & w_{93} & w_{94} \end{bmatrix}, \quad B = [b_1 \quad b_2 \quad b_3 \quad b_4] \quad (2.6)$$

Tamamen bağlı katmandaki her bir nöronun kendisindeki önceki katmandaki bütün nöronlara olan bağlantısının ağırlıklı toplamı denklem 2.7’ de gösterilen formül ile hesaplanabilir.

$$z_j = b_j + \sum_{i=1}^n x_i * w_{ij}, \quad j = 1 \dots d \quad (2.7)$$

Burada z_j tamamen bağlı katmandaki j^{th} nöronu, b_j nörona ait sapma değerini, n önceki katmanın nöron sayısını, d katmandaki nöron sayısını, x_i önceki katmandaki i^{th} nöronu, w_{ij} tamamen bağlı katmandaki her bir nöronun önceki katmandaki bütün nöronlara olan bağlantısındaki ağırlık değerini temsil eder.

Evrimsel sinir ağları çok sayıda tamamen bağlı katmana sahip olabilir. Her bir katmanın nöron sayısı değişiklik gösterebilir. Bir katmandan çıkan çıktı verileri aynı şekilde bir diğer tamamen bağlı katmana girdi olarak aktarılabilir. Aynı şekilde

ikinci katmanın nöronları bir önceki katmanın bütün nöronları ile bağlantılıdır ve her bir bağlantı yine bir ağırlık değerine sahiptir. Bu ağırlık matrisleri her bir katmanda farklıdır. İkinci katman olan tamamen bağlantılı katmanın bütün nöronları aynı şekilde denklem 2.7' deki formül ile hesaplanır ve farklı nöron değerleri elde edilir.

Tamamen bağlı katmanın son katmanında sınıflandırma yapılması için softmax gibi aktivasyon fonksiyonları kullanılır. Son katmandan önceki tamamen bağlı katmanlar ReLU gibi düzleştirici fonksiyonlar ile aktive edilmesi eğitim sürecini kısaltır. Ayrıca bu katmanların sonunda unutturma teknikleri kullanılabilir. Unutturma tekniği ağıın genelleme yeteneğini iyileştirmek için yapılır. Bu konu sonraki bölümlerde detaylı bir şekilde anlatılacaktır. Örnek olarak tamamen bağlı katman 2 katmandan oluşuyor ise aşağıdaki adımlar izlenir.

a) 1.Tamamen bağlı katman

Bu katmanda kullanılan ağırlık ve sapma matrisleri farklı oluşturulur. Bu katmanda nöronların bütün değerleri denklem 2.8a' daki formül ile hesaplanır. Daha sonra nöron değerleri ReLU gibi hesaplaması kolay düzleştirici fonksiyonlar ile aktive edilir. Denklem 2.8b' de aktivasyon fonksiyonu tanımlanmıştır. Aktive edilmiş nöronlar, unutturma tekniği ile unutturma parametresi kadar rastgele sıfırlanarak unutturulur. Denklem 2.8c' de unutturma fonksiyonu tanımlanmıştır.

$$z_j^1 = b_j^1 + \sum_{i=1}^n x_i * w_{ij}^1, \quad j = 1 \dots m \quad (2.8a)$$

$$a_j^1 = \sigma(z_j^1), \quad a_j^1 = ReLU(0, z_j^1) \quad (2.8b)$$

$$D^1 = Unutturma(A^1, unutturma oranı) \quad (2.8c)$$

Burada z_j^1 katmandaki nöron değerini, b_j^1 katmandaki sapma değerini, x_i önceki katmandan gelen çıktı matrisinin i^{th} sırasındaki değeri, w_{ij}^1 ağırlık matrisindeki değeri, n önceki katmandan gelen nöronların sayısını, m bu katmanın nöron sayısını, a_j^1 aktive edilmiş değeri, A^1 aktive edilmiş matrisi, D^1 unutturma sonrası elde edilen matrisi temsil eder. Daha sonra bu katmadaki bütün değerler bir sonraki tamamen bağlı katmana girdi olarak aktarılır.

b) 2.Tamamen bağlı katman

Bu katman tamamen bağlı katmanın son katmanıdır. Tamamen bağlı katmanın son katmanında sınıflandırma işlemi gerçekleştirilir. Önceki katmandan gelen değerlerin sınıflandırma yapılabilmesi için sınıflandırma işlemlerinde sık kullanılan softmax fonksiyonundan faydalanılır. Bu katmanın nöronlarını elde etmek

için yine denklem 2.9a' daki formül kullanılır. Denklem 2.9b' de normalizasyon fonksiyonu ve denklem 2.9c' de sınıflandırma işlemi için softmax fonksiyonu tanımlanmıştır.

$$z_j^2 = b_j^2 + \sum_{i=1}^n d_i^1 * w_{ij}^2, \quad j = 1 \dots m \quad (2.9a)$$

$$Z^3 = Toplu Normalizasyon(Z^2) \quad (2.9b)$$

$$y_j = Softmax\left(\frac{e^{z_j^3}}{\sum_{k=1}^k e^{z_k^3}}\right) \quad (2.9c)$$

Burada z_j^2 katmandaki nöron değerini, b_j^2 katmandaki sapma değerini, d_i^1 önceki katmandan gelen çıktı matrisinin i^{th} sırasındaki değeri, w_{ij}^2 bu katmanın ağırlık matrisini, n önceki katmandan gelen nöronların sayısını, m bu katmanın nöron sayısını yani son katman olduğu için sınıf sayısını, Z^2 katmandaki nöronları, Z^3 normalizasyon yapılmış katmanın nöronlarını, y_j çıktı matrisinin sınıfını temsil eder. Softmax fonksiyonundan çıkan bütün sonuçlar pozitif değerdir. Bu fonksiyon çıktı sınıfındaki bütün elemanların toplam değerinin “1” olduğundan emin olur. Bu fonksiyon daha istirarlı tahminler yapılmasını sağlar. Örnek verirken 4 elamanlı $Z = [1 \ 1.5 \ 2.2 \ 3]$ çıktısı olsun. Sınıflandırma işleminden sonra çıktı değerleri denklem 2.10' daki gibi hesaplanır.

$$y_j = Softmax\left(\frac{e^{z_j}}{\sum_{k=1}^k e^{z_k}}\right) \Rightarrow \quad (2.10)$$

$$y_1 = Softmax\left(\frac{e^{z_1}}{\sum_{k=1}^k e^{z_k}}\right), y_2 = Softmax\left(\frac{e^{z_2}}{\sum_{k=1}^k e^{z_k}}\right),$$

$$y_3 = Softmax\left(\frac{e^{z_3}}{\sum_{k=1}^k e^{z_k}}\right), y_4 = Softmax\left(\frac{e^{z_4}}{\sum_{k=1}^k e^{z_k}}\right) \Rightarrow$$

$$y_1 = \frac{e^1}{e^1 + e^{1.5} + e^{2.2} + e^3}, y_2 = \frac{e^{1.5}}{e^1 + e^{1.5} + e^{2.2} + e^3},$$

$$y_3 = \frac{e^{2.2}}{e^1 + e^{1.5} + e^{2.2} + e^3}, y_4 = \frac{e^3}{e^1 + e^{1.5} + e^{2.2} + e^3} \Rightarrow$$

$$y_1 = \frac{2.718}{2.718 + 4.482 + 9.025 + 20.086}, y_2 = \frac{4.482}{2.718 + 4.482 + 9.025 + 20.086},$$

$$y_3 = \frac{9.025}{2.718 + 4.482 + 9.025 + 20.086}, y_4 = \frac{20.08}{2.718 + 4.482 + 9.025 + 20.086} \Rightarrow$$

$$y_1 = \frac{2.718}{36.305}, y_2 = \frac{4.482}{36.305}, y_3 = \frac{9.025}{36.305}, y_4 = \frac{20.08}{36.305} \Rightarrow$$

$$y_1 \approx 0.075, y_2 \approx 0.123, y_3 \approx 0.249, y_4 \approx 0.553$$

Çıktı matrisi $Y = [0.075 \ 0.123 \ 0.249 \ 0.553]$ olur. Softmax fonksiyonu çıktı matrisinin değerlerin toplamının “1” olduğundan emin olur. Bu işlemde çıktı

matrisindeki en yüksek skor modelin tahmin ettiği sınıfı temsil eder. Bir sonraki adım kayıpların belirlenerek bu kayıpların radyanlarının hesaplanması adımıdır. Bir başka deyişle tahmin edilen veriler, veri setindeki gerçek sınıf etiketleri ile karşılaştırılır, türev fonksiyonları ile kayıpların eğimleri elde edilerek yeni ağırlıkların güncellenmesi için algoritma geriye doğru yayılım yaptırılır.

2.3. Geri Yayılım Algoritması

Geri yayılım algoritması sinir ağlarının eğitilmesinde kullanılan en önemli algoritmalarından biridir. Geri yayılım algoritması ilk olarak 1974 yılında P.Werboos[36] tarafından bir doktora tezi ile tanıtılmış, 1986 yılında da sinir ağlarının eğitilmesinde etkili olduğu gösterilmiştir[37].

Geri yayılım algoritması ileri ve geri yayılım olmak üzere iki adımdan oluşmaktadır. İleri yayılımda, girdi görüntüleri katman katman işlenerek çıktı değerleri hesaplanır. Evrimsel katmanlarda rastgele başlatılmış ağırlıklar ile girdi görüntüleri evrimleştirilir, evrimleştirmelerden elde edilen özellik haritalarının değerleri aktivasyon işleminden geçirilir, havuzlama işleminden sonra çok boyutlu matris verileri düzleştirilerek tam bağlantılı katmana iletilir, bu katmanda sınıflandırmalar yapılarak modelin tahminleri üretmesi sağlanır. Bu işlemler ileri yayılım olarak adlandırılır.

İleri yayılım işlemlerinden sonra model yeni tahminler üretir. Bu tahminler gerçek değerlerle ile karşılaştırılır. Kayıp hesaplama fonksiyonları ile kayıplar hesaplanır.

Eğitim algoritmasının ana amacı kayıpları en aza indirmektir. Dolayısıyla eğitim sırasında kayıpları en aza indirgeyen ağırlıkların ve sapmaların hesaplanarak ağırlıkların ve sapmaların yeni güncellenmiş şekli ile tekrar eğitim yaptırmak gerekir. Öncelikle kayıpların ağırlıklara göre türevlerini hesaplamak gerekir. Amacımız, fonksiyonların türevlerini hesaplayarak kayıpları minimize etmektir. Bu işlem için gradyan iniş algoritmasından faydalanabiliriz. Denklem 2.11' de tanımlanan formülde kayıpların ağırlıklara göre gradyanı, denklem 2.11a' da kayıpların sapmalara göre gradyanı hesaplanır ve hesaplanan yeni ağırlıklar ve sapmalar güncellenir.

$$w_{ij}^* = w_{ij} - \eta * \frac{\partial L}{\partial w_{ij}} \quad (2.11)$$

$$b_i^* = b_i - \eta * \frac{\partial L}{\partial b_i} \quad (2.11a)$$

Ancak, kayıpların ağırlıklara ve sapmalara göre türevlerini almak bir dizi hesaplama gerektirecektir. $\frac{\partial L}{\partial w_{ij}}$ hesaplamasını yapabilmek için calculusun zincir kuralından faydalanırız. Denklem 2.12’ de softmax ve katagorik çapraz entropi kayıp fonksiyonları tanımlanmıştır.

$$y_j = \left(\frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \right), L = - \sum_{i=1}^C x_i \log(y_i) \quad (2.12)$$

Burada, y_j çıkış değerlerini sınıflaran softmax fonksiyonu, L ise kayıpları hesaplayan kategorik çapraz entropi fonksiyonudur. Geri yayılımda kaybın y ’ ye göre gradyanını bulabilmek için denklem 2.13’ deki formül ve softmax fonksiyonun z ’ ye göre gradyanını bulabilmek için de denklem 2.13a’ daki formül kullanılır.

$$\frac{\partial L}{\partial y_i} = - \frac{x_i}{y_i} \quad (2.13)$$

$$\frac{\partial y_i}{\partial z_j} = y_i (\delta_{ij} - y_i) \quad (2.13a)$$

Burada, δ_{ij} kronecker deltasıdır ve $i=j$ olduğunda sonuç 1, aksi takdirde sonuç “0” değerini alır. $\frac{\partial L}{\partial y_i}$ ve $\frac{\partial y_i}{\partial z_j}$ türevleri hesapladıktan sonra bu iki eğim birleştirilerek kaybın girdi değerlerine(z) göre eğimini hesaplamak için denklem 2.14’ deki formül kullanılır.

$$\frac{\partial L}{\partial z_i} = \sum_{j=1}^N \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial z_i} = \sum_{j=1}^N \left(- \frac{x_i}{y_i} \right) * y_i (\delta_{ij} - y_i) => \frac{\partial L}{\partial z_i} = (y_i - x_i) \quad (2.14)$$

Kaybın girdi değerlerine(z) göre eğimini hesapladıktan sonra girdi değerlerinin(z) ağırlıklara göre gradyanı hesaplanır. Denklem 2.15’ de z fonksiyonun hesaplama formülü, denklem 2.16’da ise girdi değerlerinin ağırlıklara göre gradyanını hesaplayan formül tanımlanmıştır.

$$z_i = w_{ij} x_j + b_j \quad (2.15)$$

$$\frac{\partial z_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} (w_{ij} x_j + b_j) = \frac{\partial}{\partial w_{ij}} (w_{ij} x_j) + \frac{\partial}{\partial w_{ij}} (b_j) = \frac{\partial z_i}{\partial w_{ij}} = x_j \quad (2.16)$$

Burada, z_i tamamen bağlı katmandaki nöronların hesaplanmasında kullanılan fonksiyon, w_{ij} ağırlık matrisini, x_j bir önceki katmanın düğümlerini, b_j sapma değerlerini temsil eder. Girdi verilerinin ağırlıklara göre gradyanı doğrusal olduğu için girdi verisini(x_j) verir. Son adımda kaybın ağırlıklara göre türevini hesaplamak için calculusun zincir kuralından faydalanılır. Denklem 2.17’ de kaybın ağırlıklara göre gradyanın calculusun zincir kuralına göre hesaplanmasını gösterir. Calculusun

zincir kuralı uygulandığında çıkan formülde hesaplanan diğer gradyanlar yerine konarak kaybın ağırlıklara göre gradyan fonksiyonu elde edilmiş olur.

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}} = (y_i - x_i)x_j \quad (2.17)$$

Burada y_i tahmin edilen değer, x_i gerçek değer, x_j bir önceki katmanın nöronlarını temsil eder. Tamamen bağlı katmanların son katmanında sınıflandırma işlemi softmax fonksiyonu ile gerçekleştirilir. Bu işlem aslında bir normalizasyon işlemidir. Çıkan bütün sınıf skorlarının değerlerinin toplamı “1” e eşittir. Bütün çıkan değerler bir sınıfı temsil eder. En yüksek skor modelin tahmin ettiği sınıftır. Daha sonra bu skorlar one-hot kodlu gerçek sınıf değerleri ile karşılaştırılarak modelin sınıfı doğru tahmin edip etmediği tespit edilir. Bu işlemlerin ardından bütün örneklerin kayıpları hesaplanarak ortalama bir kayıp tespit edilir. Kayıplar tespit edildikten sonra, kayıpları bir sonraki döngüde en aza indirmek için bütün katmanlarda kayıpların ağırlıklara göre gradyanının $(\frac{\partial L}{\partial w_{ij}})$ hesaplanması gerekecektir. Geri yayılım ile geriye doğru bütün katmalardaki ağırlıklar kayıpların ağırlıklara göre gradyanı hesaplanarak yeniden güncellenir. Bu hesaplama gerçekten karmaşık hesaplamalar gerektirecektir. Son katmandaki bu işlemleri bu bölümde anlatmaya çalıştık. EK.A.1’ de son katmandaki gradyanların hesaplanması ayrıntılı olarak örnekler verilerek gösterilmiştir.

Bu işlemler geriye doğru bütün katmanlarda uygulanır. Bütün katmanlarda da ağırlık ve sapmaların gradyanlarını hesaplamak için calculusun zincir kuralından faydalanılır. Son katmandan başlayıp ilk katmana doğru yapılan bu işlem geri yayılım olarak adlandırılır. Hesaplanan gradyanlar aslında kayıpların hangi yönde ve ne kadar değiştiğini gösteren türevlerdir. Gradyan hesaplamaların asıl amacı modelin doğruluğunu artırmak ve kayıpları en aza indirmektir. Eğitimin her aşamasında bu gradyanlar hesaplanır. Modelin parametre sayısına bağlı olarak bu hesaplama süreci de değişir. Kompleks yapılarda hem parametre sayısının fazlalığı hem de yapılan işlemlerin karmaşıklığı eğitimi ciddi oranda yavaşlatır.

2.4. Optimizasyon Algoritmaları

Evrişim sinir ağlarında optimizasyon algoritmaları, modelin eğitiminde kayıpları en aza indirmek için ağ içerisindeki ağırlık ve sapma parametrelerini ayarlama önemli bir role sahiptir. Hatayı en aza indirmek için eğitim sırasında

ağıdaki ağırlık ve sapma parametreleri daima güncellenir. Ağırlıklara göre gradyanları hesaplanan kayıpların büyüklüğü öğrenme parametresi ile ayarlanır ve geri yayılım algoritması ile yeni ağırlıklar güncellenir. Bu işlemler modelin performansı artıncaya kadar ve kayıplar en aza indirgene kadar devam eder. Öğrenme oranı önemli bir parametredir ve makul bir oranda seçilirse öğrenme süreci kötü şekilde etkilenmez. Öğrenme oranının çok büyük seçilmesi eğitim süresini kısaltır ancak ağırlık yakınsamasını engelleyebilir ve kayıpların artmasına neden olabilir. Öğrenme oranının çok küçük seçilmesi de eğitim sürecini uzatacaktır ancak ağırlık yakınsamasına ve kayıpların oldukça azalmasına katkıda bulunacaktır. Düşük öğrenme oranı modelin optimum doğruluk oranına ulaşmasını sağlar.

Modelin eğitiminde doğru optimizasyon algoritmalarının belirlenmesi, hem eğitimin hızını hem de modelin performansını olumlu yönde etkiler. Doğru optimizasyon algoritmaları kayıpları hızlı bir şekilde en aza indirerek eğitim sürecini önemli oranda azaltır ve modelin genelleme yeteneğini geliştirir.

2.4.1. Toplu Gradyan İnişi

Toplu gradyan inişi algoritması, temel optimizasyon algoritmalarından biri olarak görülür[37, 38]. Her bir eğitim döngüsünde tüm eğitim örnekleri için, model ağırlıklarına göre tüm kayıpların eğimi hesaplanır ve bütün ağırlıklar güncellenir. Denklem 2.18' de toplu gradyan inişi formülü tanımlanmaktadır.

$$W^* = W - \eta * \frac{\partial L}{\partial W} \quad (2.18)$$

Burada W^* yeni ağırlık matrisini, W önceki ağırlık matrisini, η öğrenme katsayısını, $\frac{\partial L}{\partial W}$ ise bütün veri setindeki kaybın ağırlıklara göre eğimini temsil eder.

Toplu gradyan iniş algoritması, gradyanları hesaplamak için tüm veri setini kullanır. Dolayısıyla, ağırlık güncellemeleri daha düzgündür, gradyan hesaplamalarını daha doğru saptayarak daha doğru ağırlık parametrelerini belirler. Ancak büyük veri setlerinde tüm veri setinin gradyanlarını hesaplamak hem hesaplama maliyetini artırır hem de daha büyük hafıza gerektirir. Eğitim sürecini yavaşlatabilir.

2.4.2. Stokastik Gradyan İnişi

Stokastik optimizasyon algoritması veri setinin tamamını kullanmaz, her bir eğitim döngüsünde sadece bir tane eğitim örneği kullanır [39]. Eğitim örneği veri setinden rastgele seçilerek eğitim verileri arasındaki ilişkinin azaltılmasına yardımcı

olur. Eğitim sürecinde rastgele bir örnek kullanılması, büyük veri setlerinde gradyanların hesaplanmasını kolaylaştırır, daha az hafıza gerektirir, ağın yakınsamasını hızlandırır. Ancak, ağırlıkların güncellenmesinde iniş çıkışlar olabilir ve buna bağlı olarak çok daha fazla eğitim döngüsü gerektirebilir. Öğrenme katsayısına duyarlıdır. Öğrenme oranının çok yüksek seçilmesi modelin optimum sonuca ulaşmasını kaçırabilir. Öğrenme oranının çok düşük seçilmesi de ağın yakınsamasını oldukça yavaşlatır. Dolayısıyla öğrenme oranının optimum bir değer seçilmesi gerekir. Stokastik optimizasyon formülü denklem 2.19' da tanımlanmıştır.

$$W^* = W - \eta * \frac{\partial L_i}{\partial W} \quad (2.19)$$

Burada W^* yeni ağırlık matrisini, W önceki ağırlık matrisini, η öğrenme katsayısını, $\frac{\partial L_i}{\partial W}$ ise tek bir eğitim örneğindeki kaybın ağırlıklara göre eğimini temsil eder.

2.4.3. Küçük Gruplu Gradyan İniş

Bu yaklaşımda eğitim örnekleri küçük gruplara bölünür[40-42]. Toplu gradyan iniş ve stokastik gradyan iniş algoritması ile benzerdir. Toplu gradyan iniş algoritmasında tüm veri setinin kayıpların eğimi hesaplanır, stokastik algoritmada sadece bir örneğin kayıp eğimi hesaplanır. Küçük toplu gradyan iniş algoritmasında ise küçük gruplara bölünen her bir veri setinin kayıp gradyanları ayrı ayrı hesaplanır. Bu optimizasyon algoritmasında ağırlık güncellemeleri her küçük eğitim grubu için yapılır. Ağırlıkların daha sık güncellenmesi ağın daha hızlı ve daha istikrarlı yakınsamasına yardımcı olur.

Ancak optimum öğrenme oranını belirlemek zordur. Öğrenme oranının çok küçük seçilmesi ağın yakınsamasının yavaşlamasına, öğrenme oranının çok büyük belirlenmesi de kayıpların dalgalanmasına sebep olabilir.

2.4.4. Momentum

Gradyan iniş optimizasyon algoritmalarında momentum ağın yakınsamasını, eğitim sürecini hızlandırmak ve modelin doğruluğunu artırmak için kullanılan bir tekniktir [43, 44]. Kayıpların en aza indirgeme sürecini hızlandırarak optimizasyon algoritmasına yardımcı olur. Bu algoritmada, geçmiş eğitim döngüsünün gradyanlarını kayıt eden bir hız vektörü vardır. Her döngüde bu vektör, momentum katsayısı ve gradyanlara göre güncellenir. Hız vektörünün hesaplanması denklem 2.20' de tanımlanmıştır. Bu vektör yeni ağırlıkların elde edilmesinde kullanılır.

Momentumla elde edilen gradyan iniş algoritması denklem 2.21' deki gibi tanımlanır.

$$m_t = \beta * m_{t-1} + \left(\eta * \frac{\partial L}{\partial W_t} \right) \quad (2.20)$$

$$W_{t+1} = W_t - m_t \quad (2.21)$$

Burada m_t t döngüsündeki hız vektörünü, β momentum katsayısını, η öğrenme oranını, $\frac{\partial L}{\partial w_{ij}}$ ise kaybın ağırlığa göre gradyanını, W ise ağırlıkları temsil eder. Momentum terimi(β) 0 ile 1 arasında değer alır.

2.4.5. RMSprop

RMSprop[38], sinir ağlarında yaygın olarak kullanılır. Son güncellenen gradyanın büyüklüğüne göre her parametrenin öğrenme oranını ayarlamak için kullanılan bir optimizasyon algoritmasıdır. Momentum hesaplaması denklem 2.22' de, RMSprop algoritması denklem 2.23' de tanımlanmıştır.

$$m_t = \beta * m_{t-1} + (1 - \beta) * \left(\frac{\partial L}{\partial W_t} \right)^2 \quad (2.22)$$

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{m_t - \epsilon}} * \frac{\partial L}{\partial W_t} \quad (2.23)$$

Burada m_t hız vektörünü, β momentum katsayısını, η öğrenme oranını, ϵ sıfır ile bölünmesini önlemek için küçük bir katsayıyı temsil etmektedir. RMSprop optimizasyon algoritması her bir parametre için öğrenme oranını dinamik olarak ayarlayarak ve hesaplanan gradyanın büyüklüğünü hesaba katarak ağı daha hızlı yakınsamasına ve performansının artmasına katkı sağlar.

2.4.6. Adam

Adam[44, 45], sinir ağlarında yaygın olarak kullanılan bir optimizasyon algoritmasıdır. RMSprop ve momentum algoritmalarından ilham alınarak geliştirilen bu teknik, ağı daha hızlı yakınsamasını ve daha iyi genelleme yapabilmesini sağlar. Adam tekniği, gradyanların birinci ve ikinci momentumlarını hesaplar, bunları gradyanları güncellemek ve öğrenme oranını ayarlamak için kullanır. Gradyanların birinci momentumun hesaplanması denklem 2.24' de, ikinci momentumun hesaplanması denklem 2.24a' da tanımlanmıştır.

$$m_t^1 = \beta_1 * m_{t-1}^1 + (1 - \beta_1) * \frac{\partial L}{\partial W_t} \quad (2.24)$$

$$m_t^2 = \beta_2 * m_{t-1}^2 + (1 - \beta_2) * \left(\frac{\partial L}{\partial W_t} \right)^2 \quad (2.24a)$$

Burada, m_t^1 ve m_t^2 gradyanların birinci ve ikinci momentumunu, β_1 ve β_2 birinci ve ikinci momentum için bozulma oranını ve $\frac{\partial L}{\partial W_t}$ ise kaybın ağırlıklara göre gradyanını temsil eder. Daha sonra bu momentum değerlerinin sapması ayarlanır. Denklem 2.25' de birinci ve ikinci momentum değerlerinin sapma değerleri ayarlanması tanımlanmaktadır.

$$m_t^3 = \frac{m_t^1}{1-\beta_1^t}, \quad m_t^4 = \frac{m_t^2}{1-\beta_2^t} \quad (2.25)$$

Sapma değerlerinin ayarlanmasından sonra ağırlıkları güncellemek için Adam optimizasyon algoritması denklem 2.26' da tanımlanmaktadır.

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{m_t^4 + \epsilon}} * m_t^3 \quad (2.26)$$

Burada W_{t+1} güncellenen yeni ağırlıkları, W_t mevcut ağırlıkları, η öğrenme oranını, m_t^4 sapma değeri ayarlanmış ikinci momentum değerini, m_t^3 sapma değeri ayarlanmış birinci momentum değerini, ϵ ise sıfıra bölümden kaçınmak için çok küçük bir katsayıyı temsil eder.

Adam, gradyanların yönüne ve büyüklüğüne göre öğrenme oranını ayarlayarak ağırlıkların yakınsamasını hızlandırır, ağırlıkların genelleme yapma yeteneğini geliştirir. Sapma değerlerini ayarlayarak gradyanların hareketli ortalamalarının doğruluğunu iyileştirir ve gradyanların hareketli ortalamalarının sıfıra doğru sapması önlenir. Adam, belleği daha verimli kullanır ve daha az hesaplama maliyetine sahiptir.

2.4.7. Diğer Optimizasyon Algoritmaları

Kayıpları en aza indirmek ve modelin performansını daha fazla iyileştirmek için Adagrad[46], Adadelat[47], Adamax[45] ve Nadam[48] gibi daha çok sayıda optimizasyon algoritması geliştirilmiştir. Bu algoritmalar öğrenme oranını daha iyi ayarlayarak kayıpları en aza indirecek şekilde ağırlık ve sapma parametrelerini güncelleyip ağırlıkların daha hızlı yakınsamasına ve ağırlıkların performansının iyileştirilmesine önemli oranda katkı sağlamaktadır.

2.5. Aktivasyon Fonksiyonları

Aktivasyon katmanı evrişim katmanından gelen çıktılar normalleştirilerek ağırlıkların esnekliği artırılır ve modelin öğrenme yeteneğinin geliştirilmesine yardımcı olur. Bu fonksiyonların türevlenebilir olması sayesinde CNN verileri içerisindeki

karmaşık ve hiyerarşik yapıları, özellikleri ve ilişkileri öğrenir. Fonksiyonların doğrusal olmama durumu sinir ağına tanıtılarak modelin eğitim sürecine yardımcı olur [49, 50]. Önceki katmandan gelen nöronlar fonksiyonun belirlediği aralığa sıkıştırılarak hem kaybolan gradyan hem de patlayan gradyan probleminin önüne geçilmiş olur. Ayrıca verilerin zikzak oluşturmada daha düzenli olması sağlanır. Nöronların normalize edilerek girdi görüntülerin daha düzenli sınıflandırılmasına imkân verir. Bazı aktivasyon fonksiyonları modelin aşırı öğrenmesinin önüne geçerek modelin performansının artmasına yardımcı olabilmektedir.

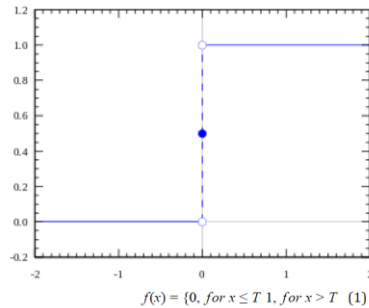
Bir CNN mimarisinde çok sayıda aktivasyon fonksiyonu bulunması, modelin eğitim sürecindeki hesaplama maliyeti, üzerinde durulması gereken konulardan biridir. CNN mimarisi tasarlanırken, modelin performansından ödün vermeden eğitim sürecini hızlandıran hesaplama maliyeti açısından daha verimli olan aktivasyon fonksiyonlarının kullanılmasına özen gösterilmelidir. CNN’de, ayrık veya basit türevli fonksiyonların kullanılması daha yaygındır [26].

2.5.1. Doğrusal ve Adım Fonksiyonları

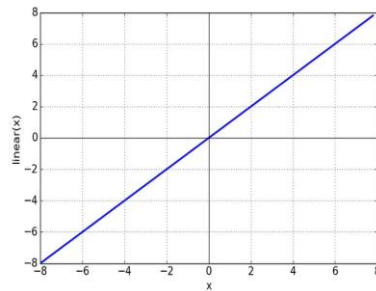
Adım fonksiyonunda çıktı verileri “0” ve “1” gibi ikili verilere dönüştürülür. Bu fonksiyonda girdi verileri belirlenen bir eşik değerinde buluşur. Denklem 2.27’ de adım fonksiyonu tanımlanmıştır. Fonksiyon, eşik değerine eşit veya eşik değerinden fazla olan girdi verilerini “1”, eşik değerinden küçük olan girdi verilerini “0” değerine dönüştürür.

$$f(x) = \begin{cases} 1, & \text{eğer } x \geq \text{eşik değeri} \\ 0, & \text{eğer } x < \text{eşik değeri} \end{cases} \quad (2.27)$$

Burada x değişkeni girdi verisidir. Fonksiyonda, en sık kullanılan eşik değeri sıfırdır. Bu fonksiyon doğrusal olmama halinin en basit halidir. Modele doğrusal olmama durumunu tanıtır. İkili sınıflandırmalarda kullanılması daha uygundur.



(a) Adım fonksiyonu



(b) Doğrusal fonksiyonu

Şekil 2.4 : Adım ve Doğrusal Fonksiyon

Adım fonksiyonun türevlenebilir olmama durumu, gradyan tabanlı algoritmalarda kullanılmasına uygun olmaz[51]. Veriler çoğu yerde sıfır olduğundan kaybolan gradyan problemi ile de karşı karşıya kalır. Bu da eğitim sırasında ağırlıkların güncellenmesini zorlaştırır. Kısaca fonksiyonun hem türevlenebilir olmaması hem de kaybolan gradyanlar problemini barındırması modelin zayıf performans göstermesine neden olur. Şekil 2.4a' da adım fonksiyonu görülmektedir.

Doğrusal fonksiyonlar, girdi verilerini orantılı olarak dönüştürmek için kullanılan matematiksel işlemlerdir [50, 52]. Girdi verileri sabit bir katsayı ile çarpılarak çıktı elde edilir. Çıktı, girdi verisi ile doğru orantılıdır. Denklem 2.28' de doğrusal bir fonksiyonun formülü görülmektedir.

$$f(x) = cx \quad (2.28)$$

Adım fonksiyonu sıfır gradyanlar problemini barındırması ve iniş eğiminin mümkün olmaması sebebi ile geri yayılım algoritmalarında eğitim sırasında ağırlıklar güncellenemez. Doğrusal bir fonksiyon adım fonksiyonun yerine kullanıldığında, birçok farklı nöron aynı anda etkinleştirilebilir. Eğitim sırasında hesaplama maliyeti ucuz olduğu için daha çok gizli katmanlarda kullanılır. Bu fonksiyon modele doğrusal olmama durumunu öğretemediğinden karmaşık problemleri öğrenmede sınırlıdır. CNN mimarisindeki katman sayısını göz ardı ederek ağırlıkların tek bir katmandan oluşmuş gibi davranmasını sağlar. Dolayısıyla, doğrusal olma durumu nedeni ile çok katmanlı bir ağda kullanımı uygun değildir. Şekil 2.4b' de doğrusal bir fonksiyon gösterilmektedir. Eğitim sırasında gradyan inişiyle ilgili sorunlar bu fonksiyonda da ortaya çıkar. Her durumda sabit bir türeve sahiptir.

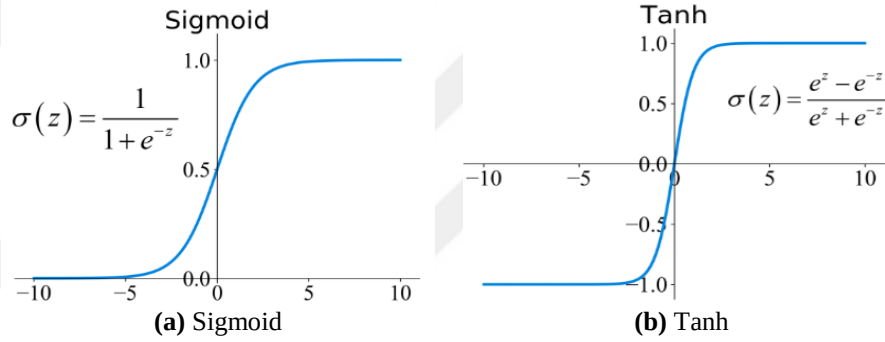
2.5.2. Sigmoid, Tanh, ArcTan, Softsign ve Softmax Fonksiyonları

Bu tür fonksiyonlar doğrusal olmayan aktivasyon fonksiyonları kategorisinde yer alır ve modern sinir ağlarında sıklıkla kullanılmaktadır. Modelin karmaşıklık seviyesini artırarak veri setleri üzerindeki karmaşık yapıları ve bilgileri daha iyi bir şekilde öğrenmesini sağlar. Nöronların çıktılarını $[-1,1]$, $[0,1]$ gibi belirli değerler arasına sıkıştırır. CNN mimarisinde evrişim ve tamamen bağlantılı katmanlardan sonra kullanılır. Bu tür fonksiyonlar türevlenebilir olduğundan gradyan tabanlı algoritmalarda kullanılması elverişlidir [1].

Sigmoid fonksiyonu sinir ağlarında yaygın olarak kullanılan bir fonksiyon olup girdi verilerini $[0,1]$ aralığına dönüştürür. Denklem 2.29’ da sigmoid fonksiyonun formülü tanımlanmıştır.

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.29)$$

Sigmoid fonksiyonu türevlenebilir olmasından dolayı gradyan tabanlı algoritmalarda kullanımı yaygındır. Girdi verilerinin geniş bir aralığa yayılması ağın öğrenmesini kolaylaştırır. Gradyanlar sıfır olmayıp düzdür ancak gradyan çok küçük olduğunda doygunluk(saturation) probleminden dolayı sıkıntı çeker. Çünkü ağırlıklar, gradyan büyüklüğüne orantılı olarak öğrenme sürecini yavaşlatır, küçük gradyanların çok sayıda olması ağı eğitilmesini zorlaştırır [1, 50, 51, 53, 54]. Şekil 2.5’ de sigmoid ve tanh fonksiyonları görülmektedir.



Şekil 2.5 : Aktivasyon fonksiyonları: (a) Sigmoid, (b) Tanh

Tanh fonksiyonu sigmoid fonksiyonuna benzer şekilde girdi verilerini çıkışı $[-1,1]$ aralığına dönüştürür. Sigmoid fonksiyonundan daha güçlü eğimlere sahiptir. Ağırlıkların güncellemesinin çok daha büyük olmasını sağlar ancak bu fonksiyon kaybolan(vanishing) gradyanlar problemine neden olur. Özellikle derin öğrenme ağlarında yavaş yakınsamaya sahiptir. Çok katmanlı sinir ağlarının eğitim süresi boyunca sigmoid fonksiyonundan daha iyi çalışır. Her ne kadar tanh ve sigmoid fonksiyonların her biri yaklaşım teorisi şartlarını yerine getirirse de, gizli katmanlarda performans zayıftır ve her iki fonksiyonda da doygunluk sorunlarına neden olur.

Arctan fonksiyonu da tanh fonksiyonuna benzer bir fonksiyon olup giriş değerlerini $[-\frac{\pi}{2}, \frac{\pi}{2}]$ aralığına dönüştürür. Bu fonksiyonda tanh fonksiyonun karşılaştığı sorunlara sahiptir.

Sinir ağlarında kullanılan başka bir fonksiyon olan softsign fonksiyonu da sigmoid ve tanh fonksiyonuna benzer şekilde giriş verilerini $[-1,1]$ aralığına

dönüştürür. Üstel olarak büyüyen tanh fonksiyonun aksine bu fonksiyon polinomial yakınsağa sahiptir. Türevlenebilir bir özelliğe sahiptir. Büyük girdi verilerini sigmoid fonksiyonundan daha iyi yakaladığından dolayı hızlı bir şekilde doyuma ulaşmaz ve modelin daha iyi öğrenebilmesine yardımcı olur. Ancak, hem hesaplama maliyeti daha fazladır hem de kaybolan gradyanlar sorunu ile mücadele etmektedir [51]. Doğrusal olmayan bu fonksiyonlar, kaybolan eğim ve doyumluk problemleri ile mücadele etmektedir ve hesaplama karmaşıklığından dolayı yüksek hesaplama maliyetine sahiptir.

Sinir ağlarında çıktı katmanında özellikle çoklu sınıflandırma problemlerinde yaygın olarak kullanılan bir diğer aktivasyon fonksiyonu softmax fonksiyonudur[55, 56]. Bu fonksiyon ile girdi verileri sınıf skorlarına dönüştürülür. Bu fonksiyon bütün çıktı değerlerinin toplamının “1” olduğundan emin olur. Denklem 2.30’ da softmax fonksiyonu tanımlanmıştır.

$$\sigma(z_i) = \text{Softmax}\left(\frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}\right) \quad (2.30)$$

Burada z_i girdi verisini, $\sigma(z_i)$ ise aktive edilmiş veriyi temsil eder.

2.5.3. ReLU, Leakly ReLU, PReLU ve RLReLU

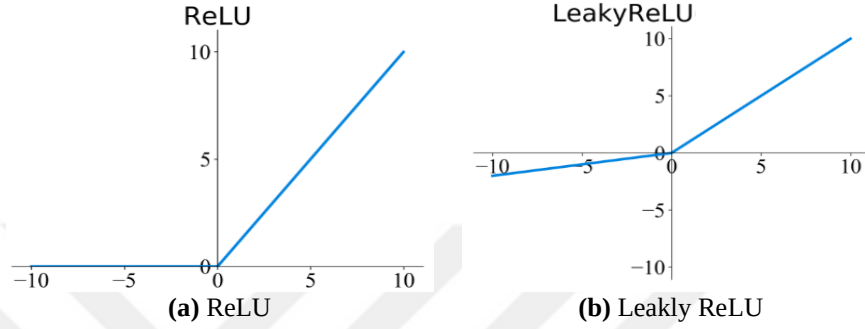
Bu tür fonksiyonlar derin öğrenme ağlarında yaygın olarak kullanılan aktivasyon fonksiyonlarıdır. Sigmoid ve tanh fonksiyonlarında karşılaşılan problemlerin üstesinden gelmek için geliştirilmiştir.

Düzeltilmiş doğrusal üniteler ilk olarak kısıtlı Boltzmann makinelerinde kullanılmıştır [57]. Daha sonra ReLU, Glorot v.d. tarafından 2011’de tanıtılmış ve özellikle evrimsel sinir ağı algoritmalarında yaygın olarak kullanılan geçerli aktivasyon fonksiyonu haline gelmiştir [58]. ReLU fonksiyonu denklem 2.31’ deki gibi tanımlanmaktadır.

$$f(x) = \max(0, x) \quad (2.31)$$

ReLU fonksiyonu, sigmoid ve tanh gibi aktivasyon fonksiyonlarıyla karşılaştırıldığında basit, hesaplama açısından ucuz ve hızlı yakınsaktır. Hesaplama maliyeti daha ucuz olmasından dolayı modelin eğitim sürecini kısaltır ve yaygın olarak evrim katmanlarında kullanılmaktadır. Pozitif değerler türevlenebilir ve gradyan hesaplamalarını basitleştirir. ReLU fonksiyonunun doğrusal olmama

durumu sayesinde, ağıın karmaşık yapılarını öğrenmesini sağlar. ReLU, tüm negatif değerleri sıfıra eşlediği için seyrek fonksiyon olarak değerlendirilir ve modelin aşırı öğrenmesine engel olur. Doygunluk ve kaybolan gradyan problemlerinin üstesinden gelebilmektedir. Ağıın içerisinde yer alan nöron değerleri sürekli olarak sıfır değeri ile eşleştirildiğinde nöronların pasif kalmasına ve bunun sonucunda ReLU' nun ölme(dying) problemi ile karşılaşılır. Bu problemin çözümü için çeşitli yaklaşımlar önerilmiştir. Şekil 2.6' da ReLU ve sızdıran ReLU fonksiyonu görülmektedir.



Şekil 2.6 : Düzeltilmiş aktivasyon fonksiyonları: (a)ReLU, (b) Leaky ReLU

ReLU fonksiyonunda oluşan ölme probleminin üstesinden gelmek için geliştirilmiş bir diğer düzeltilmiş fonksiyon sızdıran ReLU(Leaky ReLU) aktivasyon fonksiyonudur[59]. Sızdıran ReLU fonksiyonu denklem 2.32' deki gibi tanımlanır.

$$f(x) = \begin{cases} x, & \text{eğer } x > 0 \\ \alpha x, & \text{eğer } x \leq 0 \end{cases} \quad (2.32)$$

Burada x girdi değeri, α ise küçük sabit bir katsayıdır. ReLU fonksiyonunda oluşan ölme probleminin üstesinden gelebilmek için negatif değerler sıfır ile eşleştirmek yerine, αx katsayısı ile negatif değerler küçük negatif değerlere dönüştürülür. Dolayısıyla, negatif değerler daha küçük değerlere dönüştürüldüğünden nöron değerlerinin aktif kalması ve bilgi kayıplarının önüne geçilmesi sağlanmış olur.

ReLU fonksiyonunun güncellenmiş bir başka sürümü de parametrik düzeltilmiş doğrusal birim (PReLU) fonksiyonudur [29]. PReLU fonksiyonunda öğrenilebilir bir parametre ile negatif eğimler kontrol edilir. PReLU denklem 2.33' deki gibi tanımlanır.

$$f(x) = \begin{cases} x, & \text{eğer } x > 0 \\ \alpha x, & \text{eğer } x \leq 0 \end{cases} \quad (2.33)$$

Burada x girdi verisi, α ise öğrenilebilir bir parametre değeridir. α sabit bir katsayı olduğunda fonksiyon sızdıran ReLU, α sıfır olursa ReLU fonksiyonu olur. Negatif değerler öğrenilebilir bir parametre ile kontrol edilerek nöronların pasif hale gelmesi önemli oranda önlenir. Fonksiyonda yer alan öğrenilebilir parametre sayesinde eğitim süreci içerisinde modelin daha iyi performans göstermesi sağlanır. Negatif değerlerden oluşturulan eğimler daha etkili bir şekilde öğrenilerek modelin performansının ve doğruluğunun artmasına yardımcı olunur. Ancak ağı fazla parametreler hesaplama maliyetini artırır, özellikle küçük veri setlerinde modelin aşırı öğrenmesine neden olabilir.

Sinir ağlarında kullanılan bir diğer aktivasyon fonksiyonu, rastgele sızdıran ReLU(RLReLU) fonksiyonudur. Eğitim sırasında, çıktı değerlerindeki negatif değerler tekdüze dağılımdan rastgele seçilen değerler ile kontrol edilir [25]. RLReLU fonksiyonu denklem 2.34' deki gibi tanımlanır.

$$f(x) = \begin{cases} x, & \text{eğer } x > 0 \\ \alpha x, & \text{eğer } x \leq 0 \end{cases} \quad (2.34)$$

Burada, x girdi değeri, α ise eğitim sırasında tekdüze dağılımdan $U(l,u)$ seçilen rastgele bir değerdir.

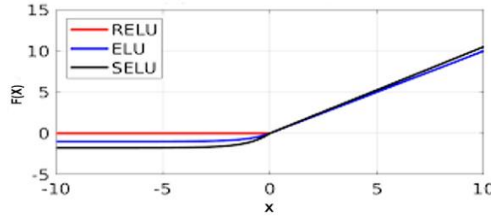
2.5.4. Doğrusal Olmayan Diğer Aktivasyon Fonksiyonları

Evrişim sinir ağlarındaki aktivasyon katmanında kullanılması için 2015 yılında D.Clevert v.d. [60] tarafından eksponansiyel doğrusal bir fonksiyon (ELU) önerilmiştir. ELU eğitim sürecini hızlandırır ve modelin sınıflandırma hataları yapmasını azaltır. ReLU, LReLU, PReLU gibi ELU fonksiyonu da kaybolan gradyan problemini azaltır. Diğer fonksiyonların aksine ELU öğrenme sürecini hızlandırır ve modelin öğrenebilmesinde önemli oranda iyileştirmeler sağlar. ELU, denklem 2.35'deki gibi tanımlanır.

$$ELU(x) = \begin{cases} x, & \text{eğer } x > 0 \\ \alpha(e^x - 1), & \text{eğer } x \leq 0 \end{cases} \quad (2.35)$$

Burada α , ELU' nun negatif değerlerin doygunluğa ulaşmasını kontrol eden pozitif sabit bir parametredir. ELU, küçük gradyanları koruyarak gradyanların kaybolmasını önler. Modelin öğrenme sürecini hızlandırır, doğruluğu artırarak modelin daha iyi performans göstermesine yardımcı olur. Ancak eksponansiyel

hesaplamaları nedeni ile hesaplama maliyetini artırmaktadır. Şekil 2.7’ de ReLU, ELU ve SeLU fonksiyonları görülmektedir.



Şekil 2.7 : ReLU, ELU ve SeLU fonksiyonları

2017 yılında G. Klambauer v.d. [61] tarafından ölçeklendirilmiş eksponansiyel doğrusal fonksiyon (SeLU) önerilmiştir. SeLU fonksiyonu denklem 2.36’ daki gibi tanımlanır.

$$SeLU(x) = \lambda \begin{cases} x, & x > 0 \\ \alpha e^x - \alpha, & x \leq 0 \end{cases} \quad (2.36)$$

Burada α ve λ iki sabit parametredir. Bu fonksiyon, kendi kendini normalize eden sinir ağları için geliştirilmiş, aktivasyon fonksiyonun çıktısını normalleştirmesine gerek kalmadan normalleştirmeyi kendi kendine yapmasını sağlar. Ağın kendi kendine normalize etmesi nedeni ile gradyanların korunmasına, kaybolan gradyan ve ReLU fonksiyonunda yaşanan gradyanların ölmesi problemlerinin azaltılmasına yardımcı olur. Daha fazla işlem gerektirmeden diğer aktivasyon fonksiyonlarına göre daha hızlı ve daha iyi öğrenirler.

ELU fonksiyonuna benzer şekilde, 2017 yılında B.Carlile v.d. [62] tarafından derin sinir ağlarında öğrenmeyi hızlandırmak için ters karekök doğrusal birim fonksiyonunu (ISRLU) önermişlerdir. ISRLU fonksiyonu denklem 2.37’ deki gibi tanımlanır.

$$f(x) = \begin{cases} x, & x \geq 0 \\ \frac{x}{\sqrt{1+\alpha x^2}}, & x < 0 \end{cases} \quad (2.37)$$

Burada α , fonksiyondaki negatif değerleri kontrol eden hiperparametredir. ISRLU, ELU fonksiyonunun avantajları ile hemen hemen aynıdır ancak ELU’ dan daha iyi bir performansa sahiptir.

2018 yılında A.Nicole [63] tarafından tanh ve ReLU fonksiyonun hibrit bir sürümü olan parçalı doğrusal birim(PLU), 2015 yılında H.Zheng v.d.[64] tarafından softplus, 2016 yılında D.Hendrycks v.d. [65] tarafından gauss hata doğrusal birim(GeLU), 2018 yılında S.Elfrwing v.d. [66] tarafından sigmoid ağırlıklı doğrusal

birim(SiLU) ve onun türevi dSiLU, 2017 yılında P. Ramachandran v.d. [67] tarafından kendiliğinden geçmeli(Swish) ve 2020 yılında M. Tanaka[68] tarafından ağırlıklı sigmoid kapı (WiG) fonksiyonları önerilmiştir.

Sonuç olarak evrişimsel sinir ağının başarılı bir şekilde eğitilebilmesi için aktivasyon fonksiyonları önemli bir rol oynar. Sinir ağının her bir katmanında hesaplama maliyeti göz önünde bulundurularak uygun olan aktivasyon fonksiyonunun kullanılması gerekir. Sigmoid gibi doğrusal olmayan fonksiyonlarda yaşanan doygunluk ve kaybolan gradyan problemlerine çözüm getirebilmek, ağın eğitim sürecini hızlandırmak ve performansını artırmak, sınıflandırma hatalarını minimize etmek için çok sayıda fonksiyon geliştirilerek önerilmiştir.

2.6. Kayıp Hesaplama Fonksiyonları

Evrişim sinir ağlarında kayıp hesaplama fonksiyonları önemli bir role sahiptir. Bu fonksiyonların amacı gerçek çıktı ile modelin tahmin ettiği çıktı arasındaki hatayı hesaplamaktır. Hesaplanan bu hatalar kayıp olarak adlandırılır. Hatalar hesaplandıktan sonra ağırlıkların güncellenmesi için çıktı katmanından girdi katmanına doğru geri yayılım yapılır. Geri yayılım sırasında her bir katmandaki gradyanlar calculus zincir kuralından faydalanılarak hesaplanır. Hataları minimize etmek için hesaplanan gradyanlar kullanılarak ağ içerisindeki her bir katmanda bulunan ağırlıklar ve sapmalar SGD ve Adam gibi optimizasyon algoritmalar vasıtası ile güncellenir. Hataların hesaplanarak ağırlıkların güncelleme döngüsü, hatalar en aza indirelene kadar ve modelin iyi bir performans gösterebilmesine kadar devam edebilir.

Sınıflandırma problemlerinde, modelin ikili veya çoklu tahminleri gerçekleştirebilmesi için kayıp hesaplama fonksiyonlarının önemi büyüktür. Sinir ağında kullanılması için seçilmiş olan kayıp hesaplama fonksiyonu modelin performansını önemli oranda etkileyebilir. Bu bölümde evrişimsel sinir ağlarında sıklıkla kullanılan kayıp hesaplama fonksiyonları üzerinde duracağız.

2.6.1. İkili Çapraz – Entropi Fonksiyonu

İkili Çapraz Entropi fonksiyonu (BCE), ikili sınıflandırma problemlerinde yaygın olarak kullanılan bir kayıp hesaplama fonksiyonudur [69]. Bir sınıfın tahmin edilen değeri ile gerçek sınıf değeri arasındaki farkı ölçer [70]. BCE kayıp hesaplama fonksiyonu, denklem 2.38'deki gibi tanımlanır.

$$BCE(x, y) = - [x \log(y) + (1 - x) \log(1 - y)] \quad (2.38)$$

Burada x gerçek değerdir ve “0” veya “1” değerini alır. y ise modelin tahmin ettiği değerdir ve “0” ile “1” arasında değer alır. BCE kayıp hesaplama fonksiyonu gerçek değer ne kadar iyi tahmin edilebildiğini değerlendirir. Gerçek değerden uzaklaşan tahminler dezavantajlı duruma düşürülüp bir ceza puanı ile cezalandırılır. Burada, gerçek etiket x=1 olduğu zaman kayıp $-\log(y)$ olur ve düşük tahmin edilen değer cezalandırılır. Gerçek etiket x=0 olduğu zaman kayıp $-\log(1-y)$ olur ve yüksek tahmin edilen değer cezalandırılır. Toplu olarak kaybı hesaplamak için denklem 2.39’ daki formül kullanılır.

$$BCE_{Toplu}(x, y) = - \frac{1}{N} \sum_{i=1}^N [x_i \log(y_i) + (1 - x_i) \log(1 - y_i)] \quad (2.39)$$

Burada x gerçek değeri, y ise tahmin edilen değeri temsil eder. İkili çapraz entropi, tahmin edilen olasılıkların her birini 0 veya 1 olabilen gerçek sınıf çıktısıyla karşılaştırır. Daha sonra, beklenen değerden uzaklığa bağlı olarak olasılıkları cezalandıran puanı hesaplar. Bu, gerçek değere ne kadar yakın veya uzak olduğu anlamına gelir.

İkili çapraz entropi kayıp fonksiyonu, ikili sınıflandırmalarda eğitim sürecinde optimizasyon süreci içerisinde yer alarak modelin performansının artırılmasında kritik rol oynamaktadır. Hesaplanması kolay ve türevlenebilirdir. Modelin çıktısının olasılıksal olarak yorumlanmasını sağlar. Diğer kayıp fonksiyonlarına göre, aykırı değerlere duyarlıdır. Ancak, bir sınıfın örnek sayısı diğerlerinden önemli ölçüde fazla olduğunda, oluşan sınıf dengesizliği problemine duyarlıdır. Bu gibi durumlarda, ağırlıklı ikili çapraz entropi fonksiyonunu kullanabiliriz.

2.6.2. Ağırlıklandırılmış İkili Çapraz-Entropi Fonksiyonu

Ağırlıklandırılmış ikili çapraz entropi fonksiyonu(WBCE), ikili sınıflandırma problemlerinde yer alan sınıf dengesizliğin olduğu durumlarda kullanılması faydalı olacaktır. Sınıflardaki örnekler düzenli bir şekilde dağılım göstermediğinde yani dengesiz olduğunda standart BCE, kayıpların hesaplanmasında istenen performansı gösteremeyebilir. Çünkü model ana sınıfa doğru sapabilir. Bu gibi durumlarda, WBCE kayıp hesaplama fonksiyonun kullanılması hem pozitif hem de negatif sınıfa farklı önem vermesi sağlanarak modelin daha iyi öğrenebilmesine yardımcı

olunabilir [71]. WBCE kayıp hesaplama fonksiyonu denklem 2.40' daki gibi tanımlanır.

$$WBCE(x, y) = - [w_1 x \log(y) + w_0 (1 - x) \log(1 - y)] \quad (2.40)$$

Burada w_0 negatif sınıf için tanımlanan katsayı, w_1 pozitif sınıf için tanımlanan katsayı, x gerçek sınıfı, y tahmin edilen sınıfı temsil etmektedir. Yeterince temsil edilemeyen sınıflardan alınan örneklere daha yüksek ağırlık atayarak, modelin bu örneklere daha fazla dikkat etmesi teşvik edilir ve modelin genel performansı geliştirilebilir.

2.6.3. Kategorik Çapraz-Entropi Fonksiyonu

Kategorik çapraz entropi fonksiyonu(CCE), çoklu sınıflandırma problemlerinde kullanılan önemli bir kayıp hesaplama fonksiyonudur. Çoklu sınıflandırmalar one-hot kodlu etiketler ile yapılır. Örneğin araba, kamyon ve bisikletten oluşan üç sınıflı bir veri setimiz olsun. Üç kategorili bir değişkeni one-hot kodlu vektör ile kodlarsak araba $\Rightarrow [1, 0, 0]$, kamyon $\Rightarrow [0, 1, 0]$, bisiklet $\Rightarrow [0, 0, 1]$ şeklinde vektörler ile temsil edilir. Sınıflar içerisinde gerçek dağılım ile tahmin edilen olasılık dağılımı arasındaki farklılığı ölçer [72, 73]. Kategorik çapraz entropi kaybı formülü denklem 2.41' deki gibi tanımlanır.

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C x_{ij} \log(y_{ij}) \quad (2.41)$$

Burada N örnek sayısını, C ise sınıf sayısını temsil eder. x_{ij} , j -th kategorisine ait olan i -th eğitim kalıbını tespit eden ikili gösterge fonksiyonudur. C sınıfına ait i th örnek için x_{ij} gerçek değer, y_{ij} ise tahmin edilen olasılık olarak yorumlanabilir. Burada kayıp her örnek için hesaplanır ve tüm veri kümesi üzerinden ortalaması alınır.

2.6.4. Seyrek Kategorik Çapraz-Entropi Fonksiyonu

Seyrek kategorik çapraz – entropi kaybı sinir ağlarında çoklu sınıflandırmalarda yaygın olarak kullanılan kayıp fonksiyonudur. Sınıflandırma indexleri one-hot şeklinde kodlanmış vektörler ile değil tek bir tamsayı ile kodlanır. One-hot şeklinde kodlama hesaplama maliyet ve hafıza açısından verimli değildir. Seyrek kategorik çapraz-entropi kayıp fonksiyonu denklem 2.42' deki gibi tanımlanır.

$$L(x, y) = -\log(y_x) \quad (2.42)$$

Burada x gerçek sınıf etiket değerini, y ise sınıflardaki tahmin edilen olasılıksal değerdir. y_x ise gerçek x sınıfı için yapılan tahmindir. Bu fonksiyon toplu N sayıda örneğe uygulandığında seyrek kategorik çapraz-entropi kayıp fonksiyonu denklem 2.43'deki gibi tanımlanır.

$$L(x, y) = -\frac{1}{N} \sum_{i=1}^N \log(y_{i,x_i}) \quad (2.43)$$

Burada, x veri kümesindeki gerçek sınıf etiket vektörünü, y ise veri kümesindeki tahmin edilmiş olasılıksal dağılımlarını gösterir matrisi, x_i veri kümesindeki i -th örnekteki gerçek sınıf etiketini, y_{i,x_i} değeri de i -th örneğin gerçek sınıfı için tahmin edilen olasılığı temsil etmektedir.

Kategorik çapraz entropi, one-hot şekilde kodlanmış etiketlerle uğraşmak yerine tamsayı etiketleri ile kodlanır. Örneğin, 3 sınıflı sınıflandırma problemi için one-hot kodlama $[1,0,0]$, $[0,1,0]$ ve $[0,0,1]$ şeklinde yapılır. Seyrek kategorik çapraz entropide kaybında ise etiketler $[1]$, $[2]$ ve $[3]$ şeklinde kodlanır.

2.6.5. Etiket Yumuşatma İle Çapraz-Entropi Fonksiyonu

Etiket yumuşatma, modelin tahminleri gerçekleştirilirken aşırı güvenmesini önlemek için kullanılan bir tekniktir. Bir modeli, büyük bir miktarda veri ile eğitirken, model tahminlerine aşırı şekilde güvenmesi yaygındır. Bu aşırı güven, görünmeyen veriler üzerinde düşük performansla yol açabilir. Etiket yumuşatma, modeli daha az güvenilir tahminler yapmaya teşvik ederek bu sorunun azaltılmasına yardımcı olur. Sınıf örneklerinin küçük bir kısmı sınıfın tamamına dağıtılarak modelin güvenini azaltmak için one-hot kodlu etiketler tekrar düzenlenir. Dolayısıyla, modelin genelleme yapmasını geliştirir [74, 75]. Etiket yumuşatmayla Çapraz Entropi kaybı formülü denklem 2.44' de tanımlanmıştır.

$$L(x, y) = -\sum_{c=1}^C [(1 - \alpha)x_c \log y_c + \frac{\alpha}{C} \log y_c] \quad (2.44)$$

Burada x gerçek etiket, y tahmin edilen etiket, C sınıf sayısı ve α yumuşatma değeridir. Genellikle sabit α katsayısına 0.1 veya 0.2 gibi küçük bir değerler atanır. Etiket yumuşatma her zaman performansı iyileştirmez, belirli bir görev ve veri seti için en iyi değeri bulmak için farklı α değerleriyle denemeler yapmak yaygındır.

2.6.6. Odak Kayıp Fonksiyonu (Focal Loss)

Odak kaybı fonksiyonu, sınıf dengesizliği problemine çözüm getirmek için 2017 yılında T.Lin v.d. tarafından önerilmiştir [76]. Odak kaybı fonksiyonu, standart çapraz entropi kaybının değişik geliştirilmiş bir sürümüdür. Veri setindeki bazı sınıflardaki örnek sayısı, diğer sınıflardaki örneklerin sayısından daha küçük olması durumunda sınıf dengesizliği problemi ortaya çıkar. Bu gibi durumlarda model daha çok örnek sayısı fazla olan sınıfa odaklanma eğilimine girer ve bu durumun sonucunda modelin performansın düşmesine neden olan az örneğe sahip olan sınıflar ihmal edilebilir. Odak kaybı fonksiyonun amacı, iyi sınıflandırılmış olan örneklerin ağırlığını azaltıp eksik sınıflandırılmış örneklerin ağırlığını artırmak, eksik örneğe sahip olan sınıflara daha iyi odaklanıp ortaya çıkan sınıf dengesizliği problemini mümkün olduğu kadar ortadan kaldırmaktır. Odak kaybı fonksiyonu denklem 2.45'deki gibi tanımlanır.

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \quad (2.45)$$

Burada p_t gerçek sınıf için tahmin edilen olasılık değeri, α_t t sınıf için ağırlıklandırma faktörü, γ örneklerin ağırlığı az olan sınıflardaki oranı kontrol eden odaklanma parametresidir. γ odaklanma parametresi [0,5] aralığında değer alır ve $\gamma=2$ değeri verildiğinde deneylerden en iyi sonuç alındığı görülmüştür [76]. $\gamma = 0$ olduğunda odaklanma formülü çapraz entropi fonksiyonuna dönüşür. Formülde, p_t değeri küçük bir değer olduğunda kayıp etkilenmez ancak p_t değeri 1 olduğunda iyi sınıflandırılmış örneklerin ağırlığı azaltılır. Odak kaybı fonksiyonu nesne algılama, anlamsal bölümlendirme [77, 78] ve insan poz tahminleme [79] gibi çalışmalarda kullanılmıştır.

2.6.7. Hinge Kayıp Fonksiyonu

Bu fonksiyon özellikle destek vektör makinaları(SVMs)[14] gibi sınıflandırıcılarda yaygın olarak kullanılmış olan kayıp fonksiyonudur[69]. Hinge fonksiyonu denklem 2.46' da tanımlanmıştır.

$$L_H(x, y) = \frac{1}{N} \sum_{i=1}^N \max(0, 1 - x_i \cdot y_i) \quad (2.46)$$

Burda N örnek sayısı, x_i i-th örnek için gerçek etiket, y_i i-th örnek için tahmin edilen skordur.

2.6.8. Ortalama Yanlış Hata Kayıp Fonksiyonu

Yanlış pozitif oran(FPR) ve yanlış negatif oran(FNR) konseptinden ilham alınarak S.Wang v.d. tarafından önerilen ortalama yanlış hata(MFE) kayıp fonksiyonu, sınıf dengesizliğine sahip veri setleri üzerinde eğitilecek olan modelin performansını iyileştirmek için geliştirilmiştir[80]. Bu fonksiyon, farklı sınıfların hatalarını ayrı şekilde hesaplayarak azınlık sınıfındaki hatalara karşı daha duyarlı olmasını sağlar. Denklem 2.47' de yanlış pozitif hata, denklem 2.47a' da yanlış negatif hata, denklem 2.47b' de ortalama yanlış hata kaybı, denklem 2.47c' de ise ortalama kare yanlış hata kaybı tanımlanmıştır.

$$\text{Yanlış Pozitif Hata} = FPE = \frac{1}{N} \sum_{i=1}^N \sum_n \frac{1}{2} (d_n^{(i)} - y_n^{(i)})^2 \quad (2.47)$$

$$\text{Yanlış Negatif Hata} = FNE = \frac{1}{N} \sum_{i=1}^P \sum_n \frac{1}{2} (d_n^{(i)} - y_n^{(i)})^2 \quad (2.47a)$$

$$\text{Ortalama Yanlış Hata} = MFE = FPE + FNE \quad (2.47b)$$

$$\text{Ortalama Kare Yanlış Hata} = MSFE = FPE^2 + FNE^2 \quad (2.47c)$$

Burada N negatif sınıftaki örneklerin sayısını, P pozitif sınıftaki örneklerin sayısını, $d_n^{(i)}$ nth nöron üzerindeki ith örnekteki gerçek değeri, $y_n^{(i)}$ nth nöron üzerindeki ith örnekteki tahmin edilen değeri temsil eder. Ortalama yanlış hata kayıp fonksiyonu yanlış pozitif hata ile yanlış negatif hata değerlerinin toplamına eşittir. Ortalama kare yanlış hata fonksiyonu ise MFE kaybının performansını iyileştirmek için tasarlanmıştır.

2.7. Evrişim Sinir Ağlarında Eğitim Öncesi Süreç

CNN, özellikle görüntü verilerini tanımak, sınıflandırmak ve görüntü üzerindeki nesneleri tespit etmek için yaygın olarak kullanılan, karmaşık bir mimariye sahip ve içerisinde birçok algoritma barındıran bir derin öğrenme sinir ağı modelidir. Bir veri setinin CNN ile düzgün bir şekilde eğitebilebilmesi için eğitim öncesi çok sayıda düzenlemeye ihtiyaç vardır. Bu bölümde CNN modelini eğitmeden önce ne gibi aşamalardan geçilir, eğitilecek olan modelin performansı nasıl artırılır, eğitim öncesi dikkat edilmesi konular ve düzenlemeler ayrıntılı bir şekilde irdelenmeye çalışılacaktır [2].

2.7.1. Veri Toplama, Etiketleme ve Verilerin Yeniden Boyutlandırılması

Modeli eğitmeden önce çalışma ile alakalı görüntüler dijital kameralardan veya başka kaynaklardan elde edilerek toplanır. Modelin iyi bir şekilde eğitilebilmesi ve modelden istenen performansın alınabilmesi için veri setinin gerektiği kadar zenginleştirilmesi, her bir kategoride yer alan örneklerin olabildiğince farklı görüntüler içermesi gerekir. Veri setinin çok sayıda farklı türde örnek içermesi modelin doğruluk oranının artmasına ve modelin aşırı öğrenmesine engel olacaktır. Toplanan veriler ne kadar fazla ve çeşitli olursa modeldeki performans o kadar artar. Aksi taktirde model test verileri üzerinde test edildiğinde modelin performansının düşmesine neden olabilir.

Çalışmadaki verileri toplarken dikkat edilmesi gereken bir başka konu görüntülerin yüksek çözünürlüklü olmasıdır. Görüntülerin yüksek çözünürlükte olması, evrişim işleminde görüntülerdeki ince ayrıntıların yakalanabilmesine ve görüntülerin doğru bir şekilde sınıflandırılabilmesine yardımcı olur.

Veriler bir araya getirildikten sonra etiketleme işlemine geçilir. Çalışmada yer alacak kategoriler belirlenir. Kategoriler belirlendikten sonra bu görüntüler veri etiketleme programlarının yardımı ile etiketlenir. Çok sayıda örnek içeren zengin bir veri setinin etiketlenmesi işlemi çalışmadaki zorlu süreçlerden biridir ve zaman açısından maliyetli bir eylemdir. Ayrıca, etiketleme işlemi yapılırken verilerin doğru bir şekilde sınıflandırılması modelin performansı açısından önemlidir. Aksi taktirde modelin doğruluğunu artırmada istenen performans alınamayabilir.

Görüntülerin CNN ile eğitilebilmesi için etiketlenen görüntülerin boyutlarının aynı standarta sahip olması gerekir. Veriler etiketlendikten sonra görüntüler CNN' in eğitebileceği 320x320, 480x480, 640x640 gibi boyutlara dönüştürülür [2].

2.7.2. Veri Setinin Bölünmesi

Toplanan verilerin CNN ile eğitiminden önce bölme işlemi gerçekleştirilir. Veri seti eğitim, doğrulama ve test gibi kategorilere ayrılarak bölünür. Model, eğitim verilerini kullanarak eğitilir. Model eğitim verileri sayesinde öğrenir. Veri setinin yaklaşık %70'i eğitim verisi olarak ayrılır. Eğitim döngüsünün sonunda modelin performansı doğrulama verileri üzerinde değerlendirilir. Hatalar hesaplanarak ağırlıklar tekrar güncellenir. Modelin eğitim verilerini aşırı öğrenmesi sorunun çözümüne yardımcı olur. Toplam veri setinin yaklaşık %20'si doğrulama seti olarak ayrılır. Eğitim tamamlandıktan sonra, eğitilen modelin performansı test verileri

üzerinde test edilerek değerlendirilir. Test verisi, toplam veri setinin yaklaşık %10' luk kısmından oluşur.

2.7.3. Ağırlıkların Başlatılması

Evririşim sinir ağında ağırlıkların başlatılması, modelin eğitim sürecinde kayda değer bir şekilde etki eder. Ağırlıkların başlatılma stratejisi üzerinde durulması gereken önemli konulardan biridir ve kritik öneme sahiptir. Ağırlık başlatma, sinir ağı katmanının ağırlık değerinin başlangıç değerlerinin ayarlanma şeklini temsil eder [81, 82]. Ağırlığın başlatılma tekniği, CNN modelinin ne kadar hızlı yakınsayacağını ve ne kadar doğru bir şekilde sonlanabileceğini direkt olarak saptar.

Derin öğrenme algoritması ilk kez başarılı bir şekilde önerildiği zaman, ortalamayı sıfıra, standart sapmayı 0.01' e eşitleyerek ağırlıkların Gauss gürültüsü ile başlatılması yaygın bir işlemdi. Ağırlıkları başlatmadaki bu yöntem, kaybolan gradyanlar, patlayan gradyanlar veya ölü nöron [82, 83] gibi problemlerden dolayı derin sinir ağlarının iyi bir şekilde eğitilebilmesine yeterince yardımcı olmamıştır. 2010 yılında Glorot v.d. [30] problemi sistematik olarak analiz etmiş, girdi ve çıktı birimlerinin sayısına bağlı olarak ağırlıkları başlatmak için bir formül önermiştir. Glorot başlatıcıları çoğu durumda iyi çalışır ve bugün hala popülerdir. 2015 yılında He v.d. [29] Glorot başlatıcısının ReLU aktivasyon fonksiyonuyla iyi çalışmadığını ileri sürmüş ve formülü ReLU tabanlı sinir ağlarını karşılayacak şekilde yalnızca giriş birimlerinin sayısını kullanarak ve ölçeklendirmeyi $\sqrt{2}$ ile artırarak genişletmiştir [84]. Bu bölümde en sık kullanılan ağırlık başlatma teknikleri üzerinde durmaya çalışacağız.

2.7.3.1. Sıfır ve Bir ile Başlatma Tekniği

Sıfır ile başlatma tüm ağırlıkların sıfırlanması işlemidir. Ağırlıklar sıfır ile başlatıldığında, geri yayılım sırasında her bir katmandaki nöronlar aynı çıktıyı ve gradyanı hesaplar. Dolayısıyla, model doğrusal bir model gibi davranıp modelin veriler üzerinde farklı özellikleri öğrenmesi zorlaşır.

Bir ile başlatma tüm ağırlıkların bir ile eşleştirilmesi işlemidir. Sıfır ile başlatma tekniğinde olduğu gibi her katmandaki gradyan hesaplamaları aynıdır ve modelin farklı özellikleri öğrenmesinde etkili değildir. Ayrıca, ağırlıkları bir ile başlatma tekniğinde hesaplanan gradyanların ya aşırı küçük ya da aşırı büyük olabilmesine sebebiyet verebilir. Bu da, kaybolan gradyan ve patlayan gradyan problemlerine neden olabilir.

2.7.3.2. Rastgele Başlatma Tekniği

Rastgele başlatma tekniğinde, tekdüze ve gauss olasılık dağılımı ile ağırlıklara rastgele küçük değerler verilir. Bu teknik sıfır ve bir ile başlatma tekniğinde görülen simetri probleminin çözülmesine yardımcı olarak ağırlıklardaki her bir nöronun veriler üzerinde farklı özellikleri öğrenebilmesine olanak sağlar. Ancak, eğitim süreci içerisinde ağırlıklar aktivasyon fonksiyonları ile düzgün bir şekilde normalize edilmez ise hesaplanan gradyanlar, kaybolan ve patlayan gradyan problemlerinin oluşmasına sebebiyet verebilir. Rastgele başlatma tekniği gauss, tekdüze ve ortogonal yöntemler içerir [2].

- a) Tekdüze Rastgele Başlatma : Bu yöntemde ağırlıklar, belirli bir aralık içerisinde tek düze dağılımla rastgele değerler ile başlatılır.
- b) Gauss Rastgele Başlatma : Ağırlıklar belirli bir standart sapma ve ortalama kullanılarak gauss dağılımı ile rastgele başlatılır.
- c) Ortogonal Rastgele Başlatma: Burada ağırlıklar ortogonal matrisler kullanılarak rastgele başlatılır. Aktivasyon ve gradyan değişimini koruyarak kaybolan ve patlayan gradyan problemlerinin çözümüne yardımcı olur.

2.7.3.3. LeCun Başlatma Tekniği

Ağırlıkları LeCun ile başlatma, geri yayılım süresi boyunca gradyanların yok olmasını ve patlamasını önlemeyi amaçlar. Bu başlatma tekniği, girdi sayıları ile büyüyen varyansı çözer ve sabit varyans ayarlar [81].

2.7.3.4. Xavier Başlatma Tekniği(Glorot Başlatma)

Ağın hızlı bir şekilde yakınsayabilmesi ve aktivasyon fonksiyonlarının değişimini dengelemek için 2010 yılında X.Glorot v.d. [30] tarafından Xavier başlatma tekniği önerilmiştir. Rastgele başlatılan derin sinir ağlarında sigmoid aktivasyon fonksiyonunun elverişli olmadığı ve hesaplanan gradyanlar doyuma ulaşabilmekte veya kaybolabilmektedir. Bu teknik gradyanların kaybolması ve patlaması probleminin çözümüne yardımcı olur. Bu teknik günümüzde çoğunlukla kullanılan ReLU aktivasyon fonksiyonu ile aynı performansı göstermez. ReLU aktivasyon fonksiyonu için K.He v.d. tarafından He başlatma tekniği önerilmiştir.

2.7.3.5. He Başlatma Tekniği (Kaiming Başlatma)

Kaiming başlatma tekniği olarak da adlandırılan He başlatma tekniği, ReLU vb. düzeltici aktivasyonlarında kullanılması için geliştirilen ağırlık başlatma

teknikidir [29]. He başlatma tekniği, sinir ağındaki her bir katmandaki aktivasyonların korunmasına, kaybolan ve patlayan gradyan problemlerinin çözümüne yardımcı olur.

2.8. CNN Performansının İyileştirilmesi İçin Geliştirilen Teknikler

Evrişimsel sinir ağlarında eğitim sırasında modelin aşırı uyum göstermesi ya da modelin yeteri kadar öğrenememesi gibi sorunlar sık karşılaşılan problemler arasında yer almaktadır. Model eğitim verileri üzerinde aşırı uyum gösterip sisteme tanıtılmamış test verileri üzerinde istenen performansı gösteremez ise bu durum aşırı uyum problemi olarak adlandırılır. Yetersiz öğrenme problemi ise model eğitim verileri üzerinde yeterince öğrenme yapamadığı durumlarda ortaya çıkar. Model, hem eğitim verileri üzerinde hem de test verileri üzerinde çok iyi performans gösteriyor ise bu, arzu edilen bir durumdur ve mükemmel uyum olarak adlandırılır.

Ayrıca, kayıpları en aza indirmek için geri yayılım sırasında ağırlıklar sürekli güncellenir ve bu durumdan aktivasyon dağılımlarının varyansı etkilenebilir. Değişen aktivasyon dağılımların ortalamasının kaymasına neden olabilir. Bu kaymayı önlemek için çok sayıda teknik geliştirilerek ağı daha istikrarlı olması sağlanır.

Bu bölümde aşırı ve yetersiz öğrenmeyi önlemek, ağı daha istikrarlı hale getirmek için geliştirilen teknikler üzerinde durmaya çalışacağız [1, 85].

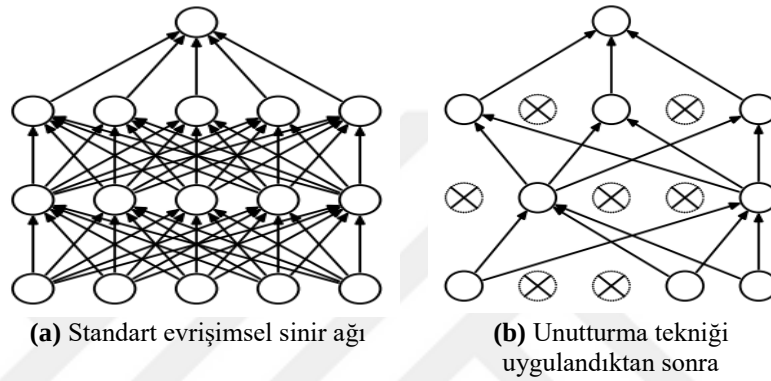
2.8.1. Unutturma Teknikleri

CNN modeli eğitilirken, aşırı uyum problemi sık karşılaşılan bir durum olup bu sorunu çözmek için CNN mimarisinin tasarım aşamasında bazı unutturma tekniklerin kullanılması modelin performansını artırılmasına yardımcı olabilir.

2.8.1.1. Nöronları Unutturma(Dropout) Tekniği

Evrişimsel sinir ağları çok sayıda nörona sahiptir ve çok sayıda nöron ağına yavaşlamasına ve modelin aşırı uyum gösterebilmesine neden olabilir. Unutturma tekniği[86] 2012 yılında N. Srivastava v.d. tarafından önerilmiştir. Bu teknikte, ağı içerisindeki nöronlar eğitim sırasında rastgele unutturularak modelin aşırı öğrenmesinin önüne geçilmiş olur. Sinir ağındaki nöronları unutturmak için unutturma parametresi kullanılır. Unutturma parametresinin çok iyi ayarlanması gerekir. Eğer unutturma oranı çok yüksek olursa modeldeki öğrenilen parametreler kaybedileceğinden model eğitim sırasında yeterince öğrenme yapamaz, eğitim süresi

uzar ve bu durum yetersiz öğrenme problemini ortaya çıkarır. Eğer unutturma oranı çok az olursa da modelin aşırı öğrenmesi probleminin önüne geçilemeyebilir. [0.2, 0.5] aralığında kullanılan unutturma parametreleri yaygın olarak kullanılan oranlar arasında yer alır [1, 87]. Sinir ağına uygulanan unutturma tekniği modelin aşırı öğrenmesini önler ancak eğitim süresini 2-3 kat uzatabilir. Bunun nedeni ağda güncellenen gradyanların çok gürültülü olmasıdır. Her eğitim süreci sonunda model farklı rastgele bir ağ mimarisine sahip olur. Hesaplanan gradyanlar her zaman aynı olmaz bu da aşırı öğrenmeyi önler. Şekil 2.8’ de evrişimsel sinir ağına uygulanan unutturma tekniği gösterilmektedir.



Şekil 2.8 : Unutturma tekniği

Sonuç olarak unutturma tekniği, aşırı öğrenme ile modelin eğitim süreci arasında hassas bir denge oluşmasına neden olur. Yüksek unutturma oranı aşırı öğrenmeyi önler ancak modelin eğitim sürecini uzatır. Düşük unutturma oranı da eğitim sürecini kısaltırken modelin aşırı öğrenmesini önlemede yetersiz olabilir.

2.8.1.2. Maksimum Unutturma(MaxDropout) Tekniği

Unutturma[86] tekniğinde ağ içerisindeki nöronlar unutturulurken, maksimum unutturma[88] tekniğinde ise aktivasyon tabanlı unutturma işlemi gerçekleştirilir. Bu teknikte, modelin genelleme yeteneğini pozitif bir şekilde etki eden faydalı ve daha fazla bilgileri öğrenmiş olan kritik öneme sahip en çok aktive edilmiş olan nöronlar unutturulur.

2.8.1.3. Ağırlıkları Unutturma(DropConnect) Tekniği

Ağırlıkları unutturma tekniği [89], normal unutturma tekniğinin farklı düzenlenmiş bir sürümü olarak görülebilir. Unutturma tekniğinde her bir katmanda yer alan rastgele seçilmiş olan nöronlar unutturulurken dropconnect tekniğinde ise,

ağ içerisinde yer alan rastgele seçilmiş ağırlıklar unutturulur. Bu teknik, aşırı öğrenmeyi önler ancak normal unutturma tekniğinden biraz daha yavaş çalışır.

2.8.1.4. Blokları Unutturma(DropBlock) Tekniği

Blokları unutturma tekniğinde[90], evrişim sonrası elde edilen özellik haritasındaki bitişik bölgeler unutturulur. Bu teknik, CNN' in her özellik haritasında uygulanır. Blok boyutu ve λ olmak üzere iki parametreye sahiptir. Blok boyutu parametresi, unutturulacak olan blokların boyutunu; λ parametresi ise unutturulacak olan aktivasyon birimlerinin sayısını kontrol eder. Ayrıca bu teknikte unutturulan nöronların sayısı yavaş yavaş artırıldığında, modelden daha iyi sonuçlar elde edilmiştir [90].

2.8.1.5. Diğer Unutturma Teknikleri

Aşırı öğrenmeyi önlemek için geliştirilen bir diğer unutturma tekniği hedef unutturma(targetdrop)[91] tekniğidir. Rastgele unutturma yöntemini uygulayan diğer tekniklerin aksine bu teknikte, küresel havuzlama yönteminden faydalanılarak özel bilgiler içeren özellik haritaları kanal tabanlı vektörler içerisine alınır. Elde edilen bu kanallar içerisinde ayırt edici bilgilerin nerede olduğu saptanır. Önemli bilgiler içeren özellikler zekice unutturulur.

Diğer bir teknik olan otomatik unutturma(autodrop)[92] tekniğinde, unutturma kalıplarını belirleyen bir kontrol mekanizması geliştirilir. Bu kontrol mekanizması, eğitim sırasında sinir ağının her bir katmanında en iyi unutturma kalıplarını öğrenir. Öğrenilen unutturma kalıpları diğer veri setlerine aktararak tekrar kullanılabilir. Ancak bu teknik hesaplama maliyeti açısından masraflı bir yöntemdir.

2.8.2. Normalizasyon Teknikleri

Evrişim sinir ağlarında her bir eğitim döngüsünde, kayıpları minimize etmek için katmanlardaki ağırlıklar devamlı bir şekilde geri yayılım algoritması ile güncellenir. Katmanlardaki ağırlıkların sürekli olarak güncellenmesi aktivasyon dağılımlarının ya artmasına ya da azalmasına yol açarak aktivasyon dağılımının ortalamasının ve varyansının değişebilmesine neden olabilir. Güncellenen bu ağırlıklar, aktivasyon değerlerinin ortalamasını kaydırabilir ve bu durum iç kovaryans kayması(internal covariate shift) olarak adlandırılır. Ortalaması ve varyansı değişen, bunun sonucunda standart sapması etiklenen bu aktivasyon

değerleri sonraki katmana girdi olarak aktarılır. Sonraki katmana girdi olarak kabul edilen varyansı kaymış olan bu aktivasyon dağılımları tekrar ayarlanır. Bu durum modelin öğrenmesini olumsuz etki eder ve eğitim sürecini uzatır. Ayrıca, aktivasyon değerleri bir düzen içerisinde olmazlar ise patlayan ve kaybolan gradyan problemlerin oluşmasına sebebiyet verebilir. Aktivasyon dağılımındaki bu kaymayı tekrar orijinal haline getirmek için normalizasyon teknikleri geliştirilmiştir. Her eğitim döngüsünde ağ katmaları içerisinde uygulanan normalizasyon teknikleri, modelin genel performansının iyileştirilmesine ve eğitim hızının arttırılmasına yardımcı olur. Veriyi düzenli bir dağılım içerisine koyarak eğitim boyunca ağın daha hızlı yakınsamasına olanak sağlar. Hesaplanan gradyanların kaybolmasını veya patlamasını önler. Modelin daha iyi öğrenebilmesine ve test verileri üzerinde daha iyi performans göstermesine yardımcı olur. Bu kısımda, geliştirilen bazı normalizasyon tekniklerinden bahsetmeye çalışacağız.

2.8.2.1. Toplu Normalleştirme

Toplu normalleştirme(batch normalization) 2015 yılında S.Ioffe v.d. [93] tarafından önerilmiştir. Toplu normalleştirme, eğitim süreci içerisinde her katmana ait aktivasyon değerlerinin ortalamasının sıfır ve varyansının bir olmasını sağlayarak normalize eder. Ağ içerisindeki bilgilerin korunmasına yardımcı olur.

Toplu normalizasyon işleminde aktivasyon verileri küçük veri kümelerine bölünür. Her bir küçük veri setindeki aktivasyon değerlerinin ortalaması ve varyansı hesaplandıktan sonra veriler normalize edilir. Verilerin normalize edilmesi işleminden sonra, normalize edilmiş değerlere gama (γ) ve beta (β) öğrenilebilir parametreler uygulanır. γ parametresi normalize edilmiş olan aktivasyon değerlerinin ölçeğini ayarlarken, β parametresi de aktivasyonların kaymasını ayarlar.

Aktivasyon verileri $B = [a_1, a_2, \dots, m]$ şeklinde küçük veri kümelerine bölünür. Denklem 2.48 ortalama alma, denklem 2.48a varyans hesaplama, denklem 2.48b normalizasyon işlemi ve denklem 2.48c de yapılan normalizasyonun ölçeklendirme ve kaydırma işleminin formülü tanımlanmaktadır.

$$\mu_B = \frac{1}{m} \sum_{i=1}^m a_i \quad (2.48)$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (a_i - \mu_B)^2 \quad (2.48a)$$

$$\hat{a}_i = \frac{a_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (2.48b)$$

$$b_i = \gamma \hat{a}_i + \beta \quad (2.48c)$$

Burada, m küçük aktivasyon veri kümelerinin sayısını, a_i veri setindeki elemanı, ϵ sıfır ile bölümden kaçınmak için formüle eklenen küçük sabit bir katsayı, γ ölçeklendirme parametresini, β kaydırma parametresini temsil eder.

γ ve β parametreleri sayesinde sinir ağı, aktivasyon dağılımının orijinal halini tekrar öğrenir. Eğer ϵ katsayısı küçük bir katsayı olduğu için dikkate alınmaz ise, normalize edilmiş olan aktivasyon verilerinin ortalaması 0 ve varyansı 1 olur.

Aktivasyon verilerinin tamamını bir anda normalize etmek hesaplama maliyetini artırır. Veri setini küçük veri setlerine bölerek ayrı ayrı normalize etmek ağıın belirli bir düzen içine girmesine yardımcı olur. Her bir küçük veri setindeki ortalama, varyans ve standart sapmalar eş zamanlı olarak hesaplanabilir, bu durum eğitim sürecini hızlandırır.

Sonuç olarak normalizasyon işlemi, her katmanda yer alan aktivasyon verilerin ortalamasını sıfıra, varyansını bir değerine yaklaştırarak aktivasyon verilerinin daha tutarlı bir şekilde dağılmasına, iç kovaryans kaymasının da mümkün olduğu kadar azaltılmasına, daha yüksek öğrenme oranlarının kullanılmasına olanak sağlayarak eğitim sürecinin hızlanmasına, ağıın daha hızlı yakınsamasına, patlayan ve yok olan gradyan probleminin oluşabilmesini önleyerek geri yayılım sırasında hesaplanan gradyanların daha tutarlı olmasına yardımcı olarak sinir ağıının performansının iyileştirilmesine, verimliliğinin artmasına ve daha istrarlı olmasına olanak sağlar.

2.8.2.2. Ağırlıkların Normalize Edilmesi

Optimizasyon sürecinin verimliliğini iyileştirmek ve gradyan hesaplamasını hızlandırmak için sinir ağıındaki ağırlık parametreleri tekrar güncellenir[94]. Toplu normalizasyon tekniğinden[93] ilham alınarak geliştirilmiştir. Bu teknikte, optimizasyon sürecini hızlandırmak için, gradyan yayılımı gerçekleştirilirken ağırlık vektörünün parametrelerine yeniden ayar yapılır. Ağırlık vektörünün normalize edilmesi denklem 2.49' daki gibi tanımlanır.

$$w = \frac{g}{\|v\|} v, \quad \|v\| = \sqrt{\sum_i v_i^2} \quad (2.49)$$

Burada g ağırlık vektörünün ölçeğini kontrol eden skalar bir değer, v k boyutlu yön vektörü, $\|v\|$ ise v vektörünün L2 normudur. İleri yayılımda,

parametreleri yeniden normalize edilen ağırlık vektörü kullanılarak her bir nöron denklem 2.50' deki gibi hesaplanır.

$$y = \sigma(w.x + b) \quad (2.50)$$

Burada w “ k ” boyutlu normalize edilmiş ağırlık vektörünü, x “ k ” boyutlu girdi vektörünü, b skalar sapma parametresini ve σ ise aktivasyon fonksiyonunu temsil eder.

Eğitimin geri yayılım sırasında kayıplar hesaplanır, optimizasyon algoritmaları ile parametreler güncellenir ve sonrasında ağırlıklar tekrar normalize edilir. Ağırlıkların normalize edilmesi, optimizasyon algoritmalarının daha iyi hesaplanmasını sağlar, hesaplanan gradyanların boyutunu kontrol ederek yok olan ve patlayan gradyan probleminin oluşumunu engeller ve daha az hesaplama ihtiyacı duyar. Ancak, toplu normalizasyon işlemi küçük veri setlerinin boyutuna bağlıdır. Küçük veri setlerinin boyutu çok küçük olursa, hesaplanan aktivasyonların ortalaması ve varyansı çok değişiklik gösterebilir bu da eğitimin yakınsamasını zorlaştırabilir.

2.8.2.3. Diğer Normalizasyon Teknikleri

Normalizasyon tekniklerinden biri olan katman normalizasyonu(layer normalization)[95] 2016 yılında J.L. Ba v.d. tarafından önerilmiştir. Katman normalizasyon işlemi her bir veri örneği için katmandaki aktivasyon verilerini normalize ederek ağırlık istikrarını korumaya ve eğitim sürecinin azaltılmasına yardımcı olur. Toplu normalleştirilmenin aksine küçük veri setlerinin boyutuna bağımlı değildir. Her bir örneğin özellik vektöründeki aktivasyonların ortalamasını ve varyansını hesaplayarak iç kovaryans kaymasının oluşmasını azaltabilir. Ancak, her bir örnek için katmandaki aktivasyonların hesaplanması maliyetli olabilir.

2016 yılında D.Ulyanov v.d. tarafından önerilen örnek normalizasyon tekniğinde(instance normalization)[96] ise, her bir görüntü verisinin her bir kanalındaki aktivasyon verileri normalize edilir. Her görüntü içerisindeki her bir kanalın aktivasyon verilerinin ortalaması ve varyansı ayrı ayrı hesaplanır. Her bir görüntünün kanalında yapılan normalizasyon işlemi denklem 2.51, 2.51a, 2.51b ve 2.51c'de sırasıyla yapılır.

$$\mu_{ic} = \frac{1}{HW} \sum_{h=1}^H \sum_{w=1}^W a_{icwh} \quad (2.51)$$

$$\sigma_{ic}^2 = \frac{1}{HW} \sum_{h=1}^H \sum_{w=1}^W (a_{ichw} - \mu_{ic})^2 \quad (2.51a)$$

$$\hat{a}_{ichw} = \frac{a_{ichw} - \mu_{ic}}{\sqrt{\sigma_{ic}^2 + \epsilon}} \quad (2.51b)$$

$$b_{ichw} = \gamma_c \hat{a}_{ichw} + \beta_c \quad (2.51c)$$

Burada H görüntünün yüksekliğini, W görüntünün genişliğini, n örneği, c örnekteki kanalını temsil eder. Toplu normalizasyon işleminde olduğu gibi burada hesaplamalar her bir görüntünün her bir kanalı için yapılır. Denklem 2.51 görüntünün her bir kanalının ortalamasını, denklem 2.51a kanalın varyansını, denklem 2.51b kanaldaki aktivasyon verilerinin standart sapmasını hesaplar, denklem 2.51c ise γ ve β parametresi ile ağda oluşan iç kovaryans kaymasını ayarlar. Örnek normalizasyon tekniği katman tekniğinde olduğu gibi toplu normalizasyondaki küçük veri setlerinin boyutuna bağımlı değildir. Bu teknikte, sinir ağının performansının iyileştirilmesine yardımcı olur.

2018 yılında yayımlanan toplu örnek normalizasyon(batch-instance normalization)[97] tekniği, toplu normalizasyon ile örnek normalizasyon tekniğini birleştirerek normalizasyon yapan bir diğer tekniktir. Batch-instance normalizasyon tekniğinde önce normal toplu normalizasyon sonra aynı şekilde örnek normalizasyon değerleri tespit edilir. $\hat{a}^B$ hesaplanan normal toplu normalizasyon, $\hat{a}^I$ ise hesaplanan normal instance normalizasyon değeri olsun. Amaç her kanal için, $\hat{a}^B$ normalizasyonu ile $\hat{a}^I$ normalizasyonu arasında bir uyarlayıcı denge oluşturmaktır. Bunu sağlamak için [0,1] aralığında bir denge parametresi α katsayısına ihtiyaç duyulur. Toplu örnek normalizasyonu denklem 2.52' deki gibi tanımlanır.

$$y_{bi} = (\alpha \cdot \hat{a}^B + (1 - \alpha) \cdot \hat{a}^I) \gamma + \beta \quad (2.52)$$

Burada, α [0,1] aralığında bir denge parametresi, $\hat{a}^B$ hesaplanan toplu normalizasyon, $\hat{a}^I$ hesaplanan örnek normalizasyon, γ ve β parametreleri ölçeklendirme ve kaydırma katsayılarını, y_{bi} ise toplu örnek normalizasyon işleminin çıktısını temsil eder.

2.8.3. Kayıp Fonksiyonunu Düzenleme Teknikleri

Kayıp fonksiyonlarının yeniden düzenlenmesi, ağın genelleme yapma yeteneğini iyileştirerek aşırı uyumun önlenmesine yardımcı olan önemli düzenleme teknikleri arasında yer alır. Bu teknikler kayıp fonksiyonuna bir ceza puanı ekleyerek

modelin performansının iyileştirilmesine yardımcı olurlar. L1, L2 ve elastik ağ düzenlemesi gibi düzenleme teknikleri vardır.

2.8.3.1. L2 Düzenlemesi (Ridge)

L2 düzenlemesi model doğruluğunu artırmak, aşırı uyumu azaltmak ve modellerin genelleme yeteneğini geliştirmek için L2 normu kullanılarak elde edilen bir tekniktir. Öklit normu olarak bilinen L2 normu, vektör değerlerinin karelerinin toplamının kare kökü alınarak hesaplanır. L2 düzenlemesinde sinir ağındaki ağırlık vektörünün L2 normu hesaplanır ve hesaplanan L2 normuna bir ceza puanı eklenilerek L2 düzenlemesi elde edilir[87, 98-102]. L2 düzenlemesi formülü denklem 2.53' de tanımlanmıştır.

$$\|w\|_2^2 = \sum_{i=1}^M w_i^2, \quad L_{L2} = \lambda + \|w\|_2^2 \quad (2.53)$$

Burada, λ ağırlık kaybı katsayısı, M ağırlıkların sayısı, w_i i-th sırasındaki ağırlığı temsil eder. Ağırlık vektörünün L2 normu hesaplanır ve bu norma λ ağırlık kaybı katsayısı eklenerek L_{L2} düzenlemesi elde edilir. Düzenlenmiş kayıp fonksiyonu, orijinal kayıp fonksiyonu ile L_{L2} düzenleme teriminin toplamından oluşur. Düzenlenmiş kayıp fonksiyonu denklem 2.54' de tanımlanmıştır.

$$L_{Toplam} = L_{orijinal} + L_{L2} \quad (2.54)$$

Burada orijinal kayıp fonksiyonu herhangi bir kayıp fonksiyonu olabilir. Orijinal kayıp fonksiyonu ile düzenlenmiş L_{L2} kaybı toplanarak yeni bir kayıp fonksiyonu oluşturulmuş olur.

2.8.3.2. L1 Düzenlemesi (Lasso)

L1 düzenlemesinde L2 normunun yerine L1 normu kullanılır. L1 normu vektör komponentinin mutlak değerlerinin toplamıyla elde edilir. Büyük ağırlıklara sahip olan bir modelin aşırı uyumunu önlemek için L1 normuna bir ağırlık kaybı katsayısı eklenir. L1 düzenlemesi formülü denklem 2.55' de tanımlanmıştır.

$$\|w\|_1 = \sum_{i=1}^M |w_i|, \quad L_{L1} = \lambda + \|w\|_1 \quad (2.55)$$

Burada, λ ağırlık kaybı katsayısı, M ağırlıkların sayısı, w_i i-th sırasındaki ağırlığı temsil eder. Ağırlık vektörünün L1 normu hesaplanır ve bu norma λ ağırlık kaybı katsayısı eklenerek L_{L1} düzenlemesi elde edilir. Düzenlenmiş kayıp

fonksiyonu, orijinal kayıp fonksiyonu ile L_{L1} düzenleme teriminin toplamından oluşur. Düzenlenmiş kayıp fonksiyonu denklem 2.56' da tanımlanmıştır.

$$L_{Toplam} = L_{orijinal} + L_{L1} \quad (2.56)$$

2.8.3.3. Elastik Ağ Düzenlemesi

Elastik ağ düzenlemesi, L1 ve L2 düzenlemelerinin avantajlarını dengelemek için geliştirilen bir tekniktir. Elastik ağ düzenlemesinde her iki teknik birleştirilir. Denklem 2.57' de L1 ve L2 normu hesaplanması gösterilmektedir.

$$\|w\|_1 = \sum_{i=1}^M |w_i|, \quad \|w\|_2^2 = \sum_{i=1}^M w_i^2 \quad (2.57)$$

L1 ve L2 normu birleştirilerek elastik ağ düzenlemesi elde edilir. Elastik ağ düzenlemesi denklem 2.58' deki gibi tanımlanır.

$$L_{Toplam} = L_{orijinal} + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2 \quad (2.58)$$

Burada λ_1 L1 düzenlenmesinde kullanılan ağırlık kaybı katsayısı, λ_2 L2 düzenlenmesinde kullanılan ağırlık kaybı katsayısı, $\|w\|_1$ L1 normu, $\|w\|_2^2$ ise L2 normudur. L1 ve L2 düzenlemesi tekniklerinin birleştirilmesi ile elde edilen elastik ağ düzenlemesi, her iki tekniğin avantajlarından faydalananak sinir ağının eğitilmesinde önemli rol oynar.

2.8.4. Etiket Yumuşatma Teknikleri

Etiket yumuşatma tekniği sinir ağlarında modelin aşırı güvenini önlemek, ağın genelleme yapabilme yeteneğini iyileştirmek için eğitim sırasında gerçek etiketler üzerinde yapılan ayarlamalardır[75, 103]. Etiket yumuşatma basit matematiksel işlemlerden oluşur ve bu teknikte farklı amaçlar için farklı varyasyonlar uygulanabilir.

2.8.4.1. Tek Tip Etiket Yumuşatma Tekniği

Tek tip etiket yumuşatma tekniğinde, gerçek etiket değerlerine çok hafif ayar yapılır. Amaç, eğitim sırasında modelin tahminlerine daha az güven duymasını sağlamaktır[75]. Bu teknikte gerçek etiketler tek düze dağılım kullanılarak yumuşatılmıştır. Tek tip etiket yumuşatma denklem 2.59' da tanımlanmıştır.

$$L_{yumu\ satma}(k) = (1 - \epsilon)\delta_{k,y} + \frac{\epsilon}{K} \quad (2.59)$$

Burada ϵ yumuŝatma parametresini, K sınıf sayısını, y dođru tahmin edilen sınıfı, $\delta_{k,y}$ ise kronecker deltayı temsil eder. Kronecker delta iŝleminde $k=y$ ise 1, $k \neq y$ ise 0 olur. Örnek verirsek $y=[0,0,1,0,0]$ one-hot ile kodlanmış 5 sınıflı gerçek etiketlerimiz olsun. ϵ yumuŝatma katsayısı 0.1 olarak kabul ederek denklem 2.59' daki formülü uygularsak deđerleri yumuŝatılmış yeni one-hot kodlu etiketler $y^1 = [0.033, 0.033, 0.933, 0.033, 0.033]$ olur. Bu teknikte dođru tahmin edilen deđerlerin diđer deđerlerden daha büyük olması engellenerek modelin tahminlemede aşırı güven duyması önlenir, daha iyi genelleme yapabilmesini sağlar, aşırı uyumu azaltır.

2.8.4.2. Uyarlayıcı Etiket Yumuŝatma Tekniđi

Uyarlayıcı etiket yumuŝatma tekniđi, tek düze etiket yumuŝatma tekniđinin bir uzantısıdır[104]. Bu teknikte yumuŝatma parametresi sabit olmayıp modelin güveni ve verilerin durumuna göre deđiŝebilmektedir. Bu teknik modelin daha iyi performans göstermesine ve daha iyi genelleme yapabilmesine olanak sağlar. Uyarlayıcı etiket yumuŝatma denklem 2.60' da tanımlanmıştır.

$$L_{yumu\ ŝatma}(k) = (1 - \epsilon_i)\delta_{k,y} + \frac{\epsilon_i}{K} \quad (2.60)$$

Burada ϵ_i uyarlayıcı yumuŝatma parametresini, K sınıf sayısını, y dođru tahmin edilen sınıfı, $\delta_{k,y}$ ise kronecker deltayı temsil eder.

Sonuç olarak etiket yumuŝatma teknikleri modelin aşırı güven duymasını önlemek ve ađın daha iyi genelleme yapabilmesini sağlamak için geliŝtirilmiştir.

2.8.5. Veri Artırma Teknikleri

Veri artırma teknikleri yeni verilere ihtiyaç duyulmadan eđitim sırasında çok sayıda teknik kullanılarak mevcut veriler üzerinden veri setinin arttırılması iŝlemdir. Yeterli miktarda veri setine ulaŝılmasındaki zorluk ve etiketleme maliyeti veri setinin oluŝturulmasını zorlaŝtırmaktadır. Dolayısıyla eđitim sırasında mevcut veriler üzerinde yapılacak olan dönüşümler ve yeni oluŝturulan veri örnekleri sinir ađının eđitilmesine, ađın daha istrarlı olmasına ve aşırı öğrenme probleminin çözülmesine yardımcı olur[1, 2, 85, 105].

Veri artırma ölçeklendirme, yansıtma(flipping), renk dönüşümü, rastgele kırpma[106], dönüŝtürme, döndürme, gürültü enjeksiyonu[105], perspektif

dönüşümü[107], cutmix[108], mixup[109], rastgele silme[110], cutout[111], cutblur[112] gibi tekniklerden oluşur.

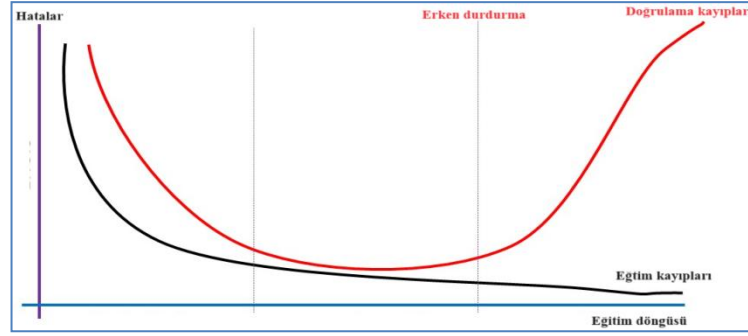
Ölçeklendirme tekniğinde bir görüntü yeniden boyutlandırılır, yansıtma işleminde bir görüntü hem dikey hem yatay şekilde çevrilerek yansıtılır, renk dönüşümü tekniğinde görüntünün renk tonu, doygunluğu ve parlaklığı gibi parametreleri değiştirilerek yeni görüntüler elde edilir. Rastgele kırpma tekniğinde dört görüntü rastgele kırılarak yeni bir görüntü oluşturulur, dönüştürme tekniğinde görüntü rastgele yukarı, aşağı, sola ve sağa kaydırılır ve döndürme tekniğinde ise görüntü kendi eksenini etrafında rastgele olacak şekilde döndürülür.

Gürültü enjeksiyonu yaklaşımında, gauss dağılımı ile rastgele değerlerden oluşan bir matris görüntüye enjekte edilir ve görüntü içerisinde rastgele gürültüler meydana gelir. Perspektif dönüşümü görüntünün perspektifini değiştirmeye çalışır.

Bir başka veri artırma tekniği olan cutmix işleminde, bir kategorideki görüntüden belirli bir oranda görüntü kesilip diğer sınıftaki bir görüntüye yama yapılır. Mixup tekniğinde, iki görüntünün pikselleri belirli bir oranda karıştırılır. Rastgele silme tekniğinde ise, görüntüden rastgele dikdörtgen şekiller seçilip içerisindeki pikseller belirli bir oranda rastgele silinir. Cutout tekniğinde, girdi görüntüsü rastgele kareler ile maskelenir ve son olarak cutblur tekniğinde ise görüntünün belirli bölgeleri düşük çözünürlük uygulanarak görüntü bulanıklaştırılır.

2.8.6. Erken Durdurma

Erken durdurma tekniği modelin aşırı uyumunu önlemek için sinir ağlarında yaygın olarak kullanılan bir tekniktir[2, 113]. Sinir ağı eğitilirken, ağıın genelleme performansının oldukça iyi olması arzu edilen bir durumdur. Doğrulama kayıpları ağıın genelleme hatasını tespit etmek için kullanılır. Doğrulama kayıpları arttığında ağıın genelleme yeteneğı azalır. Eğitim süreci boyunca ağıın doğrulama verileri üzerindeki performansı sürekli takip edilmeli, eğer doğrulama kayıplarında sürekli artış görülüyor ise ağıın genelleme yeteneğini kaybetmemesi için eğitim belirli bir döngüden sonra durdurulması gerekebilir. Şekilde 2.9' da ağıın eğitim sonuçlarını gösterir grafik görölmektedir. Grafikte göröldüğü üzere eğitim kayıpları azalırken belirli bir döngüden sonra doğrulama kayıplarının arttığı görölmektedir.



Şekil 2.9 : Eğitim sonuçlarını gösterir grafik

Erken durdurma işlemi, eğitimden önce tanımlanan bir sabır parametresi ile kontrol edilebilir. Erken durdurma işleminden önce eğitim sabır parametresindeki değer kadar bekler. Eğer ağı performansı daha kötüye gidiyor ve doğrulama kayıpları azalmayıp sürekli artıyor ise tanımlanan sabır parametresi kadar döngüden sonra eğitim sonlandırılır. Erken durdurma işleminin bir diğer tekniği, eğitim sırasında ağırlık parametrelerinin kayıt edilmesidir. Ağ, doğrulama verileri üzerinde performans iyileştirmesi gösteriyor ise belirlenen eğitim döngüsünde en iyi ağırlık parametreleri kayıt edilir.

2.8.7. Öğrenme Transferi

CNN modellerini eğitmek için çok büyük bir miktarda veri setine ihtiyaç vardır. Her çalışmada büyük miktarda veri setine ulaşmak her zaman mümkün olmayabilir ve modelin eğitiminde kullanılacak devasa veri setini etiketlemek de çok yorucu olabilir. Bu nedenle büyük veri setine ulaşamadığı durumlarda öğrenme transferi tekniğinden faydalanmak yararlı olabilir [114].

Öğrenme transferi, modelin performansını iyileştirmek için önceden eğitilmiş olan bir ağ modelini, yeni bir veri seti üzerinde yeni bir modelin eğitiminde kullanılması için geliştirilen bir tekniktir. Yeterli miktarda veri setine ulaşamadığı durumlarda ağı genelleme yapabilmesine ve aşırı uyumun önlenmesine yardımcı olmak için öğrenme transferinden yararlanılır. Önceden eğitilmiş modeller büyük veri setleri ile eğitildikleri için zengin özellik parametrelerine sahiptir ve eğitim başlarken öğrenilmiş olan bu özellikler eğitilecek olan modele aktarılır. Dolayısıyla, sinir ağının eğitilmesinde önceden eğitilmiş olan bir modelin kullanılması, ağı performansını artırır, ağı görülmemiş veri setlerinde genelleme yapma yeteneğini geliştirir, eğitim sürecini kısaltır, modelin yetersiz öğrenme problemini ortadan kaldırır ve küçük veri setlerinde oluşabilecek olan aşırı uyumu önler[1].

Öğrenme transferi tekniğinde bilgilerin aktarılma yöntemi örnek tabanlı, özellik tabanlı ve parametre tabanlı şeklinde kategorilere ayrılabilir[114, 115]. Örnek tabanlı öğrenme transferinde, öğrenme sürecine yardımcı olmak için kaynak modeldeki örnekler yeniden kullanılır, eğitim sürecinde ağırlıklar otomatik olarak öğrenilir[114-116]. Kaynak modeldeki alakalı örnekler seçilerek bu örneklerin ağırlıkları tespit edilir. Özellik tabanlı öğrenme transferinde, kaynak modeldeki ortak özellikler yeni modele özellik dönüşümü yapılarak aktarılır. Son olarak parametre tabanlı öğrenme transferinde ise yeni model kaynak modelin ağırlık parametreleri aktararak başlatılır.

Sonuç olarak öğrenme transferi, sınırlı veri setine sahip bir modelin geliştirilmesine yardımcı olarak ağırlıklı görülmemiş veri setlerinde genelleme yapabilmesine, eğitim sürecinin kısa süre içerisinde tamamlanabilmesine, aşırı uyum ve yetersiz öğrenme problemlerinin önlenmesine katkıda bulunur[1, 117-122].

2.8.8. Sınıf Dengesizliği Çözüm Teknikleri

Derin öğrenme sinir ağlarında, farklı sınıflardaki örnek sayılarının dağılımı yüksek oranda farklılık gösteriyor ise sınıf dengesizliği problemi ortaya çıkar. Sınıf dengesizliği, derin öğrenme modeli eğitilirken istenmeyen sonuçların oluşmasına sebebiyet verebilir. Model çok örnek sahip olan sınıfa yönelim gösterirken azınlık sınıflarını ihmal edebilir. Eğitim sırasında model, doğrulama verileri üzerinde ölçüm değerlendirmesi yaparken azınlık sınıfındaki örnekleri yanlış tahminde bulunması halinde duyarlılık ve kesinlik ölçüm değerleri bu durumdan kötü yönde etkilenebilir[123-125].

Sınıf dengesizliği, geri yayılım algoritmaları üzerinde önemli bir etkiye sahiptir [126]. Modelin ağırlıklarının çoğunluk sınıfı tarafından baskın bir şekilde güncellenmesi, eğitimde çoğunluk sınıfının hatalarını hızlı bir şekilde azaltırken azınlık sınıfının hatalarını artırır. Bu durum ağırlık yavaş yakınsamasına sebep olur [127, 128].

Sınıf dengesizliği probleminin çözümü için çok sayıda yaklaşım önerilmiştir.

a) Aşırı ve yetersiz örnekleme

Aşırı örnekleme tekniğinde, veri artırma teknikleri kullanılarak azınlık sınıfının örnekleri artırılırken yetersiz örnekleme tekniğinde ise çoğunluk sınıfının örnekleri azaltılarak sınıfların daha dengeli olması sağlanır[129, 130]. Bu sayede

modelin çoğunluk sınıfına sapması azaltılır. Ancak bu teknikler eğitim sürecini artırır ve aşırı uyum sorununa da neden olabilir.

b) Kayıp tabanlı dengeleme

Bu yaklaşımda, kayıp fonksiyonları tekrar düzenlenerek modelin azınlık sınıfına daha fazla önem vermesi sağlanır. Kayıp fonksiyonlarında, yanlış tahminde bulunan azınlık sınıfı kayıplarına daha yüksek ceza puanı eklenerek modelin azınlık sınıfını daha iyi tahmin edebilmesi sağlanır. Sınıf dengesizliğine sahip veri setleri üzerinde model geliştirilirken hatalı yanlışların ortalaması kaybı(MFE)[80], hatalı yanlışların karelerinin ortalaması kaybı(MSFE)[80], odak kaybı[76], ağırlıklı çapraz entropi gibi sınıf dengesizliğine duyarlı kayıp fonksiyonlarının kullanılması modelin performansının artmasına yardımcı olabilir.

c) Uygun ölçüm tekniklerini kullanma

Sınıf dengesizliğine sahip veri setlerini eğitirken, kesinlik, duyarlılık, F1 skoru ve eğri altındaki alan(AUC) ölçümü gibi tekniklerin kullanılması daha uygun olacaktır.

2.8.9. Sinir Ağı Eğitilirken Dikkat Edilmesi Gereken Konular

a) Model karmaşıklığı

Evrişim sinir ağı eğitilirken ağın mimari yapısının optimum bir dengede olması gerekir. Eğer model çok basit bir mimari yapıya sahip olursa model eğitim verileri üzerinde yeterince öğrenme yapamaz ve bu durum yetersiz öğrenme problemini ortaya çıkmasına neden olur. Modelin karmaşıklık yapısını artırmak için evrişimsel gibi katmanların sayısı daha fazla artırılabilir, katmanlarda daha fazla çıktı üretmek için daha fazla filtre kullanılabilir. Tamamen bağlı katman sayısı artırılabilir ve katmanlardaki nöron sayısını artırmak için daha büyük boyutlu ağırlık matrisleri kullanılabilir. Eğer model çok karmaşık bir yapıya sahip ise de, eğitim verileri üzerinde yüksek doğruluğa sahip olur ancak görülmemiş test verilerinde genelleme yeteneğini gösteremez. Karmaşık bir mimariye sahip modeller eğitim verilerinin özelliklerini daha fazla öğrenerek genelleme yeteneğini kaybedebilirler. Sonuç olarak veri setinin büyüklüğüne göre modelin karmaşık yapısının optimum bir dengede olması, modelin genelleme yapma yeteneğini iyileştirmeye yardımcı olabilir.

b) Öğrenme oranı

Evrişim sinir ağlarında öğrenme oranı önemli bir parametre olup öğrenme oranının iyi ayarlanması ağın daha iyi yakınsamasına olanak sağlar. Yüksek öğrenme oranı ağın performansını kötü yönde etkileyerek optimum değerler atlanabilir ve ağın yakınsamasında başarısız sonuçlar verebilir. Eğitim ve doğrulama kayıpları değerlerinde sert dalgalanmaların oluşumuna sebebiyet verir. Diğer taraftan çok düşük öğrenme oranı seçimi, ağın yakınsamasını yavaşlatarak eğitim sürecini uzatır.

Optimum öğrenme oranını tespit etmek için öğrenme oranını uyarlayıcı Adam, rmsPROP ve SGD gibi optimizasyon algoritmaları seçilmelidir.

c) Toplu veri setlerinin boyutunun(batch-size) iyi ayarlanması

Evrişim sinir ağlarında eğitimde kullanılan küçük veri setlerinin boyutu öğrenim sürecini, ağın yakınsamasını, hesaplama maliyetini ve modelin performansını etkileyen önemli bir parametredir. Veri setlerinin(batch-size) daha küçük olması gradyanların gürültülü olmasına ve bunun sonucu olarak gradyanların daha fazla güncellenmesini gerektirebilir. Bu durum ağın eğitim sürecini uzattığı gibi başlangıçta ağın hızlı yakınsamasına neden olabilir. Ancak küçük veri setleri daha az hafıza gerektirir. Dolayısıyla kısıtlı hafızaya sahip GPU' larda rahat eğitim yapılabilir. Diğer taraftan veri setlerinin daha büyük olması, gradyanların daha doğru ve daha az gürültülü olmasını sağlar ve bunun sonucu gradyanlar daha az güncellemeye ihtiyaç duyar. Optimizasyon sürecini yoluna koyarak ağın daha istrarlı olmasına yardımcı olur. Verilerin eş zamanlı işlenmesini sağlayarak eğitim sürecini kısaltır. Daha çok hafıza gerektirdiğinden kısıtlı hafızaya sahip GPU' larda eğitilmesi zorlaşır.

d) Ağırlıkların doğru başlatılması

Ağırlıkların doğru teknikler kullanılarak başlatılması eğitim süresi boyunca modelin daha hızlı yakınsamasına, patlayan ve yok olan gradyan problemlerinin çözümüne yardımcı olur. Ağırlıklar çok küçük değerler ile başlatılır ise yok olan gradyan, büyük değerler ile başlatılırsa patlayan gradyan probleminin oluşumuna sebebiyet verebilir.

e) Veri çözünürlüğü

Yüksek çözünürlüklü veriler daha fazla ayrıntıya, özelliğe ve bilgiye sahip olduğundan model eğitilirken yüksek çözünürlüklü verilerin kullanılması, ağın

performansını artırır, küçük ayrıntıların öğrenilerek ağı daha hızlı yakınsamasına ve genelleme yapmasına yardımcı olabilir. Ancak, yüksek çözünürlüklü veriler ile modelin eğitilmesi daha fazla hesaplama maliyeti ve hafıza gerektirir. Yeterli miktarda veri yok ise aşırı uyum probleminin oluşmasına sebebiyet verebilir. Daha fazla bilginin öğrenebilmesi için ağı mimari yapısında daha çok katman ve filtre boyutları gerektirebilir. Aksi takdirde yüksek çözünürlüklü verilerdeki bilgilerin öğrenilmesi zorlaşır.

f) Eğitim Döngüsü

Sinir ağı modelinin eğitimi için en uygun eğitim döngüsünü belirlemek, modelin aşırı uyumunu ya da yetersiz öğrenmesini önleyerek modelin en uygun döngüde yakınsamasını sağlar. Çok sayıda eğitim döngüsü modelin aşırı uyum göstermesine, az sayıda eğitim döngüsü modelin veri setini yeteri kadar öğrenememesine neden olabilir. Eğitim sürerken doğrulama kayıpları izlenerek yakından takip edilmeli, doğrulama kayıplarının artış göstermeye başladığı aralıklarda modelin genelleme yeteneğini kaybetmesini engellemek için en uygun döngüde erken durdurma işlemi yapılarak modelin performansının düşmesi engellenmelidir.

2.9. CNN Değerlendirme Metrikleri

Evrişim sinir ağlarında değerlendirme metrikleri eğitim aşamasında modelin performansını değerlendirir. Tahmin edilen değerler gerçek değerler ile karşılaştırılarak modelin tahminleme işleminde ne kadar başarılı olduğu ölçülür. Bazı metrikler sınıf dengesizliği problemini ele alarak azınlık sınıfının performansına odaklanır. Her bir metrik farklı problemleri değerlendirir, modelin hangi konularda başarısız olduğu tespit edilir ve modelin zayıflığına neden olan problemlere yönelinmesine katkı sağlar[1, 69].

2.9.1. Hata Matrisi

Hata Matrisi (Confusing Matrix)				
		TAHMİN		TOPLAM
		YOK	VAR	
GERÇEK	YOK	TN	FP	
	VAR	FN	TP	
	TOPLAM			

Şekil 2.10 : Hata matrisi

Hata matrisi(Confusion Matrix) sınıflandırma modellerin performansını ölçmek için gerçek değerler ile tahmin edilen değerlerin karşılaştırılması için kullanılır. Bu matrisdeki değerler:

- a) TP : Gerçekte var olan bir örneğin doğru tahmin edilmesi
- b) TN: Gerçekte yok olan bir örneğin yok olarak doğru tahmin edilmesi
- c) FP : Gerçekte yok olan bir örneğin var olarak yanlış tahmin edilmesi
- d) FN: Gerçekte var olan bir örneğin yanlış tahmin edilmesi

Hata matrisindeki TN, FP, FN ve TP değerleri kullanılarak modelin doğruluğu, kesinliği, duyarlılığı ve F1 skoru gibi değerler hesaplanabilir.

2.9.2. Doğruluk (Accuracy)

Doğruluk ölçümü nesne sınıflandırmasında en yaygın kullanılan metriktir. Doğru tahmin edilen örnek sayısının, toplam örnek sayısına bölümüyle elde edilir. Sınıf dağılımı dengeli olduğunda doğru sınıflandırılmış örneklerin orantısını ölçer. Doğruluk ölçümünün hesaplanması denklem 2.61' de tanımlanmıştır.

$$Doğruluk = \frac{\text{Doğru tahmin edilen örnek sayısı}}{\text{Toplam örnek sayısı}} \Rightarrow Doğruluk = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.61)$$

2.9.3. Kesinlik (Precision)

Kesinlik, pozitif tahminlerin doğruluğunu ölçer. Doğru pozitif tahminlerin sayısının, doğru ve yanlış pozitif tahminlerin sayısına bölümü ile elde edilir. Kesinlik değerinin yüksek olması, modelin yanlış pozitif üretmediği, daha fazla doğru pozitif ürettiği anlamına gelir. Kesinlik ölçümünün hesaplanması denklem 2.62' de tanımlanmıştır.

$$Kesinlik (Precision) = \frac{TP}{TP+FP} \quad (2.62)$$

2.9.4. Duyarlılık (Recall)

Gerçek pozitif oran (TPR) olarak da bilinen bu ölçüm, veri kümesindeki örnekleri ne kadar iyi tanımlayabileceğini ölçer. Yüksek duyarlılık değeri, modelin daha az sayıda yanlış negatifi ürettiğini ve var olan örnekleri daha doğru şekilde tahmin ettiği anlamına gelir. Duyarlılık ölçümünün hesaplanması denklem 2.63' de tanımlanmıştır.

$$Duyarlılık (Recall) = \frac{TP}{TP+FN} \quad (2.63)$$

2.9.5. F1 Skoru

F1 skoru, kesinlik ve duyarlılık ölçümlerini birleştirip sınıflandırma modelinin tüm performansını genelini temsil eden tek bir değer sağlar [69, 131]. Kesinlik ve duyarlılık ölçümlerinin harmonik ortalamasını hesaplar. Veri setinin dengesiz olduğu durumlarda kesinlik ve duyarlılık ölçümleri için bir denge oluşturmasını sağlar. Yüksek F1 skoru bu dengenin iyi olduğunu gösterir. F1 skorunun hesaplanması denklem 2.64' de tanımlanmıştır.

$$F1 Skoru = 2 * \frac{Kesinlik * Duyarlilik}{Kesinlik + Duyarlilik} \quad (2.64)$$

2.9.6. F2 skoru

Modelin performansını ölçmek için kullanılan bir diğer metrik olan F2 skoru, F1 skorunun değiştirilmiş bir versiyonudur ve duyarlılık ölçümünün kesinlik ölçümünden daha önemli olduğu durumlarda kullanılır. F2 skorunun hesaplanması denklem 2.65' de tanımlanmıştır.

$$F2 Skoru = \frac{5 * Kesinlik * Duyarlilik}{4 * Kesinlik + Duyarlilik} \quad (2.65)$$

2.9.7. Gerçek Negatif Oran(TNR)

TNR, model tarafından doğru tahmin edilmiş gerçek negatiflerin oranını ölçen bir ölçümdür. Gerçek negatif oranının hesaplanması denklem 2.66' da tanımlanmıştır.

$$TNR = \frac{TN}{TN + FP} \quad (2.66)$$

2.9.8. Yanlış Pozitif Oran (FPR)

FPR, pozitif gibi yanlış sınıflandırılan gerçek negatif örneklerin orantısını hesaplar. Negatif tahminlemede yüksek doğruluk gerektiren uygulamalarda kritik öneme sahip olan bir ölçüm tekniğidir. Yanlış pozitif oranının hesaplanması denklem 2.67' de tanımlanmıştır.

$$Yanlış Pozitif Oran(FPR) = \frac{FP}{FP + TN} \quad (2.67)$$

2.9.9. Negatif Tahmin Değeri(NPV)

Negatif tahmin değeri ölçüm tekniği, doğru şekilde tanımlanan negatif örneklerin oranını ölçer [132]. Sınıf dengesizliği olduğu durumlarda kullanılması yararlıdır. Negatif tahmin değerinin hesaplanması denklem 2.68' de tanımlanmıştır.

$$Negatif Tahmin Değeri(NPV) = \frac{TN}{TN+FN} = 1 - FPR \quad (2.68)$$

2.9.10. Yanlış Keşif Oranı(FDR)

Yanlış keşif oranı ölçümü, tüm olumlu tahminler arasında yanlış pozitiflerin oranını ölçer. FDR ölçümünün düşük olması modelin daha doğru tahminde bulunduğunu, yanlış tahminlerin daha az olduğunu gösterir. Yanlış keşif oranının hesaplanması denklem 2.69' da tanımlanmıştır.

$$Yanlış Keşif Oranı(FDR) = \frac{FP}{TP+FP} \quad (2.69)$$

3. NESNE TESPİTİ

Nesne tespiti, bilgisayarlı görme uygulamalarında kullanan en önemli uygulamalarından biridir. Nesne tespiti önceleri geleneksel yöntemlerle gerçekleştirilirken, son yıllarda derin öğrenme tekniklerin gelişmesi ile nesne tespiti derin öğrenme metotları kullanılarak gerçekleştirilmeye başlanmıştır. Nesne tespiti, sınıflandırma ve yerleştirme olmak üzere iki aşamalı bir süreçten oluşmaktadır. Sınıflandırma, görüntü veya görüntülerin tanımlanarak sınıflandırılma sürecidir. Yerleştirme ise, verilen bir görüntü üzerindeki belirli bir nesnenin veya nesnelerin tespit edilip bulunan nesnelerin etrafına bir sınırlayıcı kutu çizildiği bir süreçtir [133]. Nesne algılama işlemi, görüntüyü tanıma işleminden farklıdır. Görüntü tanıma işleminde görüntüye sadece etiket ataması yapılırken, görüntü üzerindeki nesnelerin tespiti işleminde ise nesneler görüntü üzerinde yerleştirilir, sonrasında nesnenin etrafına bir sınırlayıcı kutu(bounding-box) çizilir ve sonrasında ise nesneye bir etiket ataması yapılır. Yıllar boyunca nesne tespiti alanında çok sayıda araştırma yapılmıştır. Kalabalıklardaki insanları saymak, özel ve yüksek güvenli alanlara izinsiz giriş tespiti, plaka tanıma, araç tanıma, araçları sınıflandırma, hayvanları algılama gibi birçok gerçek hayat alanlarındaki uygulamaların geliştirilmesinde nesne tespiti algoritmaları kullanılmıştır. Nesne tespiti modellerindeki öncelikli hedef, bir giriş görüntüsündeki nesne veya nesnelerin yerini belirlemek ve sonrasında bu nesneleri sınıflandırmaktır. Önceleri geleneksel yöntemlerle nesne tespiti yapılırken son yıllarda derin öğrenme teknikleri ile nesne tespitinin yapılması yaygınlaşmaya başlamıştır.

3.1. Geleneksel Yöntemlerle Nesne Tespitinin Yapılması

Geleneksel nesne algılama metotlarında nesne tespiti üç adımda gerçekleştirilir[7-9]. İlk adımda, görüntünün tamamı incelenip görüntüden bölge önerileri üretilmektedir. Bölge önerilerini üretmek için seçici arama(selective search) algoritması[10] kullanılır. Bu bölge önerileri her bir görüntüde bir kaç bin tane olabilmektedir. İkinci adımda ise, özellik çıkartımı işlemidir. Üretilen bölge önerileri analiz edilerek özellik çıkartımı veya görüntü kalıpları elde edilir. Geleneksel nesne algılama teknikleriyle bu özelliklerin tasarımı algoritmanın performansı açısından oldukça önemlidir. Özellik çıkartımı ve bölge önerileri üretimi için aşağıdaki algoritmalar kullanılır.

3.1.1. Yönlü Gradyan Histogramı

Yönlü gradyan histogramı (Histogram of Oriented Gradient - HOG) [11], görüntü işleme ve bilgisayarlı görme alanında, görüntüden özellik çıkartımı yapabilmek için kullanılan bir algoritmadır. Görüntüdeki nesnelerin sınırlarını ve kenarlarını tespit etmek için gradyan bilgilerini yakalayıp yönlü gradyan histogramını hesaplar. HOG, yayaların tespitinde yaygın olarak kullanılmıştır.

3.1.2. Haar Benzeri Özellikler

Haar benzeri özellikler (Haar-Like Features) algoritması, nesne tespitinde kullanılan basit dikdörtgen filtrelerdir [134]. Bu özellikler, görüntünün belirli bölgelerinde değişim yoğunluğunu yakalar. Gerçek zamanlı uygulamalarda hesaplama maliyeti açısından verimlidir [135]. Viola-Jones algoritması yüz tanıma sistemlerinde kullanılan en hızlı algoritmalarından biri olmuştur.

3.1.3. Özellik Eşleştirme

Ölçekle değişmeyen özellik dönüşümü(SIFT) [12] ve hızlandırılmış sağlam özellik(SURF) [13] gibi özellik eşleştirme algoritmaları, görüntüdeki ayırt edici yerel özellikleri tespit eder ve görüntü üzerinde bu özellikleri eşleştirirerek nesne tespiti işlemi yapılır.

3.1.4. İkili Sağlam Bağımsız Temel Özellikler

İkili sağlam bağımsız temel özellikler (BRIEF) [136] algoritması, görüntüyü hızlı bir şekilde analiz edip eşleştirmeyi hızlı bir şekilde yapabilme yeteneğine sahiptir. Diğer algoritmalarla göre performans açısından daha etkilidir.

3.1.5. Seçici Arama Algoritması

Seçici arama(Selective Search) [10] algoritması, görüntüyü analiz edip görüntü içerisinde nesne içerebilecek olan tüm bölge önerilerini üretir. Nesne bölgelerini saptamak için görüntüyü renk, metin ve boyut tabanlı bölümlleme yapar.

Sayıdığımız bu metotlar ile görüntü üzerindeki nesnelerden özellik çıkarımı yapıp görüntü üzerinde bu çıkarılan özellikler eşleştirilerek görüntüdeki nesneler tespit edilmeye çalışılmaktadır. Son adımda da, bir nesne içerip içermediğine bakılmaksızın bu varlıkların sınıflandırma algoritmaları kullanılarak sınıflandırılmasından oluşur. Sınıflandırma işleminde, Destek Vektör Makinası(Support Vector Machine - SVM)[14], Uyarlanabilir Güçlendirme(Adaboost) [137], Deforme Olabilen Parça Tabanlı Model (DPM) [134] ve K-En Yakın Komşular (K-Nearest Neighbors)[138] gibi sınıflandırma algoritmaları kullanılmaktadır. Geleneksel tekniklerle nesne tespiti yapılması işleminde bütün görüntü taranır, değişken olan görüntülerde nesneleri görüntü üzerinde yerleştirmek hesaplama maliyeti açısından oldukça masraflı bir yöntemdir.

3.2. Derin Öğrenme Yöntemleri İle Nesne Tespitinin Yapılması

Nesne algılamada son yıllarda yaygın olarak kullanılan diğer bir yöntem ise derin öğrenme teknikleri ile gerçekleştirilen metotlardır.

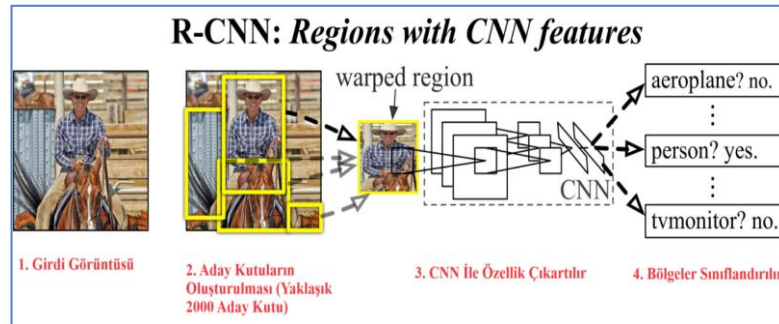
Derin öğrenme tabanlı algoritmalar iki aşamalı ve tek aşamalı olmak üzere iki bölüme ayrılır. İki aşamalı algoritmalar bölgesel tabanlı olup önce bölge önerileri üretilir, üretilen bölge önerilerinden CNN ile özellik çıkarımı yapılır, daha sonra bu özellikler CNN veya SVM algoritmaların yardımı ile sınıflandırılır. Bölgesel tabanlı algoritmalarda her bir görüntüden üretilen bölge önerileri yaklaşık 2000 adet olabilmekte ve çok sayıda bölge önerisinin CNN ile özellik çıkarımı ve bu özelliklerin kategorilere ayrılması hesaplama maliyeti açısından pahalı bir yöntem olup gerçek zamanlı uygulamalarda çok verimli olamamaktadır. Tek aşamalı algoritmalarda ise görüntü ızgaralara bölünür ve sonra her bir ızgara hücresinde CNN yardımı ile bir nesne içerip içermediğinde bakılır. Daha sonra IoU metotları ile bu ızgara hücreler birleştirilerek görüntü üzerindeki nesnelerin tespiti yapılmaktadır. Tek aşamalı nesne algılama algoritmalarından hız ve doğruluk ölçümü açısından yüksek performanslar elde edilmeye başlamasından sonra son yıllarda gerçek zamanlı uygulamalarda kullanımı giderek yaygınlaşmaya başlamıştır.

3.2.1. İki Aşamalı Nesne Algılama Metotları

Bölge tabanlı nesne tespit algoritmaları iki aşamalı olup görüntülerdeki nesneleri tespit etmede kullanılan en yaygın teknikler arasında bulunmaktadır. Nesne tespitinde ilk önce görüntü üzerinde bölgeler aranır, CNN ile özellik çıkarımı yapılır, SVM gibi algoritmalar ile sınıflandırma işlemi gerçekleştirilir [7]. İki aşamalı nesne algılama metotları, R-CNN algoritmasından türetilmiştir. R-CNN yöntemlerinde, seçici bir arama yöntemi kullanılarak[10] ilgili bölgeler algılanır, sonra bu bölgelerden özellik çıkarımı yapılarak sınıflandırılır.

3.2.1.1. Bölgesel Bazlı Evrimsel Sinir Ağı (R-CNN)

R.Girshick v.d.[15] tarafından önerilmiş olan R-CNN algoritmasında görüntü üzerindeki nesneler lokalize edilerek tespit edilir[7, 139]. Bu teknikte geleneksel yöntem olan bölge önerileri yaklaşımı ile CNN algoritması birleştirilerek görüntü üzerindeki nesneler tespit edilmeye çalışılmıştır. R-CNN algoritması birkaç aşamadan oluşur. İlk aşamada, geleneksel yöntemlerdeki gibi görüntü üzerinden bölge önerileri üretilir. Bölge önerilerinin üretilmesi Uijlings J.R.R. v.d.[10] tarafından geliştirilmiş olan seçici arama algoritma yardımıyla yapılır. Üretilen bu bölge önerilerinde nesne içerme olasılığı yüksektir. Bir görüntüde, bölge önerilerinin sayısı yaklaşık 2000 adet olabilmektedir. İkinci aşamada, üretilen bu bölge önerilerinden CNN yardımı ile özellik haritalarının elde edilmesi sağlanır. Üçüncü aşamada ise, elde edilen bu özellik haritaları destek vektör makinalarının(SVM) yardımı ile sınıflandırılır ve daha sonra bu bölge önerileri görüntü üzerinde yerleştirilerek koordinatları tahmin edilir. Aynı zamanda aynı bölgede birden fazla bölge önerisi örtüşüyor ise maksimum olmayan bastırma(NMS) tekniği kullanılarak aynı bölgede örtüşen bölge önerileri elenir. Şekil 3.1’ de R-CNN algoritmasının mimarisi görülmektedir.



Şekil 3.1 : R-CNN mimarisi

3.2.1.2. Uzamsal Piramit Havuzlama Ağı (SPPNet)

Uzamsal piramit eşleştirme (SPM)[140] konseptine dayanılarak geliştirilen SPPNet[141], temel olarak R-CNN algoritmasının geliştirilmiş versiyonu olup nesne tespitindeki verimliliği iyileştirmek için tasarlanmış bir metottur[7, 141]. Bu algoritmada, girdi görüntüleri farklı boyutlarda olabilir. Havuzlama katmanında SPP konseptinden faydalanılmıştır. İlk adımda, girdi görüntüleri evrişim işleminden geçirilerek girdi görüntülerinin boyutlarına bağlı olarak özellik haritaları elde edilir[141]. İkinci adımda, bu özellik haritaları SPP katmanında farklı seviyeli filtrelerle havuzlama işleminden geçirilir ve elde edilen bu farklı boyutlu özellik haritaları tek bir özellik vektörüne dönüştürülür. Üçüncü adımda ise, sabit boyutlu bu özellik vektörleri tamamen bağlı katmana aktarılır, nesneler ve bu nesnelerin etrafını çevreleyen sınırlayıcı kutular için model tahminler üretir.

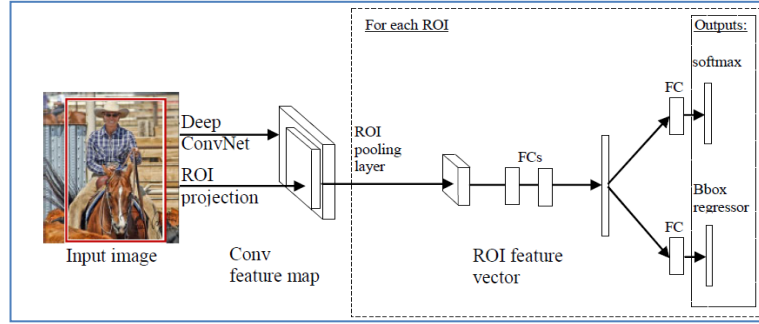
SPPNet, evrişim katmanında R-CNN algoritmasından farklı bir yaklaşım kullanır. R-CNN algoritmasında seçici arama algoritması kullanılarak görüntüden yaklaşık 2000 tane bölge önerisi üretilir ve bölge önerileri CNN ile işlenerek özellik haritaları oluşturulur. Diğer taraftan SPPNet ise evrişim katmanında görüntüyle sadece bir kez ilgilenir[142]. Dolayısıyla, SPPNet R-CNN'den çok daha hızlı ve daha doğru sonuçlar verebilmektedir.

3.2.1.3. Hızlı R-CNN

R-CNN algoritmasındaki sorunları iyileştirmek, hız ve doğruluğu artırmak için R.Girshick 2015 de hızlı R-CNN adında yeni bir algoritma sunmuştur[16]. Hızlı R-CNN algoritmasında özellik çıkarımı ve bölge önerisi adımları birleştirilmiş, algoritmaya yeni bir RoI havuzlama katmanı eklenmiştir. Bu algoritmada R-CNN' de sınıflandırma için kullanılan SVM sınıflandırıcıları kaldırılarak regresyon ve sınıflandırma yapmak için tamamen bağlı katman eklenmiştir[139]. Modelde, AlexNet[6] yerine VGGNet[143] kullanılmıştır. Hızlı R-CNN mimarisi şekil 3.2' de gösterilmektedir.

Bu algoritmanın ilk adımında, girdi görüntüleri evrişimsel ve havuzlama katmanında işlenerek özellik haritalarının elde edilmesi sağlanır. İkinci adımda, RoI havuzlama katmanında her bir bölge önerisi için bu özellik haritalarından sabit boyutlu özellik vektörleri üretilir. Üçüncü adımda ise, üretilen sabit boyutlu bu vektörler tamamen bağlı katmana iletilerek softmax fonksiyonu ile sınıflandırılır, regresyon algoritmalarıyla da bu nesnelerin koordinatlar tespit edilir. Ayrıca

maksimum olmayan bastırma(NMS) tekniği kullanılarak örtüşen sınırlayıcı kutular elenir. Regresyon kaybını hesaplamak için L1 veya L2 kayıp fonksiyonları kullanılır[16].



Şekil 3.2 : Hızlı R-CNN Mimarisi

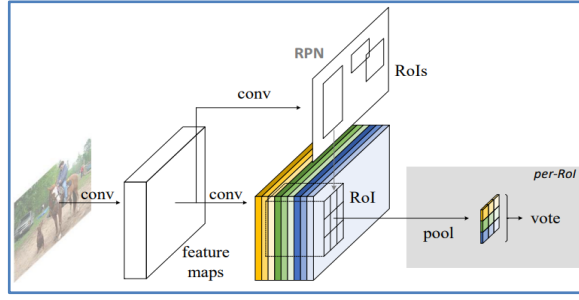
3.2.1.4. Daha Hızlı R-CNN

Hızlı R-CNN algoritmasının iyileştirilmiş bir sürümü olan daha hızlı R-CNN[17] algoritmasında bölge öneri ağı(RPN) tanıtılmıştır. Algoritmanın ilk adımında, girdi görüntüleri evrimleştirilerek özellik haritaları elde edilir. İkinci adımda, bölge önerisi ağında bu özellik haritalarından bölge önerileri tahmin edilerek bağlantı kutular ile ilişkili olan sınırlayıcı kutular üretilir. Daha sonra RoI havuzlama katmanında her bir bölge önerisi için evrimsel özellik haritalarından sabit boyutlu özellik haritaları elde edilir. Üçüncü adımda, bu sabit boyutlu özellik haritaları düzleştirilerek tamamen bağlı katmana iletilir. Bu katmanda bu özellikler sınıflandırılır ve regresyon kaybı algoritmaları ile sınırlayıcı kutuların yerleştirilmesi yapılır. Aynı zamanda aynı bölgede örtüşen birden fazla sınırlayıcı kutular maksimum olmayan bastırma tekniği ile elenir[7, 17, 139, 142].

3.2.1.5. Bölge Tabanlı Tamamen Evrimsel Ağ(R-FCN)

Hızlı R-CNN veya Daha Hızlı R-CNN gibi bölgesel tabanlı algoritmalarda nesneler iki aşamada tespit edilir. R-FCN[144], daha hızlı R-CNN algoritmasının geliştirilmiş bir sürümüdür. R-FCN algoritmasında, bölge önerisi üretimi ile nesne tespit işlemi tamamen evrimsel sinir ağı içerisinde birleştirilerek hem nesnenin tespiti hem de nesnenin yerleştirilmesi eş zamanlı olarak yapılır. Şekil 3.4’ de R-FCN mimarisi görülmektedir.

Bu algoritmada, girdi görüntüleri ResNet-101 ile işlenerek evrimsel özellik haritaları üretilir. Bu özellik haritaları ızgara hücrelerine bölünerek konuma duyarlı skor haritaları üretilir. Her bir ızgara hücresi sınıf skorlarını tahmin edererek doğru sınıfları tahmin etmesi sağlanır.



Şekil 3.3 : R-FCN Mimarisi

Konuma duyarlı skorlar üretildikten sonra, her bir bölge önerisi için evrişimsel özellik haritalarından özelliklerin elde edilmesi için konuma duyarlı RoI havuzlama işlemi gerçekleştirilerek sınırlayıcı kutu tahminleri üretilir. Daha sonra tamamen bağlı katmanda, konuma duyarlı bu skorlar kullanılarak nesnelerin sınıflandırılması ve sınırlayıcı kutuların yerleştirilmesi yapılır.

3.2.2. Tek Aşamalı Nesne Algılama Metotları

Bölge önerisine dayanan R-CNN gibi algoritmalar bölge önerisinin oluşturulması, CNN ile özellik haritalarının elde edilmesi, bu özellik haritalarından sınıflandırma ve bu özelliklerin görüntü üzerinde yerleştirilmesi gibi birkaç aşamadan oluşmaktadır ve ayrı bir şekilde eğitim yapılmaktadır[142]. İki aşamalı algoritmalar karmaşık bir mimariye sahip olduğundan görüntü üzerindeki nesnelerin tespit edilmesi maliyetli bir hesaplama gerektirir. Dolayısıyla bu algoritmalar gerçek zamanlı uygulamalarda çok tercih edilmez. Tek aşamalı nesne tespit algoritmalarında ise öncelikle hesaplama ve çalışma verimliliği öne çıkmaktadır. Bu modellerde sınırlayıcı kutular doğrudan sınıflandırılır ve görüntü üzerinde yerleştirilir [7].

3.2.2.1. SSD

Tek atış çoklu kutu dedektörü(SSD), R-CNN ve Yolo modellerinden sonra 2016 yılında W. Liu [145] tarafından geliştirilen bir algoritmadır. Bölge tabanlı nesne tespit algoritmaları nesne tespitini iki aşamada gerçekleştirirken, SSD algoritması hem nesnelerin sınıflandırılmasını hem de nesnelerin görüntü üzerinde yerleştirilmesini ileri yayılım ile tek seferde gerçekleştirir. SSD farklı boyutlardaki nesneleri tespit etmek için bir dizi farklı ölçekli sınırlayıcı kutu kullanır. SSD modelinde VGA-16 [143] CNN ağı kullanılmıştır. Sınıf kaybını hesaplamak için softmax çapraz entropi kayıp fonksiyonu, regresyon kaybının hesaplanmasında ise L1 kayıp fonksiyonu kullanılmıştır[7].

Bu yöntemde girdi görüntüleri öncelikle CNN algoritması ile evrimleştirilerek özellik haritaları elde edilir. Daha sonra bu özellik haritaları FPN gibi teknikler kullanılarak farklı boyutlarda ölçeklendirilir. SSD algoritmasında evrimsel katmanlar kullanılarak özellik haritalarının sınıfı ve sınırlayıcı kutular görüntü üzerinde tahmin edilir. Maksimum olmayan bastırma(NMS) tekniği ile aynı bölgede örtüşen sınırlayıcı kutular elenir.

3.2.2.2. YOLOv1

YOLOv1 modeli[18] J.Redmon tarafından 2016 yılında geliştirilmiş olup görüntü üzerindeki nesneleri tespit etmek için farklı bir yaklaşım sunulmuştur. Bölge tabanlı hızlı R-CNN modelinin aksine[146], görüntünün tamamını tek bir seferde ele alarak hem nesnelerin sınıflandırılmasını hem de bu nesnelerin görüntü üzerinde yerleştirilmesini tek bir aşamada gerçekleştirir.

YOLOv1 algoritması evrimsel ve tamamen bağlı katmanlardan oluşur. Girdi görüntüsü üzerindeki nesnelerin tespitini gerçekleştirmek için öncelikle girdi görüntüsü $S \times S$ şeklinde hücrelere bölünür. Bu hücreler genellikle 7×7 boyutunda olur. Her bir hücre içerisinde tek bir nesnenin olup olmadığı kontrol edilir. Ayrıca her bir hücre B sayıda sınırlayıcı kutuyu ve bu kutuların güven skorlarını tahmin etmekten sorumludur. Algoritmada GoogleNet[147] CNN mimarisi uygulanır. Ağ mimarisinde 24 adet evrimsel katman ve 2 adet tamamen bağlı katman bulunmaktadır. Son katmanda $S \times S \times (B \times 5 \times C)$ boyutlu tensör bulunur. B sınırlayıcı kutu sayısını, C ise sınıf sayısını belirtir. Her bir sınırlayıcı kutu 5 adet değişkene sahiptir. Bunlar: (x, y, w, h, skor). Burada x ve y sınırlayıcı kutunun koordinatlarını, w ve h sınırlayıcı kutunun genişliğini ve yüksekliğini, skor ise sınırlayıcı kutunun nesne içerip içermediği ve nesnenin ne kadar doğru olduğunu temsil etmektedir.

YOLOv1 yerleştirme, güven ve sınıf kaybı olmak üzere üç adet kayıp hesaplama fonksiyonu kullanır.

a) Yerleştirme Kaybı

Yerleştirme kaybında gerçek ve tahmin edilen sınırlayıcı kutu arasındaki farklar ölçülür. Yerleştirme kaybı hataların karelerinin toplamı alınarak bulunur. Yerleştirme kaybı, ızgara hücresi bir nesne ve tahmin edilen sınırlayıcı kutuyu içeriyor ise hesaplanır. Yerleştirme kaybı denklem 3.1' deki gibi hesaplanır.

$$L_{Yerelle\ ştirme} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - x_i^1)^2 + (y_i - y_i^1)^2 + (w_i - w_i^1)^2 + (h_i - h_i^1)^2] \quad (3.1)$$

Burada λ_{coord} koordinat kaybını ağırlıklandıran hiper bir parametreyi (genellikle değeri 5'dir), S hücre sayısını, B sınırlayıcı kutu sayısını, 1_{ij}^{obj} i-th hücresindeki j-th sınırlayıcı kutusunda nesne var ise 1 yok ise 0 olur. x_i, y_i, w_i, h_i tahmin edilen sınırlayıcı kutuların koordinatlarını, $x_i^1, y_i^1, w_i^1, h_i^1$ ise gerçek sınırlayıcı kutunun koordinatlarını temsil eder. İkinci olarak güven skorunun hesaplanmasıdır. Her bir ızgara hücresi içerisindeki sınırlayıcı kutular bir güven skoruna sahiptir.

b) Güven Skoru

YOLOv1 algoritması her bir hücre içerisindeki her bir sınırlayıcı kutu için güven skoru üretir. Nesne içeren veya içermeyen sınırlı kutular için güven skorları hesaplanır. Güven skorunun hesaplanması denklem 3.2' tanımlanmıştır.

$$L_{Güven\ Skoru} = \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} (C_i - C_i^1)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{noobj} (C_i - C_i^1)^2 \quad (3.2)$$

Burada λ_{noobj} nesnel olmayan güven kaybını ağırlıklandırmak için kullanılan bir hiper parametreyi (genellikle değeri 5'dir), S hücre sayısını, B sınırlayıcı kutu sayısını, 1_{ij}^{obj} i-th hücresindeki j-th sınırlayıcı kutusunda nesne var ise 1 yok ise 0 olur. C_i sınırlayıcı kutu için tahmin edilen güven skorunu, C_i^1 ise gerçek güven skorunu temsil eder. Nesne var ise $C_i^1=1$ nesne yok ise $C_i^1=0$ olur.

c) Sınıflandırma Kaybı

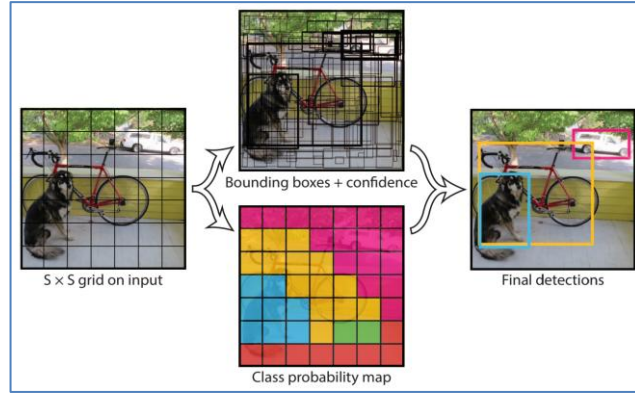
YOLOv1 her bir hücre için sınıf tahmini yapar. Eğer hücre nesne içeriyor ise sınıflandırma kaybı hesaplanır. Sınıflandırma kaybı denklem 3.3' de tanımlanmıştır.

$$L_{Sınıflandırma\ Kaybı} = \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes}^B (p_i(c) - p_i^1(c))^2 \quad (3.3)$$

Burada $p_i(c)$ i-th hücresindeki c sınıfı için tahmin edilen olasılığı, $p_i^1(c)$ sınıfın gerçek değerini (nesne c sınıfına ait ise 1 değilse 0 olur) temsil eder. toplam kayıp denklem 3.4' de tanımlanmıştır.

$$L_{Toplam} = L_{Yerelle\ ştirme} + L_{Güven\ Skoru} + L_{Sınıflandırma\ Kaybı} \quad (3.4)$$

Kayıplar hesaplandıktan sonra belirli bir eşik değerinin altında kalan sınırlayıcı kutular elenir. Aynı bölgeye örtüşen sınırlayıcı kutular maksimum bastırma tekniği ile elenir. Şekil 3.5' YOLOv1 modeli görülmektedir.



Şekil 3.4 : YOLOv1 modeli

Kayıplar hesaplandıktan sonra, yapılan her bir tahmine göre kayıpların gradyanları hesaplanarak geri yayılım ve optimizasyon algoritmaları ile bütün ağırlıklar tekrar güncellenir.

3.2.2.3. YOLOv2 ve YOLOv3

2017 yılında yayımlanan ve aynı zamanda YOLO9000 olarak adlandırılan YOLOv2[148] 9000'den fazla nesne sınıfını gerçek zamanlı olarak algılama kapasitesine sahip bir nesne tespit algoritmasıdır. İlk sürümündeki sorunları çözmek için çok sayıda güncellenmiş özelliklere sahiptir. Çok ölçekli eğitim modeli sunarak hız ve doğruluk arasında bir değiş tokuş yapıp değişik boyutlarda nesne tespitini yapabilmektedir. Yani küçük boyutlu modelde daha hızlı nesne tespiti yapılırken doğruluk düşmekte, büyük boyutlu modelde ise doğruluk artarken nesne tespit hızı düşmektedir. YOLOv2'de yerleştirme hataları ve düşük duyarlılığı geliştirilmesine odaklanılmış ve Darknet-19 mimarisi kullanılmıştır. Ağın yakınsamasını iyileştirmek için tüm evrişimsel katmanlara toplu normalleştirmeler eklenmiş, tamamen bağlı katmanlar kaldırılarak sınırlayıcı kutuları tahmin etmek için bağlantı kutuları kullanılmıştır. Bağlantı kutularının kullanılmasıyla modelin doğruluğunda az da olsa düşüşler olmuştur. Çıktı katmanında $S \times S \times (B \times (5 + C))$ boyutlu tensör bulunur. YOLOv2' de bağlantı kutularının kullanılmasıyla her bir ızgaradaki hücrede, model birden fazla sınırlayıcı kutuyu tahmin edebilmektedir. Dolayısıyla, YOLO' nun ilk sürümünde bir görüntüde 98 sınırlayıcı kutuyu tahmin edilirken, YOLOv2 de 1000 den fazla sınırlayıcı kutu tahmin edilebilmiştir. Gerçek pozitiflerin kaybının hesaplanmasından nesne için her bir sınırlayıcı kutu sorumludur. Bu nedenle en yüksek IoU seçilir [7].

YOLOv3[149] algoritması YOLOv2' nin iyileştirilmiş bir sürümüdür. Omurga katmanında Darknet-19 yerine Darknet-53 ve sızdıran ReLU aktivasyon

fonksiyonu kullanılmıştır. Darknet-53, 53 evrimsel katmana sahiptir. Girdi görüntüsünün boyutlarını 32, 16, 8 şeklinde alt örnekleme yaparak üç ölçekli tahmin gerçekleştirir[7]. Sınıf tahminlerini ve kaybı hesaplamak için softmax ve MSE fonksiyonun yerine ikili çapraz-entropi fonksiyonunu kullanılmıştır.

YOLOv3, her ölçekte 3 tane olmak üzere toplam olarak 9 bağlantı kutusu kullanılmaktadır. 9 bağlantı kutusunu oluşturmak için K-Means kümelemesine güvenilir. YOLOv3, YOLOv2'den daha fazla sınırlayıcı kutuya sahiptir. 416×416 çözünürlüğündeki bir görüntü için YOLOv2 her bir ızgara hücresinde $13 \times 13 \times 5 = 845$ kutuya sahiptir. 5 bağlantı kutunun kullanımı ile her ızgara hücresinde 5 kutu tespit edilmektedir. YOLOv3 ise, 3 ayrı ölçekte 416×416 çözünürlükteki bir görüntüde toplamda 10.647 kutu tahmin edilir. Yani YOLOv3, YOLOv2'deki tahmin edilen kutu sayısından 10 kattan fazla kutu tahmin eder. Her ölçek için, her ızgara 3 bağlantı kullanarak 3 adet kutu tahmin edebilir. Üç tane ölçek olduğundan dolayı dokuz adet bağlantı kutusu kullanılmaktadır.

3.2.2.4. YOLOv4

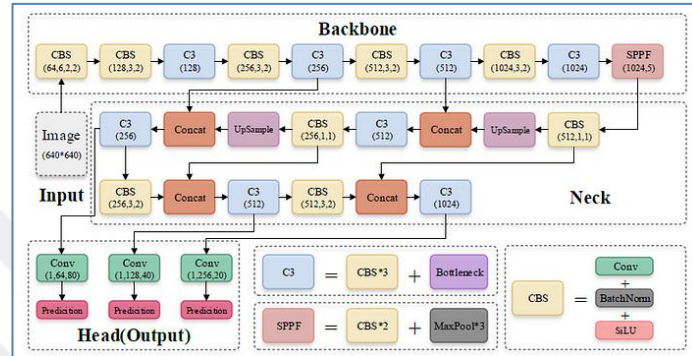
A.Bochkovskiy v.d. tarafından 2020 yılında geliştirilmiş olan YOLOv4[150], YOLOv3 modelinin gelişmiş hali olan bir nesne algılama algoritmasıdır. EfficientDet ile karşılaştırıldığında 2 kat daha hızlı performansa sahiptir. Ayrıca YOLOv3 ile karşılaştırıldığında, YOLOv4'deki ortalama kesinlik(AP) %10 ve FPS (saniyedeki kare sayısı) %12 artmıştır. YOLOv4'de girdi görüntüsünün çözünürlüğü, evrimsel katmanların sayısı, parametre sayısı ve çıktı katman sayısı arasında en uygun bir denge hedeflenmiştir. YOLOv4 mimarisinin omurga kısmında CSPDarknet53 ağı kullanılarak modelin öğrenme yeteneği iyileştirilmiş ve hesaplama maliyeti azaltılmıştır. Algoritmanın boyun kısmında PANet kullanılarak farklı katmanlardan daha iyi özellik elde edilebilmesine ve farklı boyutlu nesnelerin daha doğru tespit edilebilmesine olanak sağlamıştır. Algoritmanın kafa kısmında ise uzamsal piramit havuzlama(SPP) yöntemi kullanılmıştır.

YOLOv4 algoritmasında performansı iyileştirmek için sınıf etiketlerini yumuşatma, mosaic ve ölçeklendirme gibi çok sayıda veri artırma tekniği, farklı unuturma teknikleri kullanılmıştır.

3.2.2.5. YOLOv5

YOLOv5, orijinal YOLO algoritmasını geliştiren aynı ekip tarafından 2020 yılında açık kaynaklı bir proje olarak tanıtılmıştır. Algoritma ölçeklendirilmiş

küçük(s), orta(m), büyük(l) ve ekstra büyük(x) olmak üzere dört modele sahiptir. Küçük ölçekli modellerde nesne tespit hızı yüksek, doğruluk oranı ise düşüktür. Büyük ölçekli olanlarında ise nesne tespit hızı biraz daha düşük ancak nesne tespit doğruluğu yüksektir. YOLOv5'te model birleştirme ve test süresini artırmak için COCO veri kümesi üzerinde eğitilmiş bileşik ölçekli nesne algılama modellerini kullanarak nesneleri tespit hızına ve doğruluğu artırmaya odaklanılmıştır. Önceki sürümlere ek olarak çeşitli yeni özellikler eklenmiş ve kayda değer iyileştirmeler yapılmıştır.



Şekil 3.5 : YOLOv5 mimarisi

YOLOv5 algoritmasının mimarisi omurga, boyun ve kafa olmak üzere üç ana bölümden oluşur. Şekil 3.5' de YOLOv5 algoritmasının mimarisi görülmektedir. Girdi görüntüleri giriş katmanında işlendikten sonra bu görüntüler omurga kısmına aktarılır. Omurga kısmında özellik haritalarının elde edilmesi için CSPDarknet kullanılmıştır. Daha sonra üretilen bu özellikler algoritmanın boyun katmanına aktarılır. Boyun kısmında PANet ve FPN kullanılmıştır. Bu kısımda küçük, orta ve büyük boyutlarda nesneleri tespit etmek için özellik haritaları alt ve üst örnekleme yapılarak ölçeklendirilir. Daha sonra ölçeklendirilen bu üç özellik haritası algoritmanın kafa bölümüne aktarılarak nesneler sınıflandırılır ve görüntü üzerinde yerleştirilir. Belirli bir eşiğin altında kalan sınırlayıcı kutular elenir. Aynı bölgede örtüşen çok sayıda sınırlayıcı kutu var ise maksimum olmayan bastırma tekniği ile elenir.

3.2.2.6. YOLOv7

YOLOv7[151] gerçek zamanlı nesne tespit uygulamalarında daha hızlı ve daha doğru performans sağlamaktadır. YOLOv7 önceden eğitilmiş ImageNet omurgalarını kullanmayıp ağırlıkları Mikrosoft' un COCO veri seti ile kullanılarak eğitilmiştir. YOLOv7 algoritması da omurga, boyun ve kafa gibi üç ana kısımda

YOLOv7'nin mimari yapısında bazı köklü değişiklikler yapılmıştır. Daha iyi performans elde edebilmek ve nesne tespitini daha doğru yapabilmek için ELAN[152] tabanlı genişletilmiş ELAN gibi daha yenilikçi teknikler kullanılmıştır. ELAN tekniğinde, daha zengin ve daha anlamlı özellik haritaları elde edebilmek için ağ içerisindeki farklı katmanlardan gelen bilgiler birleştirilir. Amaç, modelin değişik ölçeklerde ve yapılarda olan nesneleri tespit edebilme yeteneğini iyileştirmek, performansı artırarak hesaplama maliyetini en aza indirmektir. Bu teknik modelin gerçek zamanlı uygulamalarda daha hızlı ve daha doğru nesne tespiti yapabilmesine olanak sağlamıştır. Şekil 3.6' da ELAN ve E-ELAN tekniği görülmektedir.



70

3.2.2.7. Diğer Nesne Algılama Dedektörleri

RetinaNet[76] 2017 yayımlanan tek aşamalı bir nesne tespit dedektörüdür. RetinaNet omurga, FPN ve iki alt ağdan oluşur. Algoritmanın omurga kısmında girdi görüntüsünden özellik haritaları üretilirken, FPN ile bu özellik haritaları farklı boyutlara ölçeklendirilir. İlk alt ağda sınıflandırma işlemi, ikinci alt ağda sınırlayıcı kutuların görüntü üzerinde yerleştirilmesi gerçekleştirir. RetinaNet eğitim sırasında sınıf dengesizliği sorununa çözüm bulmak için odak kaybı fonksiyonunu kullanarak doğruluğu önemli ölçüde artırmıştır. Odak kaybı fonksiyonu iyi sınıflandırılmış örneklerle atanan kaybın ağırlığını azaltır.

MegDet[153], büyük ölçekli veri setlerindeki nesneleri tespit etmek için tasarlanmış bir algoritmadır. Bu algoritmada hız, doğruluk ve ölçeklendirmeye odaklanılmıştır. Modelin eğitimi 256'ya kadar artırılabilen büyük mini yığınlar ile yapılır. Eğitim süresini önemli ölçüde kısaltmak için 128 GPU' ya kadar grafik kartı kullanılır. Modeli daha kısa zaman içerisinde eğitebilmek ve daha iyi doğruluk elde etmek için çapraz GPU toplu normalleştirme önerilmiştir.

EfficientDet[154] dedektörü daha yüksek doğruluk ve daha az hesaplama maliyetine odaklanan bir nesne tespit algoritmasıdır. Algoritmanın omurga kısmında önceden eğitilmiş EfficientNet[155] kullanılır. Özellik haritalarını çok yönlü ölçeklendirebilmek için iki yönlü FPN önerilmiştir. EfficientDet, yoğun nesne tespiti için odak kaybı fonksiyonunu kullanmıştır.

PAA[156], eğitim süreci içerisinde gerçek nesneleri olasılığa dayalı olarak bağlantı kutularına yerleştirilmesini iyileştiren nesne tespitinde farklı bir yaklaşım getiren gelişmiş bir modeldir. 2020 yılında yayımlanan makalede modelin öğrenme durumuna göre sınırlayıcı kutuların tahmin edilebilmesi için bağlantıları pozitif ve negatif örneklerle ayıran bağlantı ataması stratejisi önerilmiştir. Öncelikle model üzerinde koşullandırılan bağlantı skorları hesaplanır, bu skorlara uygun olasılıksal dağılım gerçekleştirilir. Daha sonra model olasılıklarına göre pozitif ve negatif şekilde ayrılmış olan bağlantılarla eğitilir. Her bir konum için çok sayıda bağlantı gerektirmediği için çok verimlidir.

3.3. Nesne Tespitinde Kullanılan Kayıp Fonksiyonları

Nesne tespit algoritmalarında nesneler hem sınıflandırılır hem de bu nesneler görüntü üzerinde yerleştirilir. Dolayısıyla bu algoritmalarda hem sınıflandırma kayıpları hem de yerleştirme kayıpları birlikte hesaplanır. Sınıflandırma kayıp

fonksiyonlarını evrişimsel sinir ağıları bölümünde geniş kapsamda ele almıştık. Bu kısımda daha çok regresyon kayıpları üzerinde duracağız.

3.3.1. Düzgün L1 Kaybı(Huber Kaybı)

Düzgün ortalama mutlak hata kaybı olarak da bilinen düzgün L1 kaybı nesne tespiti uygulamalarında sınırlayıcı kutuların yerleştirilmesinde yaygın olarak kullanılan bir kayıp fonksiyonudur[69]. Bu fonksiyon Fast R-CNN[16] modelinde tanıtılmıştır. Düzgün L1 kaybı, küçük hatalara duyarsız ile aykırı değerlere aşırı duyarlı arasında bir denge kurmayı amaçlayan ortalama mutlak hata (MAE) kaybının bir değiştirilmiş sürümüdür. Düzgün L1 kaybı denklem 3.5’ de tanımlanmıştır.

$$L(x_i, y_i) = \begin{cases} 0.5 * (x_i - y_i)^2, & \text{eğer } |x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, & \text{değilse} \end{cases} \quad (3.5)$$

Burada x_i tahmin edilen değer, y_i ise gerçek değerdir. Düzgün L1 kayıp fonksiyonu doğruluk ve verimlilikte ortalama kare hatasının üstesinden gelerek sınırlayıcı kutu koordinatlarının regresyonunu düzenlemek için iki aşamalı nesne tespit algoritmalarında yaygın olarak kullanılmıştır.

3.3.2. L2 Kaybı

Karesi alınmış ortalama hata olarak da adlandırılan L2 kaybı tahmin edilen sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki kare farkını ölçer. L2 kaybı denklem 3.6’ daki gibi tanımlanır.

$$L2 \text{ Kaybı} = \frac{1}{N} \sum_{i=1}^N (x_i - y_i)^2 \quad (3.6)$$

Burada N örnek sayısını, x_i i^{th} örnekteki gerçek sınırlayıcı kutuyu, y_i i^{th} örnekteki tahmin edilen sınırlayıcı kutuyu temsil eder. L2 kaybı modern nesne tespit algoritmalarında daha az kullanılır.

3.3.3. L1 Kaybı

Mutlak ortalama hata olarak da bilinen L1 kaybı, tahmin edilen sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki mutlak değeri ölçer. L1 kaybı denklem 3.7’ deki gibi tanımlanır.

$$L1 \text{ Kaybı} = \frac{1}{N} \sum_{i=1}^N |x_i - y_i| \quad (3.7)$$

Burada N örnek sayısını, x_i i^{th} örnekteki gerçek sınırlayıcı kutuyu, y_i i^{th} örnekteki tahmin edilen sınırlayıcı kutuyu temsil eder.

3.3.4. Birleşim Üzerinde Kesişen Kayıp(IoU)

IoU, bir görüntü üzerindeki nesnenin gerçek konumu ile tahmin edilen konum arasındaki farkı hesaplayan nesne tespit algoritmalarında yaygın olarak kullanılan bir ölçümdür. IoU ve IoU kaybı denklem 3.8' de tanımlanmıştır.

$$IoU = \frac{Kesişim\ Alan_1}{Birleşim\ Alan_1}, \quad L = 1 - IoU \quad (3.8)$$

Bu fonksiyon tahmin edilen sınırlayıcı kutunun konumu ile gerçek sınırlayıcı kutunun konumunun örtüşmesine yardımcı olur. Yüksek IoU değeri tahmin edilen kutunun konumunun gerçek konumuna yakın olduğunu ifade ederken, düşük IoU değeri ise tahmin edilen sınırlayıcı kutunun gerçek konumundan uzaklaştığını ifade etmektedir.

3.3.5. Genelleştirilmiş IoU Kaybı(GIoU)

IoU regresyon kaybı nesne algılama algoritmalarında kullanılan en popüler değerlendirme ölçütüdür. IoU' nun zayıf yönleri ele alınarak 2019 yılında H.Rezatofighi v.d. tarafından genelleştirilmiş bir sürüm GIoU [157] tanıtılmıştır. Bu sürümde hem ölçüm hem de kayıp her ikisi de ele alınmıştır. Genelleştirilmiş IoU son teknoloji nesne tespit algoritmalarına dâhil edilerek hem standart IoU hem de GIoU kullanılarak performansta tutarlı iyileşmeler olmuştur. A ve B için, nesne C'yi içeren en küçük dışbükey bulunur. Genelleştirilmiş IoU ve genelleştirilmiş IoU kayıp fonksiyonu denklem 3.9' da tanımlanmıştır.

$$C \subseteq S \in R, \quad IoU = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|}, \quad GIoU = IoU - \frac{|C \setminus (B_p \cap B_{gt})|}{|C|}, \quad L_{GIoU} = 1 - GIoU \quad (3.9)$$

Burada B_p tahmin edilen sınırlayıcı kutu, B_{gt} gerçek sınırlayıcı kutu, C ise B_p ve B_{gt} ' nin her ikisini de kapsayan en küçük çevreleyici kutudur.

3.3.6. Uzaklık-IoU Kaybı(DIoU)

Mevcut yöntemlerde l_2 -norm kaybı sınırlayıcı kutuların regresyonunda yaygın olmakla birlikte bu IoU gibi değerlendirme metriğine uyarlanmamıştır. IoU ölçümünden faydalanmak için IoU ve genelleştirilmiş IoU kayıp fonksiyonu önerilmiştir ancak doğru olmayan regresyon ve yavaş yakınsama problemleri

yaşandığından 2020 yılında Z. Zheng v.d. tarafından uzaklık-IoU(DIoU) kaybı[158] önerilmiştir. DIoU kaybı gerçek sınırlayıcı kutu ile tahmin edilen sınırlayıcı kutu arasındaki normalize edilmiş mesafeyi uyarlar. Dolayısıyla eğitim sırasında IoU ve GIoU kayıp fonksiyonlarından daha hızlı yakınsama gösterir. İki sınırlayıcı kutunun merkezi noktalar arasındaki normalize edilmiş uzaklığı en aza indirmek için normal IoU kaybına bir ceza terimi eklenir. Bu ceza terimi dekle 3.10' daki gibi tanımlanır.

$$R_{DIoU} = \frac{p^2(b_p, b_{gt})}{C^2} \quad (3.10)$$

Burada $p^2(b_p, b_{gt})$ öklit uzaklığı, C ise B_p ve B_{gt} 'nin her ikisini de kapsayan en küçük çevreleyici kutudur. Öklit uzaklığını hesaplamak için denklem 3.11' deki formül kullanılır.

$$p^2 = (x_p - x_{gt})^2 + (y_p - y_{gt})^2 \quad (3.11)$$

Burada (x_p, y_p) tahmin edilen sınırlayıcı kutunun koordinatlarını, (x_{gt}, y_{gt}) gerçek sınırlayıcı kutunun koordinatlarını, p ise öklit uzaklığını temsil etmektedir. Hesaplanan ceza terimi normal IoU kayıp fonksiyonuna eklenerek uzaklık IoU kaybı bulunur. Uzaklık IoU kaybı denklem 3.12' deki gibi hesaplanır.

$$L_{DIoU} = 1 - IoU + \frac{(x_p - x_{gt})^2 + (y_p - y_{gt})^2}{C^2} \quad (3.12)$$

Burada IoU tahmin edilen sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki kesişimi, (x_p, y_p) tahmin edilen sınırlayıcı kutunun koordinatlarını, (x_{gt}, y_{gt}) gerçek sınırlayıcı kutunun koordinatlarını, C ise B_p ve B_{gt} 'nin her ikisini de kapsayan en küçük çevreleyici kutuyu temsil etmektedir. DIoU kaybının ceza terimi doğrudan iki merkezi nokta arasındaki mesafeyi minimize ederken, GIoU kaybı ise $C - A \cup B$ alanını azaltmayı amaçlamaktadır.

3.3.7. Tam IoU Kayıp Fonksiyonu (CIoU)

Sınırlayıcı kutu regresyonunda iyi bir kayıp için üç önemli geometrik faktör dikkate alınır: örtüşme alanı, merkezi nokta mesafesi ve en-boy oranı. Koordinatları birleştirerek IoU kaybı örtüşme alanını dikkate alır ve GIoU kaybı büyük ölçüde IoU kaybına dayanır. DIoU kayıp fonksiyonu ise eş zamanlı olarak sınırlayıcı kutuların örtüşme alanını ve merkezi nokta mesafesini göz önüne almayı amaçlar. Ancak

sınırlayıcı kutular için en-boy oranlarının tutarlılığı da önemli bir geometrik faktördür. Bu nedenle DIoU kaybına dayalı olarak en boy oranının tutarlılığını uygulayarak CIoU[158] kaybı önerilmiştir. En boy oranının tutarlılığını ayarlamak için uzaklık IoU kaybına uygulanan ceza terimine αv katsayısı eklenmiştir. Yeni oluşturulan ceza terimi denklem 3.13’ deki gibi tanımlanır.

$$R_{CIoU} = \frac{p^2(b_p, b_{gt})}{c^2} + \alpha * v \quad (3.13)$$

Burada α pozitif değiş-tokuş parametresini, v en boy oranının tutarlılığını, $p^2(b_p, b_{gt})$ öklit mesafesini, C ise B_p ve B_{gt} ’ nin her ikisini de kapsayan en küçük çevreleyici kutuyu temsil etmektedir. v en-boy katsayısını hesaplamak için denklem 3.14’ deki formül kullanılır.

$$v = \frac{4}{\pi^2} (\arctan \frac{w_{gt}}{h_{gt}} - \arctan \frac{w_p}{h_p})^2 \quad (3.14)$$

Burada w_p, h_p tahmin edilen sınırlayıcı kutunun genişliğini ve yüksekliğini, w_{gt}, h_{gt} gerçek sınırlayıcı kutunun genişliğini ve yüksekliğini temsil eder. Değiş tokuş parametresi olan α katsayısı da denklem 3.15’ deki gibi tanımlanır.

$$\alpha = \frac{v}{(1-IoU)+v} \quad (3.15)$$

α ve v katsayılarının tanımlanması ile tam IoU kaybı denklem 3.16’ daki gibi tanımlanır.

$$L_{CIoU} = 1 - IoU + \frac{p^2(b_p, b_{gt})}{c^2} + \alpha * v \quad (3.16)$$

3.3.8. Ölçeğe Duyarlı IoU Kayıp Fonksiyonu (SIoU)

Ölçeğe duyarlı IoU(SIoU)[159] kaybı, 2021 yılında S. Du v.d. tarafından ortaya atılan IoU, CIoU ve Ln modelindeki kayıp fonksiyonlarından farklı yeni bir kayıp fonksiyonudur. Eğitim aşamasında bir görüntüdeki hedef alan ölçekleri çok büyük bir değişiklik gösterdiği zaman mevcut kayıp fonksiyonlar bazı özel durumlarda iki sınırlayıcı kutuyu ayırt edemeyebilir. Bu da uygun olmayan regresyon kaybı hesaplamasına ve optimizasyonun yavaşlamasına yol açar. Bu gibi durumlarda, SIoU kayıp fonksiyonu bu problemleri çözebilir. Sınırlayıcı kutuların kayıp değerlerini ayarlamak için CIoU kaybı temelinde, bir alan ölçeği düzenleme faktörü γ eklenir. Bu, teoride tüm kutuları niceliksel olarak ayırt edebilir, böylece daha hızlı

bir yakınsama hızı ve daha iyi bir optimizasyon elde edilir. Çeşitli kayıp fonksiyonları arasındaki analiz ve simülasyon karşılaştırması yoluyla SIoU kaybının üstünlüğü doğrulanır. Ölçeklendirilmiş IoU kaybı denklem 3.17’ deki gibi tanımlanır.

$$L_{SIoU} = (\gamma + 1)(1 - IoU) + \frac{p^2(b_p, b_{gt})}{c^2} + \alpha * v \quad (3.17)$$

Burada γ ölçeği düzenleme faktörünü, α pozitif değiş-tokuş parametresini, v en boy oranının tutarlılığını, $p^2(b_p, b_{gt})$ öklit mesafesini, C ise B_p ve B_{gt} ’ nin her ikisini de kapsayan en küçük çevreleyici kutuyu temsil etmektedir. Ölçeğe duyarlı IoU kayıp fonksiyonu mevcut olan CIoU kayıp fonksiyonunu üzerinde ince bir ayar yapmayı önermektedir. Bu ince ayar, regresyon kaybının hesaplanmasını daha fazla kolaylaştırır ve eğitim aşamasının başlarında yaşanan yavaş optimizasyon hızının üstesinden gelir[159]. CIoU da mevcut olan regresyon kaybı hesaplamasını etki eden örtüşme alanı, merkez nokta mesafesi ve en-boy oranı geometrik faktörlerine ek olarak alan farkı olarak adlandırılan dördüncü bir faktör tanıtlmıştır. Nesne tespitinin doğruluğu büyük ölçüde değişen ölçeklerle sahnede iyileştirilmiştir ve önerilen kayıp fonksiyonu birçok klasik modelde geniş bir uygulanabilirliğe sahiptir.

3.3.9. Odaksal Verimli IoU

Nesne tespitinde, ln-norm hem de IoU tabanlı kayıp fonksiyonları sınırlayıcı kutuların regresyonunda yetersiz olduğundan yavaş yakınsama ve hatalı regresyon sonuçlarına neden olduğu, kayıp fonksiyonlarının çoğu dengesizlik problemini görmezden gelmesinden dolayı, Y. Zhang [160] v.d. tarafından 2021 yılında yayımlanan bir makalede verimli IoU (EIoU) kaybı önerilmiştir. Öncelikle, EIoU sınırlayıcı kutuların regresyonundaki örtüşme alanı, merkez nokta ve kenar uzunluğu gibi üç geometrik faktörün tutarsızlıklarını açıkça ölçer. Daha sonra regresyon sürecinin yüksek kaliteli bağlantı kutularına odaklanmasını sağlamak için odak kaybının regresyon versiyonu Odaksal EIoU önerilmiştir. Odaksal kayıp denklem 3.18’ de, Odaksal verimli IoU kaybının hesaplanması da denklem 3.19’ da tanımlanmıştır.

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t) \quad (3.18)$$

$$\text{Odaksal Verimli IoU} = FL(p_t) * IoU \quad (3.19)$$

Burada $FL(p_t)$ odaksal kaybı temsil etmektedir.

3.3.10. YOLO Modelinde Kullanılan Kayıp Fonksiyonları

YOLO modellerinde yerleştirme, güven ve sınıflandırma kaybı olmak üzere 3 kayıp fonksiyonu kullanılır. Yerleştirme kaybı gerçek sınırlayıcı kutu ile tahmin edilen sınırlayıcı kutu arasındaki farktır. Yerleştirme kaybının hesaplanması YOLOv1 bölümünde denklem 3.1' de tanımlanmıştır. Güven kaybı bir konumda var olan bir nesnenin tahmin edilemediği durumlarda oluşan kayıptır. Nesnellik kaybı olarak da adlandırılır. Bu kayıp gerçek sınırlayıcı kutu ile tahmin edilen sınırlayıcı kutu arasında hesaplanan ikili çapraz-entropi kaybıdır. Güven kaybı denklem 3.2' de tanımlanmıştır. Sınıflandırma kaybı ise nesnenin hatalı sınıflandırılmasında oluşan kayıptır. Sınıflandırma kaybı denklem 3.3' de tanımlanmıştır.

YOLOv3 sürümüne kadar sınırlayıcı kutuların regresyonu için karesel hata kaybı toplamı fonksiyonu, sınıflandırma kayıplarını hesaplamak için ikili çapraz-entropi kaybı fonksiyonu kullanılmıştır[18, 148, 149]. YOLOv4 sürümünden sonra IoU tabanlı kayıp fonksiyonlarına odaklanılmış olup YOLOv4 [150], YOLOv5 ve YOLOv7 [151] modellerinde sınırlayıcı kutuların regresyon kayıplarında Tam IoU (CIoU) kullanılmıştır. YOLOv6 modelinde ise regresyon kayıpları için ölçeklenebilir IoU(SIoU) ve geliştirilmiş IoU(GIoU) kayıp fonksiyonları, sınıflandırma görevlerinde ise değişken odaklı kayıp fonksiyonu kullanılmıştır. YOLOv8 modelindeki regresyon problemleri için CIoU ve dağıtılmış odak kaybı kullanılmıştır.

3.4. Nesne Algılama Değerlendirme Ölçümleri

Nesne tespitinde ölçümleri hesaplamak için doğru pozitif, yanlış pozitif, doğru negatif ve yanlış negatif değerlerini tespit etmek zorundayız. Bu tanımlamalar IoU tabanlı skorlardır. IoU skoru kullanılarak ölçülen bir nesnenin gerçek konumu ile tahmin edilen konumu arasındaki eşleşmedir. IoU skoru 0 ile 1 arasında değişmektedir. IoU skorunun 1 olması nesnenin gerçek konumu ile tahmin edilen nesnenin mükemmel bir şekilde örtüşmesidir. Mükemmel bir eşleşmenin başarılması zor olduğundan bir tahminin doğru pozitif olup olmadığını saptamak için bir eşik değeri tanımlanır. Yaygın olarak kullanılan eşik değerler 0.25, 0.50 ve 0.75 dir. Bu eşik değerler sabit olmayıp uygulamanın gereksinimlerine göre ayarlanabilmektedir.

3.4.1. Birleşim Üzerinde Kesişme (IoU)

IoU, gerçek sınırlayıcı kutu ile tahmin edilen sınırlayıcı kutunun kesişim alanı, bu kutuların birleşim alanına bölünmesi ile hesaplanır. IoU hesaplanması denklem 3.20’ deki gibi tanımlanır.

$$IoU = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \quad (3.20)$$

Bu formül, esas olarak tahmin edilen sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki örtüşme derecesini ölçmektedir. Modelin tahmininin gerçek nesne konumuyla ne kadar iyi hizalandığını gösteren sayısal bir değer sağlar. Daha yüksek bir IoU puanı, tahmin edilen ve gerçek sınırlayıcı kutular arasında daha iyi bir eşleşme olduğunu gösterir. Bu da yerleştirme doğruluğunun mükemmel olduğu anlamına gelir. IoU önemli bir ölçümdür çünkü modelin görüntüler içindeki nesneleri hassas bir şekilde yerleştirme yeteneğini ölçer, nesnelerin konumlarını ne kadar iyi tanımladığını değerlendirir.

3.4.2. Ölçümde Kullanılan Parametreler

a) Nesne tespitte Doğru Pozitif (TP)

Model nesneyi doğru olarak sınıflandırıp doğru olarak konumlandırır ve tahmin edilen sınırlayıcı kutu ile doğru olan sınırlayıcı kutu arasındaki IoU skoru eğitim öncesi tanımlanan eşik değerinden büyük veya eşit ise tahmin Doğru Pozitif olur.

b) Nesne tespitte Yanlış Pozitif (FP)

Model bir görüntüde belirtilen konumda olan bir nesneyi yanlış olarak tanımlarsa veya tahmin edilen sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki IoU eşik değeri, eğitim öncesi belirlenen eşik değerden daha düşük olduğunda tahmin Yanlış Pozitif olur. Doğruluk metriğini etkiler.

c) Nesne tespitte Yanlış Negatif (FN)

Model, bir görüntüdeki belirtilen konumda var olan bir nesneyi tahmin edemediği durumlarda Yanlış Negatif oluşur. Recall metriğini etkiler.

d) Nesne tespitte Doğru Negatif (TN)

Model, bir görüntüde mevcut olmayan bir nesneyi doğru olarak tahmin ettiğinde Doğru Negatif oluşur. Nesne tespitte uygulanabilir değildir.

3.4.3. Nesne Tespitinde Yaygın Olarak Kullanılan Eşik Değerleri

0.25 eşik değeri daha yüksek recall gerektiren uygulamalarda kullanılır. Çünkü eksik yapılan tespitler yanlış teşhise yol açabilir.

0.5 eşik değeri, nesne tespitinde dengeli eşik değeri olarak yaygın kullanılan bir değerdir. Eğer, görüntüdeki tahmin edilen sınırlayıcı kutu ile doğru konumlu kutunun eşik değeri 0.5' e eşit veya 0.5' den büyük ise bu Doğru Pozitif olarak düşünülür.

0.75 eşik değeri, daha yüksek doğruluk gerektiren uygulamalarda kullanılan bir değerdir. Çünkü Yanlış Pozitif tespitler kritik sonuçlara yol açabilir.

3.4.4. Maksimum Olmayan Bastırma Tekniği(NMS)

NMS tekniği onlarca yıldır bilgisayarla görmenin önemli bir parçası olmuştur. Yaygın olarak kenar algılamada, özellik noktası algılamada ve nesne tespiti dedektörlerinde kullanılır [161]. Maksimum olmayan bastırma tekniği nesne algılama dedektörlerinde çoklu tespitleri eler ve en alakalı olan sınırlayıcı kutular seçilir. YOLO algoritmasında NMS tekniğini kullanılarak belirli bir eşik altındaki çoklu tespitler elenir, güven skoru yüksek olan sınırlayıcı kutular güven skoruna göre sıralanıp en yüksek güven skorlu olan sınırlayıcı kutu seçilir.

3.4.5. Kesinlik (Precision)

Model tarafından yapılan olumlu tahminlerin kesinliğini ölçmeye hizmet ettiğinden model değerlendirmesinde kritik bir ölçümdür. Özellikle modelin doğru nesneleri yanlış pozitiflerden ne kadar iyi ayırt ettiğini değerlendirir. Esasında kesinlik, modelin gerçekten tahminleri doğru yapıp yapmadığı hakkında bir fikir sağlar. Yüksek kesinlik puanı modelin yanlış pozitiflerden kaçınma konusunda yetenekli olduğunu ve güvenilir pozitif tahminler sağladığını gösterir. Kesinlik formülü aşağıdaki gibi hesaplanır. Kesinlik ölçümünün hesaplanması denklem 3.21' deki gibi tanımlanır.

$$Kesinlik = \frac{TP}{TP+FP} \quad (3.21)$$

3.4.6. Duyarlılık (Recall)

Doğru pozitif oranı olarak da bilinen duyarlılık, modelin performansının ölçülmesinde kullanılan bir diğer metriktir. Duyarlılık metriği, modelin görüntüdeki tüm ilgili nesneleri yakalama yeteneğini ölçer. Temelde duyarlılık metriği, modelin ilgili nesneleri tanımlamadaki bütünlüğünü değerlendirir. Yüksek duyarlılık puanı,

modelin verilerdeki ilgili nesnelerin çoğunu etkili bir şekilde tanımladığını gösterir. Duyarlılık ölçümü denklem 3.22' deki gibi hesaplanır.

$$Recall = \frac{TP}{TP+FN} \quad (3.22)$$

3.4.7. F1 Skoru

Kesinlik ve Recall ölçümlerinin harmonik ortalamasıdır. Hem yanlış pozitifleri hem de yanlış negatifleri dikkate alarak modelin performansının dengeli bir ölçümünü sağlar. Bu ölçüm özellikle veri kümesindeki pozitif ve negatif sınıflar arasında bir dengesizlik olduğunda kullanışlıdır. F1 skoru denklem 3.23' deki gibi hesaplanır.

$$F1 = \frac{2*(Kesinlik (Precision) * Duyarlılık (Recall))}{Kesinlik (Precision) + Duyarlılık (Recall)} \quad (3.23)$$

3.4.8. Ortalama Kesinlik(AP)

Nesne algılama modelleri, bir görüntüdeki birden çok nesne kategorisini tanımlar ve görüntü üzerinde konumlandırır. AP ölçümü, her kategorinin ortalama kesinliğini(AP) ayrı ayrı hesaplar ve ardından bütün kategorilere karşı bu AP' lerin ortalamasını alır. AP, farklı duyarlılık seviyelerindeki kesinlik değerleri toplanarak tahmin edilebilir. Bu nedenle mAP olarak da adlandırılır. Bu yaklaşım modelin bütün performansının her bir kategoride ayrı ayrı kapsamlı olarak değerlendirilmesini sağlar. Görüntülerdeki nesnelerin konumunu doğru bir şekilde yerleştirme yapmak ve tahmin edilen sınırlayıcı kutuların kalitesini değerlendirmek için AP IoU ölçümünü kullanır. IoU, tahmin edilen kutu ile doğru kutu arasındaki kesişim oranıdır[69]. Her bir IoU değerinde kesinlik değerlerinin ağırlıklı ortalaması denklem 3.24' deki gibi hesaplanır[162].

$$AP = \sum_{n=1}^N (R_n - R_{n-1}) P_n \quad (3.24)$$

Burada, P_n ve R_n her bir IoU değerindeki kesinlik ve duyarlılıktır.

3.4.9. mAP, mAP 50 ve mAP 50-95 Ölçümleri

Eğitilmiş bir modelin performansını değerlendirmek için kesinlik ortalamalarının ortalaması(mAP) hesaplanır. mAP, AP'lerin ortalaması alınarak hesaplanır. mAP skoru nesne tespit modellerinde yaygın olarak kullanılan bir ölçümdür. mAP denklem 3.25' deki gibi hesaplanır.

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (3.25)$$

Burada, N sınıf sayısını, AP_i ise i-th'deki sınıfın ortalama kesinliğidir. mAP@50 ölçümünde 0.5 eşik değerindeki IoU' ya sahip sınırlayıcı kutuların ortalama AP'si hesaplanırken, mAP 50-95 ölçümünde ise 0.50 den 0.95 e kadar olan değişen IoU değerlerine sahip ortalama AP'lerin ortalaması alınarak hesaplanır. mAP bütün sınıflara göre ortalama kesinliklerin ortalamasıdır.

3.5. Nesne Tanıma Çalışmalarının Amacı ve Önemi

Nesne tespiti, video ve resimlerdeki istenen nesneleri algılayıp nesneleri görüntü üzerinde göstermek ve bu nesneleri kategorilere ayırmak için geliştirilmiş olan temel bilgisayarlı görme teknikleridir. Amaç, görüntüdeki nesnelerin varlığını bulup kesin konumlarını saptayarak nesnelerin etrafını sınırlayıcı kutular çizmektir. Daha sonra tespit edilen bu nesneler önceden belirlenmiş kategorilere ayrılır.

Nesne tespiti için geliştirilen algoritmaların temel amacı, görsel verileri analiz ederek sistemin anlayabileceği verilere dönüştürmek ve dönüştürülen verileri yorumlayarak istenen kararları almasını sağlamaktır. Nesne tespiti, gerçek dünya uygulamalarında önemli bir rol oynamakta ve geliştirilen bilgisayarlı görsel sistemlerinde temel bir ihtiyaç haline gelmiştir[135]. Nesne tespitinin kullanım alanlarındaki önemini ve amacını bu bölümde açıklanmıştır.

3.5.1. Otonom Sistemler

Otonom araç, robot ve dron teknolojisi gibi alanlarda ortamdaki nesnelerin tespiti kritik rol oynamaktadır. Otonom araçlar, karmaşık ve dinamik çevre ortamındaki nesneleri etkili ve güvenli bir şekilde algılayarak karar alma işlemi yaparlar. Araçlara yerleştirilen bu otonom sistemler çevredeki yayaları, araçları, trafik işaretlerini, yol üzerindeki şeritleri veya algılanması istenen alakalı diğer tüm nesneleri saptayıp sistemin karar almasına yardımcı olurlar. Görüş alanındaki nesnelerin algılanması sayesinde bu sistemler araçların yollarda kendi kendine sürüş yapabilmesine yardımcı olmakta, önündeki araçları yeterli mesafeden takip edebilmesi için sürücüyü uyarıp sürücünün önündeki araca çarpmasını engelleyebilmekte, araçların önüne çıkan yayaları zamanında tespit edip yayalara çarpmaması için zamanında tedbir alınmasını sağlamakta, trafik işaretlerini ve şeritleri takip ederek sürücünün hata yapmasını engel olabilmekte ve belirlenen

rotalarda aracın otonom olarak sürüş yapabilmesini olanak sağlamaktadır. Sonuç olarak otonom sistemlerinde kullanılan nesne tespiti algoritmalarının çevre ortamını gerçek zamanlı olarak hızlı bir şekilde analiz edip ortamdaki nesneleri kesin ve doğru bir şekilde saptaması hayati öneme sahiptir.

3.5.2. Gözetleme ve Güvenlik

Nesne tespiti gözetleme ve güvenlik sistemlerinde potansiyel tehditlerin, davetsiz misafirlerin ve şüpheli aktivitelerin saptanmasında önemli rol oynamaktadır [135, 163]. Çok kameralı güvenlik sistemlerinde birden fazla çevre ortamında hareket eden araç, insan veya diğer başka nesnelerin insan gözüyle izlenebilmesi zorlu ve devamlılık arz eden bir gözetleme gerektirdiğinden birçok kameradan farklı ortamların güvenlik nedeniyle izlenip tehditlerin gerçek zamanlı olarak tespit edilmesi önem arz eder. Dolayısıyla gözetleme ve güvenlik sistemlerinde ortamdaki nesnelerin tespit edilmesi ve tespit edilen bu nesnelerin veri tabanına kayıt edilmesi güvenlik sistemlerine katkıda bulunmaktadır.

Nesne algılama, insan pozlarını izlemeye kullanılarak bilgisayarlar ve cihazlarla daha sezgisel ve doğal etkileşimler sağlar.

3.5.3. Tıbbi Görüntü İşleme

Tıp alanında nesne tespiti, tıbbi görüntüler üzerinden hastalıkların teşhis edilmesi, sınıflandırılması, görüntüler üzerinde hastalık olup olmadığının belirlenmesi, görüntülerdeki hastalığın konumunun saptanması gibi konularda hayati önem arz eder [164]. Örneğin beyin MR görüntülerinde tümörün yerinin sistem tarafından saptanması tıp alanında yaşanan gelişmelerdir.

Normal ve hastalıklı hücrelerin geleneksel olarak insan gözü ile manuel olarak tespit edilip sınıflandırılması istenen doğrulukları elde etmede tatmin edici bir uygulama olmayıp günümüz teknolojisinde hücre ve patolojik hücrelerin sayımı ve sınıflandırılması nesne tespit algoritmaları ile sağlanarak sağlık alanındaki gelişmelere katkı sağlamaya devam etmektedir [165].

3.5.4. Nesne Tespitinin Endüstri Alanındaki Katkısı

Nesne tespiti algoritmalarının gelişimi ile endüstri alanında ürünlerin kalite ve durum kontrolü gibi birçok durum nesne tespiti sistemleri ile izlenebilmekte ve ayıplı ürünlerin ayıklanabilmesine yardımcı olmaktadır.

3.5.5. Nesne Tespitinin Çevresel İzlemeye Katkısı

Nesne tespiti algoritmaları sayesinde, vahşi yaşamın izelenerek hayvanların tespiti ve sınıflandırılması, çiçek türlerinin kimliklendirilmesi, doğa ortamındaki değişikliklerin takip edilmesi gibi alanlarda uygulamaların geliştirilmesine yardımcı olmaktadır.

3.6. Nesne Algılama Algoritmalarında Karşılaşılan Zorluklar ve Çözümler

Gerçek zamanlı nesne algılama algoritmalarında görsel verilerin hızlı ve hassas işlenmesine bağlı olarak birçok zorluğu beraberinde getirmektedir. Bu zorluklar, gerçek dünyadaki seneryoların karmaşıklığından ve gerçek zamanlı performansın gerekliliklerinden kaynaklanmaktadır. Algoritma içerisinde görüntüdeki nesnelerin tespit hızı ile tespit edilen nesnelerin doğruluk oranında hassas bir denge vardır [135].

3.6.1. Hesaplama Karmaşıklığı

YOLO ve R-CNN gibi derin öğrenme tabanlı nesne algılama modellerinde sahip olduğu karmaşık mimarileri ve çok sayıda parametre içermesi nedeniyle önemli oranda karmaşık hesaplamalara ihtiyaç duyulmakta ve bu da hesaplama maliyetini artırmaktadır. Kısıtlı kaynaklara sahip platformlarda gerçek zamanlı performansa ulaşmak için modelin çok iyi bir şekilde optimize edilmesi, bu platformlarda hızlandırma teknikleri ile görüntüdeki nesnelerin hızlı bir şekilde algılanması hala karşılaşılan zorluklardan biridir. Ayrıca, algoritma hem karmaşık mimariye sahip olması hem de algoritma içerisinde modelin aşırı öğrenmesinin önüne geçmek için veri artırma tekniklerinin kullanılması modelin eğitiminin süresini önemli oranda artırmaktadır.

3.6.2. Hız ve Doğruluk Arasındaki Hassas Denge

Gerçek zamanlı nesne algılama algoritmalarında hız ve doğruluk arasında bir denge sorunu vardır. Kısıtlı kaynaklar için geliştirilen küçük ölçekli algoritmalarda, görüntünün hızlı bir şekilde işlenebilmesini sağlamak için model daha basit hale getirilip bazı parametrelerin çıkarımı yapılması sağlanır ancak bu da modelin doğruluğunu kayda değer oranda azaltıp modelin performansını önemli oranda olumsuz etkileyebilir. Gerçek zamanlı uygulamaların gereksinimlerini karşılamak ve kabul edilebilir bir performansa ulaşmak için nesne algılama algoritmalarında

doğruluk ve hız arasında optimum sonuca ulaşmak arzu edilen bir sonuç olup hala optimum sonuca ulaşmak için bu alandaki çalışmalar devam etmektedir.

3.6.3. Farklı Ölçekli Nesneleri Algılama

Görüntüde yer alan birçok nesne sabit boyutta olmayıp nesneler değişken ölçeklerde olabilmektedir. Örneğin kamera ile yollardaki araçları gözlemlediğimizde, araçlar uzaktan yakına doğru gelmekte olup araçların boyutları değişmektedir. Dolayısıyla, her bir kategorideki nesnelerin değişken ölçekteki örnekleri veri setinde bulunması ve bu örneklerin modelin eğitiminde mevcut olması gerekir. Ancak, son yıllarda geliştirilen algoritmalarda önemli oranda iyileştirmeler yapılmış, özellikle YOLO algoritması içerisinde yer alan ölçeklendirme tekniği ile veri artırımı yapılarak nesnelerin değişken boyuttaki örnekleri modelin eğitiminde kullanılmaya başlanmış olup modelin performansını olumlu bir şekilde etkilemeye başlamıştır. Ayrıca özellikle YOLOv7 sürümünden sonra algoritma içerisinde evrimsel hesaplamalar ELAN gibi yeni teknikler ile hesaplanarak çok sayıda ölçeklendirme işlemi yapılarak küçük nesnelerin daha doğru tespit edilmesi hedeflenmiştir.

3.6.4. Farklı Açı ve Görünüşteki Nesnelerin Algılanmasındaki Zorluklar

Görüntülerdeki nesnelerin açıları değiştiğinde nesnelerin görünüşleri de değişmektedir [166]. Özellikle görüntüdeki araçları tespit ederken araçların farklı açılardaki görünüşlerinde farklılıklar olacaktır. Video görüntülerinin her bir görüntüsü sistem tarafından işlenirken araçların görünüşleri değişmesi sebebi ile araçların şekilleri, boyutları ve renkleri değişmektedir. Gerçek zamanlı olarak araçlar tespit edilirken sistemin nesnelerin bu poz değişikliğinden etkilenmeden nesneleri doğru bir şekilde tespit etmesi gerekir. Dolayısıyla veri setinin nesnelerin bu görünüşleri zenginleştirilmesine ihtiyaç vardır. Ancak, nesnelerin her bir açıdan görünüşü özellikle trafikteki araçları sınıflandırırken her bir sınıftaki araçların farklı görünüşlerini elde etmek ve bu verileri etiketlemek gerçekten çok zorlu bir süreçtir. Kısıtlı veri setini veri artırma teknikleri ile zenginleştirmek farklı görünüşteki araçları tespit etmesini kolaylaştıracak bir çözümdür.

3.6.5. Birbirlerine Karışmış Olan Nesnelerin Algılanması

Görüntüdeki nesneleri tespit etmedeki bir diğer zorluk nesnelerin birbirine yakın veya birçok nesnenin karışık halde olma(occlusion) durumudur. Birbirine karışmış olan bu nesneler hem aynı kategoride hem de farklı kategoride

olabilmektedir. Birbirlerine karışmış vaziyette olan nesneler görüntüde gözle kısmi olarak görülür [167]. Birbirlerine karışmış vaziyette ve her biri kısmi olarak görünen bu nesneleri algoritma tek bir nesne olarak algılayabilir veya bu nesneleri farklı bir sınıf olarak etiketleyebilir. Dolayısıyla bu nesnelerin birbirinden ayrıştırılarak algılanması, aynı veya farklı kategorilerde olabilen bu birleşik nesneleri tespit edip etiketlemek nesne algılama algoritmalarında zorluklar arasında yer almaktadır.

Özellikle yoğun araç trafiğindeki araçlar veya kalabalık insan topluluğu bu probleme örnek verilebilir. Yoğun trafikteki araçların tespit edilip sınıflandırılması zorlu bir problemdir.

Birbirlerine karışmış şekilde olan nesneleri birbirlerinden ayrıştırarak doğru bir şekilde sınıflandırmak için farklı çözümler geliştirilmiştir.

- a) Veri setinin boyutunu artırmak ve veri setini çeşitlendirmek karışık vaziyette olan nesnelerin tespit edilip ayrıştırılmasında modelin performansını artırabilmektedir.
- b) Birbirine yakın olan nesneleri doğru bir şekilde etiketlenmesi, nesneler etiketlenirken boşluk bırakılmaması
- c) Çok fazla parametre içeren YOLO' nun daha büyük modellerinin tercih edilip modelin karmaşıklığını artırmak, görüntü üzerindeki ince ayrıntıları tespit etmede faydalı olabilir.
- d) Çalışmanın türüne bağlı olarak optimum sonuca ulaşmak için YOLO algoritması içerisinde yer alan öğrenme oranı ve veri artırma tekniklerinde kullanılan parametrelerin çok iyi ayarlanması

Birbirlerine karışmış şekilde olan nesneleri tespit etmek hala günümüzde araştırma konusu olup bu alandaki çalışmalar devam etmektedir.

3.6.6. Ortamlara Uyum Sağlamadaki Zorluklar

Gerçek zamanlı nesne algılama sistemlerinde ortamlardaki değişimlere, aydınlatma şartlarına uyum sağlama bir diğer zorluklar arasında yer alır. Örneğin, ortamın karanlık veya aydınlık olması, hava şartlarından dolayı havanın yağmurlu, güneşli, bulutlu, karlı olması görüntülerdeki nesneleri tespit etmede modelin performansını olumsuz yönde etkileyebilmektedir. Dolayısıyla, geliştirilecek olan çalışmada sistemin bu gibi ortamlara uyum sağlaması göz önünde bulundurularak farklı ortamlardan elde edilen örneklerin veri setine eklenerek veri setinin zenginleştirilmesi gerekmektedir. Ancak, her bir ortamda büyük çaplı veri setini elde

etmek zorlu bir süreç ve devasa veri setini etiketlemek zaman açısından da maliyetli bir iştir. Sistemin farklı ortamlara uyum sağlaması için farklı yaklaşımlar önerilmiş, geliştirilmiş ve iyileştirilmiş olan nesne algılama algoritmalarında yer alan veri artırma teknikleri ile mevcut olan örnek görüntülerin parlaklığı ve doygunluğu rastgele değiştirilip yeni örnekler elde edilerek bu örneklerin modelin eğitiminde kullanılmış ve farklı ortamlarda da modelin performansında önemli iyileştirmeler yapılmıştır.

3.6.7. Küçük Nesnelerin Tespitindeki Zorluklar

Küçük nesneleri özellikle belirli bir mesafede bulunan veya düşük çözünürlüğe sahip nesneleri tespit etmek bir hayli zorluk teşkil etmektedir. Nesne algılama modellerinde küçük nesnelerin gerçek zamanlı olarak genel performanstan ödün vermeden tespit edilmesinde zorluklarla mücadele edilmektedir [168].

Özellikle YOLO gibi modern nesne algılama algoritmalarında nesnelerin özellik haritalarını çıkartmak için ağırlıklı olarak evrimsel katmanlar kullanılmaktadır. Evrimsel katmanların çok sayıda olmasına bağlı olarak katmanlar arasında evrimleştirilerek geçirilen nesnelerin boyutları azalır. Dolayısıyla, küçük nesneler bir kaç katmandan sonra kaybolur ve tespit edilemez hale gelir.

Görüntüdeki küçük nesneler düşük çözünürlüğe sahip olduğundan nesnelere ait kenar bilgisi gibi detaylar tam belirgin olamaz[169]. Dolayısıyla düşük çözünürlüğe sahip bir görüntüde nesnelere ait küçük detaylar atlanabileceğinden küçük nesneleri tespit etmek zorlaşacaktır. Küçük nesneler genellikle düşük doygunluğa sahip olması, nesne ile arka plan arkasındaki parlaklığın çok az olmasına bağlı olarak küçük nesnelerin tespit edilmesi zor olacaktır[169]. Küçük nesnelerin, boyutları nedeni ile kenar, yapısı vb. çok az görsel bilgiye sahip olması sebebi ile tespit edilmesi ve doğruluk oranının yüksek olmasını engel olabilmektedir. Algoritma küçük nesneleri tespit ederken kolaylıkla hatalı sınıflandırmalar yapabilir, görüntü üzerinde doğru şekilde saptanamayabilir. Küçük nesneleri görüntü üzerinde tespit etmek için çok ölçekli aramalara ihtiyaç duyulur. Bu da hesaplama maliyetini artırır.

Derin öğrenme teknikleri ile küçük nesnelerin tespiti yapılabilmesi için görüntünün yüksek çözünürlüğe sahip olması çok önem arz etmektedir. Geliştirilen YOLOv7 algoritması genişletilmiş ELAN gibi yenilikçi teknikler kullanarak önceki sürümlerinden farklı olarak yüksek çözünürlükteki görüntüleri de işleyebilmesi daha

küçük nesnelerin tespit edilebilmesinde modelin doğruluk performansını önemli oranda iyileştirmiştir. Ancak, yüksek çözünürlüklü görüntüler üzerinden nesne tespiti yapılabilmesi için hesaplama maliyetini de olumsuz yönde etkilemektedir.

Derin öğrenme tabanlı nesne algılama algoritmalarında küçük nesneleri tespit etmek hala zorluklar içermekte ve bu alandaki çalışmalar devam etmektedir.

4. NESNE TESPİTİ İLE İLGİLİ YAPILAN ÇALIŞMALAR

Nesne tespiti, resimlerdeki veya kameralardan elde edilen görüntülerdeki nesneleri tanımlamak, bu nesneleri görüntü üzerinde konumlandırmak ve bu nesneleri sınıflara ayırmak için derin öğrenme teknikleriyle geliştirilen bir bilgisayarlı görme tekniklerindendir [170]. Görüntülerdeki nesneleri tespit etmek, nesneleri görüntü üzerinde yerelleştirmek ve onları sınıflara ayırmak için birçok çalışmalar yapılmış ve bu konuda birçok algoritma geliştirilerek makaleler yayımlanmıştır. Geliştirilen bu algoritmalar sayesinde birçok alanda görüntülerdeki birçok sayıda nesnelerin kategorilere ayırarak tespit edilmesi sağlanmıştır. Geliştirilen bu algoritmalar arasında SSD ve YOLO en popüler olanıdır. Çünkü bu algoritmalar sayesinde gerçek zamanlı uygulamalarda nesne tespitinin yapılabilmesi sağlanabilmiştir.

Nesne tespiti algortimalarının son yıllarda popüler olması ile hemen hemen tüm alanlarda büyük çalışmalar yapılmıştır. Araçların otonom hale gelmesi ile sürücülere büyük kolaylıklar sağlayarak sürücülerin hata yapmasına veya otomatik sürüş gibi yardımcı hizmetler ile araçların otonom şekilde yolda gidebilmesine, sürücülerin uykulu halini tespit edip zamanında gerekli olan tedbiri almasına yardımcı olmaya başlamıştır [171]. Güvenlik alanında birçok kameradan tespit edilen görüntüler eş zamanlı olarak işlenerek oluşabilecek tehditlerin algılanması ile güvenlik görevlilerine yardımcı olmaktadır [163]. Tıp alanında ise medikal görüntülerde hastalık olup olmadığının hızlı bir şekilde teşhis edilmesi, görüntülerdeki hastalığın nesne tespiti algoritmaları ile yerinin tam olarak saptanması, patolojik ve kan hücrelerinin sınıflandırılması ile bu alanda harcanan zamanın kısaltılmasına katkıda bulunmaktadır [164, 165]. Endüstri alanında ise, örnek olarak ayıplı ürünlerin insan gözü ile yapılırken bu sistemlerin gelişmesi ile

ürünlerin otomatik olarak izlenip ayıplı ürünlerin ayrılmasına yardımcı olarak işlemlerin daha hızlı ve daha doğru bir şekilde yapılması sağlanmaktadır.

K. Vaishnavi v.d.[170], SSD algoritmasını kullanarak SSD methodunun performansını artırmaya çalışmışlardır.

2022 yılında Abubakar M. Ashir [172] ise, yoğun trafik akışında farklı acil durum araçlarının gerçek zamanlı olarak tespitini sağlayan bir uygulama geliştirmiştir. Bu çalışmanın acil durum araçlarının gideceği güzergâhı açacağını, yolun boş olması hasta taşıyan ambulanslar için zamanı kısaltacağı için en kısa sürede hedefine ulaşacağını, ayrıca suçların da azalacağını dile getirmiştir. YOLOv5 algoritmasını kullanarak acil durum araçlarını itfaiye, ambulans, polis aracı ve normal araç olmak üzere 4 kategoriye ayırmıştır. YOLOv5 ile eğitilen modelde polis araçları için %98, itfaiye araçları için %96, ambulans araçları için %89 ve normal araçlar için %97' lik bir mAP değeri elde edilmiştir. Modelin eğitimini öğrenme transferi yaparak gerçekleştirmiştir.

2021 yılında S.KIRCA [173] tarafından derin öğrenme yöntemi kullanılarak araçları ve araçların plakasını tespit eden bir çalışma yapılmıştır. Yayımlanan makalede araçların yeni modelleri çıktığında eğitilen modelin eski araçları tanıdığından yeni araçları tespit edemeyeceği dile getirilmiş, çalışmada araçların amblemleri üzerinden araç tanıma işlemi yaklaşımı kullanılarak araç tanıma işlemi geliştirilmiştir.

2022 yılında Z.Yu[174] tarafından, araç hedeflerini yanlış tespit etme oranını azaltmak için farklı trafik seneryolarında araçları tespit etmek için geliştirilmiş bir YOLOv5 algoritması önerilmiştir. Önerilen bu metotda, flip-mosaic algoritması kullanılarak modelin küçük hedefleri algılamasını artıracak şekilde dile getirmiştir. Flip – mosaic algoritması aynı sınıfta benzer olan bir kaç görüntüyü birleştirerek tek bir görüntü hale getirmektedir. Çalışmada modelin flip – mosaic metodu ile eğitildiğinde araçların tanıma doğruluğunu artırabileceği, yanlış tespit oranını azaltabileceği gösterilmiştir.

2021 yılında M.Priymak[175], karayollarındaki trafik akışının izlenebilmesine ve yönetilebilmesine yardımcı olmak için Melbourne yolundaki araçlardan elde edilen temsili veri kümesi ile son teknoloji derin öğrenme tekniği ile geliştirilmiş olan YOLOv5 algoritmasını araç tanıma uygulaması geliştirerek değerlendirmiştir. Makalede, kısıtlı kaynaklara sahip olan trafik izleme cihazlarında çalışması için YOLOv5' in nesneyi tespit doğruluğu, model büyüklüğü ve gerçek

zamanlı nesne tespiti yapabilme kapasitesi arasında mükemmel bir denge sağladığını dile getirilmiştir.

2021 yılında K. Margrit [176], YOLOv5 algoritmasını kullanarak kışın park edilmiş ağır yük taşıyan tırları tespit eden bir çalışma yapmıştır. Kışın havanın karlı olması ve geceleyin görüntülerin tam net olmaması nedeniyle termal kamera kullanılmıştır. YOLOv5 modeli ile çalışma eğitilirken hızlı ve daha iyi bir doğruluk elde etmek için öğrenme transferi yapılmıştır.

2022 yılında bir benzer çalışma Z.Kaijie v.d. [177] tarafından YOLOv5 algoritmasından faydalanılarak trafikteki araçları tespit eden, araçları tanıyan ve sayan bir çalışma yapılmıştır.

F.S.Mim[178] v.d., 2022 yılında YOLOv4 algoritması ile trafikteki bileşenleri çok hızlı ve doğru bir şekilde tespit edip sınıflandıran bir çalışma yapmışlardır. Bu bileşenler, araba, otobüs, kamyon, yaya, trafik ışığı, trafik işareti, araç plakası, motosiklet, ambulans ve bisikletten oluşmaktadır. Model gün ışığı, düşük ışık, yoğun trafik, sisli ve yağmurlu bir ortamdaki veri setleri ile eğitilmiş ve %65.95'lik bir mAP doğruluk oranına ulaşılmıştır.

K. Andhy Panca Saputraa v.d.[179] çöpleri cam, metal, kağıt ve plastik olarak ayrıştırmayı önermiş ve modeli geliştirmek için YOLOv4 ve YOLOv4-tiny derin öğrenme algoritmasını kullanarak bu çöpleri ayrıştırmayı amaçlamışlardır. Veri seti 3870 çöp görüntüsünden oluşup 4 kategoriye ayrılmıştır. YOLOv4 algoritmasının küçük ölçekli modelden nesne tespitinde daha iyi bir performans sergilemiştir. Geliştirilen model %89,59 mAP, %76 kesinlik ve %90 duyarlılık değerine ulaşmıştır. Modelde ayrıca mosaic veri artırım metodu kullanılmıştır.

A.Y.Abdulkadhim[180] tarafından yayımlanan bir başka makalede YOLOv5 algoritması kullanılarak trafikteki araçlar tespit edilmiştir. Araçlar araba, taksi, minibüs, kamyon ve tuk-tuk motor taksi olmak üzere 5 kategoriye ayrılarak sınıflandırılmıştır. 100 döngüde mAP değeri 0.924'lere kadar ulaşmıştır.

5. GELİŞTİRİLEN MODEL

Geliştirecek olduğumuz modelimizin amacı derin mimari teknikleri kullanarak dijital kameralardan elde edilen görüntüler üzerinde seyir halinde olan

araçları tespit edip bu araçları çok sayıda kategoriye ayırmaktır. Çalışmamıza geçmeden önce özellikle evrişim sinir ağları hakkında çok detaylı literatür taraması yaparak sinir ağının mimari yapısını, modelin nasıl eğitildiği, sinir ağı modelinin nasıl geliştirildiği, modelin performansını artırmak için ne gibi teknikler kullanıldığı, modelin performansının artmasını etkileyen problemler ve bu problemlere ne gibi çözümler getirildiği hakkında gerekli olan bilgileri edinmiş olduk. Sonrasında evişimsel sinir ağları kullanılarak görüntü üzerindeki nesnelerin nasıl tespit edildiği, nesnelerin görüntü üzerinde yerleştirilmesinde yaşanan sorunlar, geleneksel ve modern nesne tespit algoritmalarının çalışma prensipleri hakkında geniş ve kapsamlı araştırmalar yapmak zorunda kaldık.

5.1. Eğitim Öncesi Süreç

Çalışmamızın amacı seyir halinde olan araçları görüntü üzerinde tespit edip bu araçları çok sayıda kategoriye ayırmaktır. Çalışmaya geçilmeden önce modeli eğitebilmek için sağlam ve büyük bir veri setine ihtiyaç duyulmuştur. Bu veri setinin toplanmasından sonraki süreç ise bu veri setinin etiketlenmesidir. Daha sonra modeli eğitebilmek için uygun bir algoritmanın seçilmesi işlemidir. Dijital kameralardan elde edilen görüntülerin işlenerek görüntüler üzerindeki nesnelerin gerçek zamanlı olarak tespit edilebilmesi, bu nesnelerin doğru bir şekilde sınıflandırılması ve bu tespit edilen nesnelerin doğruluk oranlarının yüksek olması gerekir. Dolayısıyla nesne tespit algoritmalarındaki süreci yakından takip ederek hangi algoritmalar bizim çalışmamıza daha uygun olduğunu belirledik.

5.1.1. Araçların Görüntü Üzerinde Tespit Edilmesi ve Sınıflandırılması

Geliştirmiş olduğumuz modelde trafikte seyretmekte olan araçlar(elektrikli ya da motorlu) çok sayıda kategoriye ayrılmıştır. Hedefimiz trafikteki araçları görüntü üzerinde tespit edip bu araçları sınıflara ayırmaktır. Buna ek olarak araçların plakalarını, trafikteki yayaları ve yollarda meydana gelen trafik kazalarını da tespit etmek istedik. Özellikle kara araçları olmak üzere havada uçak ve helikopter, denizde gemi, tekne ve jetski araçlarını da sınıflandırdık. Veri setimiz toplam 148.567 görüntüden oluşmaktadır. Her bir görüntü birden fazla nesne içerebilmekte olup görüntülerde toplam 459.180 nesne içermektedir. Görüntülerin çözünürlükleri ve derinlikleri farklılıklar göstermektedir. Her bir görüntü içinde birden fazla araç sınıfı

bulunabilmektedir. Seçmiş olduğumuz YOLO algoritması aynı görüntü içinde birden fazla sınıfa ait etiketlemeye olanak sağlamaktadır.

5.1.2. Veri Toplama

Karayollarındaki araçları tespit etmede bazı problemler oluşmaktadır. Birincisi, araçlar kameralardan izlenirken, araçlar farklı açılarda olması araçların görünüşlerini de değiştirmektedir. Araçların ön, arka, yan ve üst açılarda farklı perspektiflerle izlendiğinde araçları tespit etmek gerçekten zor bir iştir. Araçları farklı perspektifle tespit etmek için çok büyük bir veri setine ihtiyaç duyulmaktadır. Büyük veri setine ulaşmak oldukça zor olduğundan ve bu veri setini etiketlemek gerçekten çok maliyetli bir zaman gerektirdiğinden algoritma içerisinde ölçeklendirme, yansıtma, dönüştürme gibi çeşitli veri artırma yöntemleri kullanılmaktadır. İkincisi, hava şartlarıdır. Havanın aşırı parlak, yağmurlu, karlı ve bulutlu olması, kameradan araçların tespit edilmesi zorlaşır. Zikredilen hava şartlarındaki gibi etiketlenmiş veri setinin de olması araçların tespitini kolaylaştıracaktır. Ancak böyle büyük bir veri setine ulaşmak gerçekten çok zor olduğundan bu gibi konular düşünülerek algoritma içerisinde yine görüntünün parlaklık ve doygunluk oranları üzerinde ince ayar yapılması ile veri artırma işlemi yapılmıştır. Üçüncü olarak, araçlar kameralardan izlenirken araçlar farklı ölçeklerde olabilmektedir. Farklı ölçeklerde olan her bir aracın tespit edilmesi işi de zor bir problemidir. Araçlar küçüldükçe veya büyükçe araçların tespiti zorlaşacaktır. Her bir sınıftaki aracın farklı ölçeklerde görüntü üzerinde doğru bir şekilde tespit edilebilmesi için bu araçların farklı ölçeklerinin veri setinde olması ve eğitimden geçmesi gerekir. Ancak bu kadar büyük bir veriyi toplamak zor olduğundan algoritma içerisinde ölçeklendirme gibi veri artırma yöntemleri kullanılmıştır. Ayrıca YOLOv7 algoritmasında ELAN gibi yenilikçi tekniklerin kullanılması sayesinde evrişim sırasında çok sayıda özellik haritaları ölçeklendirilerek küçük nesnelerin tespit edilebilmesini olanak sağlamıştır. Dördüncü olarak yoğun trafikte araçlar birleşik olarak görüntülenir. Bu yoğun trafikte araçları birbirinden ayırt ederek tespit etmek gerçekten zor bir problemidir. Birden fazla araç tek bir araç gibi tespit edilebilir. Dolayısıyla, yoğun trafikte olan araçların etiketlenerek veri setine eklenmesi, araçların yoğun trafikte tespit edilmesini kolaylaştıracaktır. Ayrıca YOLOv7 algoritması içerisinde bulunan mosaic hiper parametresi, bu gibi durumlara

yardımcı olabilmek için veri artırma işlemi yaparak modelin doğruluk oranının iyileştirilmesine yardımcı olabilmektedir.

Veri setini oluştururken birçok açık kaynaktan faydalanılmıştır. Araç sınıfımız 35 kategoriden oluştuğu için birçok kaynaktan yararlanılmaya çalışılmıştır. Öncelikle, <https://www.kaggle.com/> sitesinden bir dataset araştırması yapılmış olup araç tanıma modeline benzer geliştirilen uygulamalar incelenmiş, uygun görülen etiketli ya da etiketsiz veriler elde edilmiştir. Etiketli veriler kısa bir algoritma yazılarak bizim istediğimiz formattaki araç sınıflarına dönüştürülmüştür. İkinci olarak, Türkiye’deki yolları gösteren birçok video elde edilip program yardımıyla framelelere ayrılarak araç içeren görüntüler manüel olarak program yardımıyla YOLO formatında etiketlenmiş ve veri setine eklenmiştir. Ayrıca, Google ve Yandex gibi arama motorlarında eksik olan araç sınıfları görsel olarak aratılarak birçok veri elde edilmiştir. Bu veriler yine YOLO formatında manüel bir şekilde etiketlenmiştir.

5.1.3. Model İçin Görüntü Etiketleme Metodu

YOLO gibi denetimli nesne tanıma algoritmalarını eğitebilmek için, görüntü üzerindeki nesnenin sınıfını, koordinatlarını, genişliğini ve yüksekliğini belirten dosyalara ihtiyaç duyulmaktadır. Her bir görüntü dosyasındaki nesnelerin sınıfı, koordinatları, genişliği ve yüksekliği aynı isimde olan txt uzantılı text dosyalarında tutulmaktadır. Şekil 5.1. araçların açık kaynak makesense.ai program yardımıyla etiketlenmesini göstermektedir.



Şekil 5.1 : Araçların Etiketlenmesi

Verileri etiketlemek için nesne etiketleme aracı olan “<https://www.makesense.ai/>” açık kaynaklı web sitesinden faydalanılmıştır.

Etiketleme şekil 5.1’ de görüldüğü gibi nesne dikdörtgen bir kutu içerisine alınarak yapılır. Siteye etiketlenecek olan görüntüler ve araç sınıfları yüklenir. Aynı görüntü üzerinde birden fazla farklı sınıfta araç etiketlenebilmektedir. Araçlar etiketlendikten sonra YOLO formatlı dosyalar aynı görüntü ismindeki dosyalar txt uzantılı olarak export edilir. Görüntü üzerinde etiketlenen bütün nesneler tek bir text dosyasına kaydedilir. Tablo 5.1’ de tek bir görüntü üzerinde etiketlenen nesnelerin koordinat ve sınıf bilgileri gösterilmektedir. İlk sütun araç sınıflarını belirtirken, x ve y sütunu nesnelerin koordinatlarını, w sütunu nesnelerin genişliğini, h sütunu ise nesnelerin yüksekliğini belirtmektedir. Her bir text dosyası sadece bir görüntüye hitap etmektedir. Görüntü üzerinde birden fazla nesne var ise YOLO algoritması bunu desteklemekte olup her bir nesnenin bilgileri text dosyasında satır satır belirtilir.

Tablo 5.1 : YOLO algoritmasındaki nesnelerin koordinatları ve sınıfları

Araç Sınıfı	X	Y	W	h
3	0.474426	0.488281	0.025023	0.078125
1	0.400879	0.515625	0.036133	0.126953
8	0.688477	0.488281	0.094727	0.400391
11	0.106201	0.703613	0.116699	0.409180
0	0.301514	0.602539	0.073731	0.275391

Yani birden fazla kategoriye ait nesnelerin sınıf, koordinat, genişlik ve yükseklik bilgileri tek bir dosyada mevcut olabilir. Görüntü üzerindeki nesnelere ait x, y, w, h bilgileri orijinal görüntünün genişliği ve yüksekliği hesap edilerek normalize edilmiş değerlerdir. Dolayısıyla, orijinal görüntünün çözünürlüğü değişse bile bilgiler normalize edilmiş bilgiler olduğu için orijinal görüntünün küçülmesi veya büyümesi nesnelerin koordinat, genişlik ve yükseklik bilgilerini etkilemeyecektir. Bu text dosyalarındaki nesne bilgileri modeli eğitmek için kullanılan bilgilerdir.

Ayrıca, mevcut veri setini manüel olarak etiketlendiği gibi, önceden eğitilmiş olan modeller kullanılarak etiketlenmek istenen görüntüler bu modelle ilgili nesneler tespit edilmiştir. Eksik olan nesneler manüel etiketlenmiş, tespit edilen nesneler de modele uygun şekilde küçük bir algoritma yazılarak uyarlanmıştır.

Görüntü etiketlemede bir başka konu ise, görüntülerin hatasız, doğru bir şekilde etiketlenmesi durumudur. Görüntülerin doğru değil de yanlış sınıflara etiket ataması yapılması, modelin eğitiminde iyi bir doğruluk oranının elde edilmesinde problemler yaşanmasına neden olacaktır. Dolayısıyla, görüntüler etiketlenirken görüntünün doğru bir sınıfa hitap etmesi gerekir. Aksi takdirde, modelin doğruluk oranı istenilen seviyeye ulaşamamasına ve doğrulama setinde yanlış sınıflanmasına neden olacaktır.

5.1.4. Veri Setini Eğitim, Doğrulama ve Test Seti Olarak Bölüm İşlemi

Veri setimiz toplam 148.567 görüntüden oluşmaktadır. Görüntüler birden fazla nesne içerebilmekte olup görüntülerde toplam 459.180 adet nesne bulunmaktadır. Mevcut veri setinin 1/5'i doğrulama seti olarak geri kalan verinin 1/10' u ise test seti olarak ayrılmıştır. Yani, toplam veri setinin %70' i eğitim seti, %20' si doğrulama seti, %10' u da test seti olarak ayrılmıştır. Eğitim setinde toplam 331.354 nesne, doğrulama setinde toplam 90.501 nesne, test setinde ise toplam 37.325 nesne bulunmaktadır. Tablo 5.2' de veri setinin eğitim, doğrulama ve test olarak bölünmesi gösterilmiştir.

Tablo 5.2 : Veri setinin eğitim, doğrulama ve test seti olarak bölümü

Veri Seti	Oran	Görüntü Sayısı	Etiket Sayısı
Eğitim	%70	106.970	331.354
Doğrulama	%20	29.712	90.501
Test	%10	11.885	37.325
Toplam	%100	148.567	459.180

Eğitim süreci boyunca modelin performansının görülmemiş veriler üzerinde değerlendirilmesi modelin sağlamlığı açısından çok önem arz etmektedir. Her bir eğitim döngüsünde modelin performansı doğrulama seti üzerinde ölçülerek kayıpların durumuna göre modelin parametreleri yeniden ince ayardan geçilir. Doğrulama verisi ile elde edilen değerlendirme ölçümleri sayesinde eğitimin durumu hakkında detaylı şekilde inceleme olanağı elde edilir. Modelin genelleme yapabilme yeteneği hakkında bilgi sahibi olunur. Doğrulama kayıpları detaylı bir şekilde yakından takip edilerek modelin aşırı uyum gösterip göstermediği hakkında bilgi edinmiş oluruz. Eğer model aşırı uyum gösteriyor ise doğrulama ölçümleri sayesinde

modelin genelleme yeteniğini daha çok kaybetmemesi için erken durdurma işlemine başvurabiliriz.

Doğrulama ve test verilerinin dikkatli bir şekilde bölünmesi, bu görüntülerin birbirlerine karıştırılmaması gerekir. Amaç modelin görülmemiş veriler üzerinde test etmektir. Aksi takdirde, modelin doğrulama kayıplarının artmasına neden olabilir ve modelin istenen doğruluk oranlarına ulaşmasına engel olabilir, eğitim sırasında çeşitli problemlerin oluşmasına zemin hazırlayabilir. Tablo 5.3’ de her bir kategorideki eğitim, doğrulama ve test setine ayrılan nesne sayısını belirtmektedir.

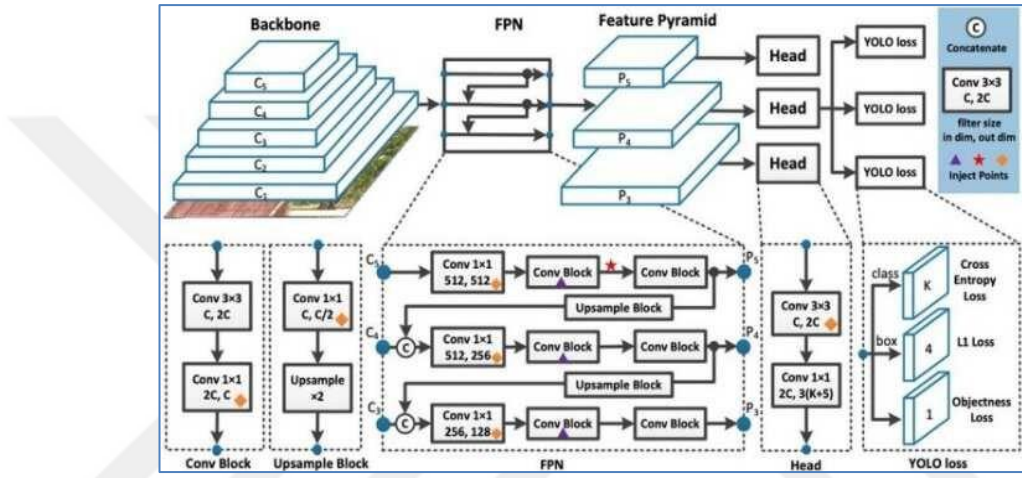
Tablo 5.3 : Eğitim, doğrulama ve test setindeki etiket sayıları

Sıra	Araç Sınıfları	Eğitim	Doğrulama	Test	Toplam
1	Plaka Numarası	47308	13202	5221	65731
2	Bisiklet	7680	2185	849	10714
3	Üç Tekerlekli Bisiklet-Tripörtör	6760	2117	748	9625
4	Motosiklet	11011	2941	1233	15185
5	ATV	506	127	53	686
6	Kar Aracı	104	26	14	144
7	Jetski	140	36	14	190
8	Araba	50424	13290	5791	69505
9	Van	9955	2781	1145	13881
10	Limuzin	331	95	43	469
11	Otobüs – Minibüs	11973	3200	1348	16521
12	Kamyon	18236	5135	2147	25518
13	Yarı Römorklu Kamyon	4852	1352	543	6747
14	Tankerli Kamyon	1707	489	176	2372
15	Çöp Kamyonu	517	141	55	713
16	Beton Mikserli Kamyon	539	143	52	734
17	İtfaiye Aracı	884	269	102	1255
18	Yol Süpürme Kamyonu	333	96	35	464
19	İş Makinesi	1515	411	173	2099
20	Çatallı Kaldıraç(Forklift)	1372	385	152	1909
21	Traktör	648	162	86	896
22	Segway	387	102	37	526
23	Fayton	499	154	56	709
24	Helikopter	656	182	73	911
25	Uçak	409	116	42	567
26	Gemi – Tekne	1569	405	171	2145
27	Tren	377	107	48	532
28	Yaya	143527	38919	16093	198539
29	Trafik Kazası	1639	428	201	2268
30	Scuter	581	160	70	811
31	Araba - Polis	1288	339	146	1773
32	Araba - Taksi	1459	388	161	2008
33	Van - Ambulans	1306	383	154	1843
34	Van - Polis	709	202	80	991
35	Van - Taksi	153	33	13	199
	Toplam	331.354	90.501	37.325	459.180

5.1.5. Algoritma Seçimi

YOLOv7[151] görüntü üzerindeki nesneleri, önceki sürümlere göre daha iyi doğruluğa sahip olduğundan modelimizi geliştirirken açık kaynak kodlu YOLOv7

algoritmasını tercih ettik. Çünkü YOLOv7 hem önceki sürümlere hem de diğer başka algoritmalarla göre daha hızlı ve daha doğru nesne tespiti yapabilme özelliğine sahiptir. YOLOv7 algoritmasının mimari yapısında bazı köklü yenilikçi değişiklikler yapılmıştır. Omurga yapısında ELAN[152] tekniği ile evrimleştirme işlemi gerçekleştirilerek elde edilen özellik haritalarının farklı boyutlara ölçeklendirilmesi sağlanmıştır. Bu farklı boyutlara ölçeklendirme işlemi sayesinde algoritmanın görüntü üzerindeki farklı boyutlardaki nesneleri tespit edebilme yeteneği iyileştirilmiştir. Şekil 5.2’ de YOLOv7 algoritmasının mimari yapısı görülmektedir[181].



Şekil 5.2: YOLOv7' nin Mimari Yapısı

YOLOv7 omurga, boyun ve kafa olmak üzere üç ana katmandan oluşmaktadır. Omurga katmanında görüntüler ELAN tekniği ile evrişimden geçirilerek özellik haritaları oluşturulur. Boyun katmanında özellik haritalarından üst ve alt örneklemeler yapılarak tahmin için bir sonraki katmana aktarılır. Algoritmanın kafa katmanında ise önceki katmandan gelen özellikler sınıflandırılır ve görüntü üzerinde yerleştirme hesaplamaları yapılır.

YOLOv7 hem yenilikçi teknikler hem de ölçeklendirilmiş farklı boyutlarda modeller sunmaktadır. YOLOv7'nin en küçük boyutlu modeli YOLOv7-tiny modelidir. Bu modelin aktivasyon katmanlarında sızdıran ReLU fonksiyonu kullanılmıştır. YOLOv7 ve YOLOv7x modelleri ise boyut olarak küçük modele göre biraz daha büyük olup daha fazla katmana ve ağırlık parametrelerine sahiptir. Bu modellerde algoritmanın performansını iyileştirmek için SiLU[66] aktivasyon fonksiyonu kullanılmıştır. SiLU aktivasyon fonksiyonunun her yerde pürüzsüz olması sayesinde eğitim sırasında daha iyi gradyan akışına yardımcı olur, modelin daha hızlı yakınsamasına ve daha iyi doğruluk oranlarının elde edilebilmesine olanak

sağlar. Diğer modeller ise YOLOv7-W6, YOLOv7-E6, YOLOv7-D6 ve YOLOv7-E6E modelleridir. Bu modeller daha çok evrişim katmanına ve daha çok ağırlık ve sapma parametresine sahiptir. Bu modeller ile eğitim yapabilmek için daha hızlı hesaplama yapabilen ve daha fazla hafıza gerektiren GPU' lara ihtiyaç vardır. Dolayısıyla kısıtlı kaynaklar ile mevcut veri setimizi bu modeller ile eğitebilmemiz biraz zor olacaktır.

Veri setimizi YOLOv7-tiny, YOLOv7 ve YOLOv7x modelleri ile eğitim yaptırabiliriz. YOLOv7-tiny modeli 6.2 milyon, YOLOv7 36.9 milyon, YOLOv7x ise 71.3 milyon ağırlık ve sapma parametresine sahiptir. Model boyutu arttıkça modelin mimari karmaşıklığı artacak, daha fazla ağırlık ve sapma parametresine sahip olacağından ekstra hafıza gerektirecektir. Çok sayıda ağırlık ve sapma parametrelerinin eğitim sırasında güncellenmesi modelin eğitim sürecinin uzamasına neden olacaktır. Ancak algoritma görüntü üzerindeki nesneleri daha iyi tespit edebilme yeteneğine sahip olacak ve tespit edilen nesnelerin daha yüksek doğruluk oranlarının elde edilmesine katkı sağlayacaktır.

5.1.6. Öğrenme Transferi

Modelimizi eğitmek için çok büyük miktarda veri setine ihtiyaç duyulmaktadır. Hem devasa veri setine ulaşmak hem de toplanan veri setini etiketlemek çok maliyetli bir iş olduğu için modeli eğitirken öğrenme transferinden faydalanmak, modeli eğitirken hem zamandan tasarruf sağlayacak hem de modelin istenen doğruluk oranını elde etmemize büyük ölçüde katkı sağlayacaktır. Devasa miktarda veriyi toplamak ve bu verileri etiketlemek zor bir görev olduğu gibi, bu kadar yüksek miktarda veri ile modeli eğitmek için çok yüksek kapasiteli GPU' lara ihtiyaç duyulmakta ve eğitim günler hatta aylar sürebilmektedir. Dolayısıyla, modelimizi eğitirken YOLOv7'deki önceden eğitilmiş olan modellerden faydalanmak bize büyük ölçüde zamandan tasarruf sağlayacaktır. Seçmiş olduğumuz YOLOv7 algoritmasında önceden eğitilmiş modeller mevcut olup modelimizi eğitmeden önce, bu önceden eğitilmiş olan modellerden parametre aktarımı yaparak modelin kısa sürede yüksek doğruluk oranlarına erişmesine katkı sağlayacağız. YOLOv7 farklı ölçekte modeller sunduğundan her ölçekte önceden eğitilmiş modeller hazırlanmıştır.

Özet olarak modelimizi eğitirken YOLOv7'de mevcut olan önceden eğitilmiş olan modellerden faydalanarak öğrenme transferi yapmamız modeli eğitirken

zamandan tasarruf sağlayacak, modelin doğruluk oranını artıracak ve büyük miktarda veri ihtiyacını azaltacaktır.

5.1.7. İhtiyaç Duyulan Donanım ve Yazılımlar

Modeli eğitmeden önce bilgisayarımızın öncelikle iyi bir ekran kartına ihtiyaç vardır. Ekran kartı olmadan CPU kullanılarak modeli bu devasa veri setini ile eğitmek bir hayli zaman alacaktır. Mümkün olduğu kadar en iyi ekran kartına sahip olmamız gerekir. Bu çalışmada NVIDIA GeForce RTX 3060 GPU kullanılmıştır. Modeli eğitmeden önce bilgisayarımıza bir dizi yazılımın kurulması gerekir. Öncelikle YOLOv7 için uygun python sürümü kurulmalıdır. Çalışmamızda python 3.9.18 sürümü kurulmuştur. Sonrasında Anaconda' yı kurmak bize artı katkı sağlayacaktır. Çünkü sürüm değişikliklerinden etkilenmemek için Anaconda yazılımının içinde Conda paketi ile algoritmamıza özel sanal çevreler oluşturabiliriz. Çünkü, başka bir modeli eğitirken o modelin ihtiyaç duyduğu yazılımlarda sürüm güncellemelerinde problemler ile karşı karşıya kalabiliriz. Sonrasında YOLOv7 paketi <https://github.com/WongKinYiu/yolov7> adresinden eğitim yaptırılacak olan bilgisayara indirilir. Paketlerin içindeki requirement.txt içerisindeki algoritmanın çalışabilmesini sağlayacak gerekli yazılımlar güncellenir. YOLOv7 modelimizi eğitmek için, derin öğrenme uygulamalarında yaygın olarak kullanılan açık kaynak yapay zeka kütüphanesi olan PyTorch yazılımına ihtiyaç duyulmaktadır. Çalışmamızda kullanmış olduğumuz donanımlar ve yazılımlar Tablo 5.5' de gösterilmiştir.

Tablo 5.4 : Kullanılan donanımlar

İşletim Sistemi	Windows 11 Home 64 bit
Kütüphane	PyTorch CUDA 11.8
Algoritma	YOLOv7
Yazılım	Python 3.9.18, Anaconda
Laptop	Monster Tulpar
İşlemci	11. Generasyon intel i7-11800H 8 çekirdek
Bellek	40 GB DDR4 RAM 3200 MHZ
Depolama Alanı	500 GB SSD + 1TB SSD
Ekran Kartı	Nvidia GeForce RTX 3060 6GB

5.1.8. YOLOv7 Algoritmasında Kullanılan Hiper Parametreler

Modelin eğitime başlamadan önce YOLOv7 paketlerinde bulunan bazı parametre ayarlarının yapılabilmesi için bu parametrelerinin modele ne gibi katkı sağladığı konusunda bazı bilgileri öğrenmemiz modelin performansının

artırılmasında bize yardımcı olabilecektir. YOLOv7 çok sayıda yenilikçi teknik kullandığından modelin performansını iyileştirmek için çok sayıda parametre sunmuştur. Bu parametrelerde yapılan ayarlamalar veri setine bağlı olarak değişebilir. Dolayısıyla bu parametrelerde doğru bir şekilde ince ayar yapılması modelin doğruluk oranlarının artmasına yardımcı olabilir. Modelin eğitime geçilmeden önce bu parametrelerinin yapılması gerekir. Modelde istenen performans alınana kadar bu parametreler üzerinde her seferinde ince ayarlar yapılarak çok sayıda eğitim denemeleri yapmak mümkündür. Bu parametrelerinin modele ne gibi katkı sağlayacağını açıklamaya çalışalım.

a) Lr0

Başlangıç öğrenme oranını ifade eder. Varsayılan değer 0.02'dir. Başlangıç öğrenme oranı parametresi ağırlıkların ne kadar hızla yakınsayacağını belirler. Eğer başlangıç öğrenme oranı çok büyük olursa optimum sonuca ulaşmak atlanabilir. Eğer bu parametre çok düşük olursa da en iyi sonuca ulaşmak için çok fazla eğitim döngüsüne ihtiyaç duyulur. Dolayısıyla öğrenme oranı parametresine doğru bir değer girilmesi modelin istenen doğruluk seviyelerine ulaşmasında oldukça önemli bir rol oynayacaktır. Çalışmamızda öğrenme oranını 0.0001 olarak belirlemeyi tercih ettik.

b) Lrf

Eğitimin sonunda olan öğrenme oranıdır. Varsayılan değer 0.2'dir. Öğrenme oranı lr_0 değerinden başlar ve $lr_0 * Lrf$ değerine ulaşmaya kadar yavaş yavaş azaltılır. Eğitimi daha büyük öğrenme oranları ile başlatma, ağırlıkların daha büyük değerler ile güncelleyerek ağırlıkların daha hızlı yakınsamasına yardımcı olur. Ancak öğrenme oranının devamlı büyük olması ağırlıkların optimum sonuca ulaşmasını kaçırabilir, optimum sonuca yaklaşırken eğitim süreci dengesizleşebilir. Modelin aşırı uyum göstermesine neden olabilir. Dolayısıyla öğrenme oranı eğitimin başlangıcından sonuna doğru azar azar düşürülmesi, ağırlık parametrelerinin optimum şekilde güncellenmesine, kayıpların minimum bir değere ulaşmasına katkı sağlayabilir ve doğrulama kayıplarındaki yaşanabilecek olan dalgalanmaları önleyerek ağırlıkların daha iyi genelleme yapabilmesine yardımcı olur.

c) Momentum

Momentum, stokastik gradyan inişi ve Adam gibi optimizasyon algoritmalarında kullanılan bir parametredir. Momentum eğitim sırasında ağırlıkların yakınsamasını hızlandıran ve oluşacak kayıpların dalgalanmasını azaltan bir

metotdur. YOLOv7’ de bu parametre hem Adam hem stokastik gradyan inişı optimizasyon algoritması ile birlikte kullanılır. Varsayılan değeri 0.937’ dir.

d) Weight_decay

Modelin aşırı uyumunu önlemek için kayıp fonksiyonlarında ağırlıklara eklenen bir cezalandırma değeridir. Büyük ağırlıklar modelin karmaşıklığını artırarak modelin eğitim verilerinde aşırı uyum göstermesine neden olur ancak doğrulama ve test verileri üzerinde genelleme yapabilme yeteneğini gösteremez. Dolayısıyla büyük ağırlıklara ceza terimi eklemek modelin performansını iyileştirebilecektir. Varsayılan ağırlık cezalandırma değeri 0.0005’dir.

e) Iou_t

Görüntü üzerinde tahmin edilen sınırlayıcı kutu ile gerçek sınırlayıcı kutu arasındaki kesişim değeridir. Eğitim aşamasında kayıp fonksiyonlarının hesaplanmasında model belirli bir değerin altındaki tahminler elenerek yanlış pozitif olarak, belirlenen eşik değerinin üstündeki tahminler ise doğru pozitif olarak değerlendirilir. Varsayılan iou_t değeri 0.20’dir. Bu eşik değerinin üzerindeki ve altındaki tahminler kesinlik ölçümünün değerlendirilmesine katkı sağlar. Bu eşik değeri sayesinde modelin performansı hakkında bilgi sahibi oluruz. Ancak çalışmanın türüne ve önemine göre bu eşik değeri değiştirilebilir. Daha kritik öneme sahip olan uygulamalarda 0.25, 0.50, 0.75 gibi eşik değerleri kullanılabilir. Daha az önem arz eden uygulamalarda ise bu eşik değeri düşürülebilir.

f) Hsv_h

Girdi görüntüsünün renk tonunun derecesini kontrol eden veri artırma tekniklerinden biridir. Görüntüdeki piksellerin renk tonu değerleri kaydırılarak görüntünün orjinal yapısı ve içeriği korunur. Görüntünün parlaklık değerlerine ince ayar çekilerek görüntülerin çoğaltılmasının amacı modelin farklı yapay ortamlara uyum sağlamasını olanak sağlamak, farklı ortamlarda da nesne tespitini kolaylaştırarak modelin performansının ve genelleme yapabilme yeteneğinin iyileştirilmesine yardımcı olmaktır. Varsayılan değeri hsv_h değeri 0.015’dir.

g) Hsv_s

Eğitim sırasında girdi görüntülerine uygulanan doygunluk ayarının derecesini kontrol eden parametredir. Daha düşük doygunluğa sahip görüntü daha soluk, yoğunluğu artırılmış olan görüntü ise daha canlı olur. Görüntünün içeriği ve orjinal yapısı korunarak renk yoğunluğunda varyasyonlar oluşturulur. Rastgele yoğunlukta

yapılan veri artırma işlemleri modelin performansını kayda değer oranda iyileştirmeye yardımcı olur. Varsayılan hsv_v değeri 0.7'dir.

h) Hsv_v

Görüntünün parlaklık ayarının derecesini kontrol eden bir parametredir. Görüntünün parlaklığı rastgele ayarlanarak modelin farklı parlaklık ortamlarında nesnelerin daha doğru tespit edilebilmesini sağlar. Örneğin gece karanlık ortamlarda veya gündüz güneşin aşırı parlak olduğu ortamlarda nesnelerin görüntü üzerinde tespit edilebilmesine olanak sağlar. Aşırı daha iyi genelleme yapabilmesine, verilerin artırılması sayesinde aşırı uyumun oluşabilmesini önlemeye yardımcı olur. Varsayılan hsv_v değeri 0.4'dür.

i) Degrees, Translate, Scale, Shear, Perspective Parametreleri

YOLOv7 içerisinde yer alan degrees, translate, scale, shear ve perspective parametreleri veri artırma tekniğinde kullanılan parametrelerdir. Degrees parametresi ile görüntü kendi eksenini etrafında rastgele derecelerle döndürülerek görüntüler çoğaltılır. Translate parametresi ile görüntü içeriği yatay ve dikey olarak rastgele sağa, sola, yukarı, aşağı kaydırılarak veri artırma işlemi gerçekleştirilir. Varsayılan değeri 0.2'dir. Scale parametresi ise eğitim sırasında girdi görüntüleri veya görüntülerdeki nesneler rastgele büyütülerek veya küçültülerek farklı boyutlara ölçeklendirilmesi gerçekleştirilir. Farklı boyutlara yapılan ölçeklendirme sayesinde modelin nesnelerin farklı ölçeklerde olanlarını da tespit edilebilme imkanı verir. Veri setinin zenginleştirilmesini sağlar. Shear parametresi görüntü içerisindeki nesnenin yapısında yapılan traşlama ile geometrik dönüşümler elde edilir. Nesnelerin perspektifinde değişkenler oluşturularak veri artırma işlemi yapılır. Perspective değişkeni ise, görüntünün perspektifi ayarlanarak görüntünün farklı açılarla görülebilir olmasını sağlar.

j) Flipud- Fliplr

Veri artırma tekniğinde kullanılan flipud görüntüdeki nesneleri yukarı aşağı veya aşağı yukarı yansıtırken, fliplr parametresi de görüntüyü sağa sola ve sola sağa olacak şekilde yansıtarak eğitim sırasında görüntüler artırılır.

k) Mosaic

Mozaik veri artırma yöntemi birkaç görüntüyü birleştirip modelin eğitiminde tek bir görüntü hale getirilmesi olayıdır. Bu teknik, ortamında kalabalık veya nesnelerin birbirlerine örtüşmüş vaziyette olması gibi karışık durumlarda modelin özellikle görüntüde araçların çok yoğun olduğu durumlarda araçların tespit

edilebilmesine yardımcı olmak için modelin bu karşışık seneryoları öğrenebilmesine yardımcı olabilmektedir. Geliştirmiş olduğumuz modelde trafikteki araçlar seyrek olmayabilir. Araç trafiği yoğun olabilmekte ve bu yoğun trafikteki araçların tespiti zor olabilmektedir. Mozaik veri artırma işlemi modelin doğruluğunu iyileştirmede yardımcı olabilmektedir. Varsayılan mosaic değeri 1.0' dır.

l) Mixup

Bu teknikte, aynı kategoride yer alan iki görüntü birleştirilerek yeni örnekler oluşturulur. Varsayılan değer mixup değeri 0.15'dir. Mixup parametresinin değeri iyi ayarlanmaz ise ağıın yakınsama sürecini uzatabilir. Modelin genelleme yeteneğinde başarısızlıklar görülebilir, modelin yetersiz öğrenmesine neden olabilir. Ayrıca, çalışmanın türüne göre bu parametrenin kullanılması modelin performansının iyileştirilmesine yardımcı olabilir.

m) Copy_paste

Eğitim setinin çeşitlendirilmesini sağlayan ve veri setinin boyutunu artıran bir veri artırma tekniğidir. Bir görüntünün tamamı değil de kısmi parçaları aynı görüntü içerisinde farklı konumlara kopyalama işlemi yapılarak eğitim veri seti artırılır. Varsayılan copy_paste değeri 0.0'dır. Eğitim sürecini uzatır.

n) Paste_in

Bu teknikte görüntünün tamamı orijinal görüntü içerisine kopyalanarak yeni eğitim setleri oluşturulur. Varsayılan paste_in değeri 0.15'dir. Eğitim sürecini uzatır.

o) Loss_ota

Özel bir kayıp fonksiyonudur. Hesaplanması standart kayıp fonksiyonlarına göre daha masraflıdır. Varsayılan loss_ota değeri 1'dir. Hesaplanması masraflı olduğundan eğitim sürecini önemli oranda artırır.

Sonuç olarak YOLOv7 algoritması ile eğitime başlamadan önce, geliştirilecek olan modelin türüne göre bu parametreler üzerinde ince ayarlar yapılması modelin performansının artırılmasında önemli rol oynar. Parametreler kaba değerler ile ayarlanması hem modelin eğitim sürecini uzatabilir hem de modelde istenen doğruluk seviyelerine ulaşılmasını zorlaştırabilir.

5.2. Modelin Eğitilmesi

Seçmiş olduğumuz YOLOv7 algoritması boyutlarına göre birden fazla model sunmaktadır. Modelin boyutu büyüdükçe modelin sahip olduğu parametre sayısı arttığından eğitim süresi oldukça uzamaktadır. Kullanacak olduğumuz bilgisayarda,

modelin bir eğitim döngüsü neredeyse 5-6 saatlere kadar sürebildiğinden 300 döngülü bir eğitim 2.5 ay gibi süre alabilir. Dolayısıyla model eğitilirken YOLOv7-tiny, YOLOv7 ve YOLOv7x modelleriyle denemeler yapılmıştır. Modeldeki parametreler değiştirilerek ve farklı metotlar kullanılarak modelin performansı iyileştirilmeye çalışılmıştır. Model farklı teknikler ve metotlar kullanılarak eğitime çalışılmıştır. Modelin performansını etkileyen problemler analiz edilerek çok sayıda denemeler yapılmış ve modelin performansında optimum sonuca ulaşıncaya kadar her seferinde yöntemler değiştirilmiştir.

5.2.1. Modelin Eğitiminde Kullanılan Parametreler

YOLOv7 modelinin eğitiminde kullanılan hiper parametreler gibi çok sayıda parametreye de sahiptir. Bu parametreler öğrenme transferinin yapılıp yapılmayacağını, eğitim döngüsünün ne sayıda olacağı, eğitimin hangi donanımda gerçekleştirileceği gibi konuları belirlenmesini sağlar. Bu parametreleri bu bölümde açıklamaya çalışacağız.

a) Workers

Eğitimde kullanılacak işlemci sayısını belirtir. Bilgisayarın sahip olduğu özelliklere göre işlemci sayısını artırmak modelin eğitimini hızlandırıp eğitim süresini kısaltacaktır. Modelin eğitiminde 8 veya 4 olarak belirlenmesi eğitimin sürecinin azaltılmasına yardımcı olacaktır.

b) Device

Eğitimin hangi donanımda gerçekleştirileceğini belirtir. Modelin eğitimi cpu ile olacaksa parametre değişkenine cpu yazılır. Eğer bilgisayarda GPU var ise device parametresi 0 olarak girilir. Büyük veri setini cpu ile eğitmek eğitim sürecini çok ciddi oranda artıracaktır. Eğitimin GPU ile yapılması eğitim sürecini kısaltır.

c) Batch_size

Bu parametre eğitimin her bir döngüsünde kullanılan eğitim örneklerinin sayısını ifade eder. Eğitimde kullanılacak veri seti bu parametre ile belirlenen küçük gruplara bölünür. Her bir eğitilen gruptan sonra ağırlıklar ve sapmalar güncellenir. Gruptaki örnek sayılarının büyük olması eğitim sürecini hızlandırır ancak daha fazla hafıza gerektirir. Gradyanların daha doğru tahmin edilebilmesine, ağırlık ve sapma güncellemelerinin daha istrarlı yapılabilmesine, eğitim performansının daha iyi olmasına olanak sağlar. Örnek sayılarının küçük olması hafıza kullanımını azaltır ancak eğitim sürecini ciddi oranda artırır. Ağırlık ve sapma güncellemeleri daha sık

olur, gradyanların doğrulukların tahmin edilebilmesi zorlaşır. Ancak ağıın genelleme yeteneğini iyileştirir.

Grupta yer alan örneklerin boyutu eğitimde kullanılacak ekran kartının özelliklerine göre değişiklik gösterebilir. Eğitimde grupta yer alan örnek sayısı olarak genellikle 16 değeri kullanılır.

d) Data

Eğitim, doğrulama ve test verilerinin, kategori sayısı ve çalışmada belirtilen sınıf isimlerinin belirtildiği dosyanın yolu bu parametrede gösterilir.

e) Img

Modelde kullanılan girdi görüntüleri değişiklik gösterebilir. Model eğitilirken bütün görüntülerin aynı boyutlarda olması gerekmektedir. Bu parametre ile girdi görüntüleri istenilen boyuta yeniden boyutlandırılır. Girdi görüntülerin büyük olması eğitimin sürecini uzatabilmektedir. Çünkü girdi görüntülerinin boyutlarının artması GPU’ daki hesaplama maliyetini artırır, hafızada taşmaların oluşumuna zemin hazırlayabilir. Girdi görüntülerinin boyutlarının küçültülmesi eğitim sürecinin hızlandırılmasına katkı sağlar ancak görüntüdeki küçük ayrıntıların kaçırılabilmesine neden olabilir. Bu da görüntüdeki nesnelerin tespitini zorlaştırabilir.

f) Epoch

Eğitimin döngü sayısını belirtmektedir. Eğitimde makul bir döngü sayısı belirlenebilir. Model eğitilirken eğitim ve doğrulama kayıpları analiz edilerek erken durdurma işlemi yapılabilmektedir. YOLOv7 algoritmasında eğitimin her 25 döngüsünde bütün ağırlık ve sapma parametreleri kayıt edilir. Bu sayede doğrulama kayıpları ve doğruluk değerleri analiz edilerek modelin aşırı uyum gösteren döngüsünden itibaren modelin daha fazla genelleme yeteneğini kaybetmemesi ve modelin doğruluğunu azaltmamak için erken durdurma işlemi yapılabilir.

g) Cfg

YOLOv7 algoritmasında hangi mimari yapının kullanılacağını belirler. YOLOv7 farklı boyutlarda ölçeklendirilmiş modeller sunmaktadır. YOLOv7-tiny, YOLOv7, YOLOv7x, YOLOv7w6 gibi modellerin her biri farklı katmanlara, farklı boyutlarda ağırlık ve sapma parametrelerine sahiptir. Eğitilecek olan modelin türüne göre hangi model eğitilecekse o modelin mimari yapısı seçilmelidir.

h) Weights

Modelin hızlı bir şekilde eğitilebilmesi için öğrenme transferinden faydalanmak eğitim sürecini ciddi oranda azaltabilir. Ağıın daha iyi öğrenebilmesine,

istenen doğruluk seviyelerine kısa sürede ulaşabilmesine, ağın daha iyi genelleme yapabilmesine olanak sağlayabilir. YOLOv7 öğrenme transferinden faydalanılması için önceden eğitilmiş olan modeller sunmaktadır. Öğrenme transferinden faydalanılacak ise önceden eğitilmiş olan modelin yolu bu kısımda belirtilebilir. Öğrenme transferi yapılmasının uygun olmayacağı düşüldüğü durumlarda ise bu kısım boş bırakılabilir.

i) Image_weights

Veri setinin dengesiz olduğu durumlarda, eğitim süresi içerisinde veri setindeki azınlık sınıfındaki görüntüler ağırlık parametresine göre eğitim görüntüleri çoğaltılır. Azınlık sınıfındaki görüntülere daha büyük ağırlık parametreler uygulanarak örneklerin çoğaltılması veri setinin dengesiz olma durumunu önler, modelin bu sınıflardaki örnekleri daha iyi öğrenebilmesini sağlar. Modelin genelleme yapabilme yeteneğini iyileştirir.

j) Label_smoothing

Etiket yumuşatma modelin tahminleme işlemini gerçekleştirirken aşırı güven duymasını önleyerek ağın performansını ve genelleme yapabilme yeteneğini iyileştirmek için kullanılan bir regülezasyon tekniğidir. Bu teknikte eğitim süresi içerisinde gerçek etiket değerleri yumuşatılır. Doğru tahmin edilen değerlerin olasılık değerleri hafif bir şekilde azaltılır, diğer sınıfların değerleri de çok hafif bir şekilde artırılarak ağın daha iyi genelleme yapabilmesi sağlanır. Küçük veri setlerinde kullanılması uygundur. Büyük veri setlerinde yetersiz öğrenmeye neden olabilir.

k) Adam

YOLOv7 algoritmasında varsayılan optimizasyon algoritması stokastik gradyan iniş algoritmasıdır. Adam Adagrad ve RMSprop algoritmalarının avantajlarını birleştirerek geliştirilen ve yaygın olarak kullanılan bir optimizasyon algoritmasıdır. Adam öğrenme oranı parametresini uyarlamak için gradyanların birinci ve ikinci momentumlarını hesaplar. Adam algoritması ağın yakınsamasını stokastik gradyan iniş algoritmasından daha hızlı gerçekleştirir. Öğrenme oranının parametrelere göre ayarlanması ağın istenen doğruluk oranlarına daha hızlı ulaşılmasını sağlar. Adam momentum değerlerinin her bir parametresi için hesaplamasını yapabilmek için daha fazla hafıza gerektirir.

l) Freeze

Öğrenme transferi yapıldığında önceden eğitilmiş modelin katmanların dondurulması bu parametre ile yapılır. Model içerisinde öğrenilmiş olan ağırlık ve

sapma parametreleri eğitim süresi içerisinde dondurularak güncellenmesi önlenir. Katmanların dondurulması eğitim sürecini kısaltır, küçük veri setlerinde aşırı uyumu önler. Ancak veri setinin boyutu ve çeşitliliği arttıkça modelin yetersiz öğrenmesine neden olabilir. Dondurulacak katman sayısı bu parametre ile belirlenir.

5.2.2. Eğitilen Modellerin Performansının Analiz Edilmesi

Çalışmaya başlamadan önce modeli eğitebilmek için güçlü bir veri setine sahip olmamız gerekir. Küçük veri setlerinde model, eğitim verilerinde aşırı uyum gösterebilir. Dolayısıyla model doğrulama ve test verileri üzerinde yeterince performans ve genelleme yapabilme yeteneğini gösteremeyebilir. Görüntü üzerinde seyir halinde olan araçları gerçek zamanlı olarak tespit edebilmek ve bu araçları çok sayıda sınıfa ayırmak gerçekten zorlu bir süreç gerektirir. Model eğitilirken çok sayıda problemlerle karşılaşmıştır. Modelin eğitim verileri üzerinde aşırı uyumunu azaltmak için veri setine mümkün olduğu kadar çok sayıda araç görüntüsü eklenmiştir. Bazı azınlıkta kalan araç görüntüleri zenginleştirilmeye çalışılmıştır. Modelin hem eğitim verilerini öğrenebilmesi hem de bu yeteneğini doğrulama verileri üzerinde gösterebilmesi gerekir. Dolayısıyla kullandığımız nesne tespit algoritmasının mimari yapısının çok hafif olmaması gerekir. Algoritmada kullanılan evrişimsel katmanları ile ağırlık parametrelerinin optimum bir değere ulaşabilmesi için mimari yapının katman derinliği ve modelin kanal derinliği artırılabilir. Yolov7-tiny modeli ile veri seti eğitilirken modelin öğrenmesinde yetersiz kaldığı anlaşılmıştır. Model eğitilirken aşırı uyumu azaltmak için ölçeklendirme, yansıtma, kaydırma, mosaic ve karıştırma gibi çok sayıda teknikten faydalanılmaya çalışılmıştır. Modelin eğitiminde optimum sonuca ulaşmak için mümkün olduğu kadar en iyi öğrenme oranı değeri ile eğitim gerçekleştirilmiştir. Öğrenme oranlarının yüksek olması modelin yakınsamasını yavaşlatmıştır.

Model eğitilirken dikkat edilmesi gereken bir diğer konu ise görüntü verilerinin doğru bir şekilde etiketlenmiş olmasıdır. Etiketlenen nesnelerin doğru sınıflara hitap etmesi modelin doğruluk oranlarının artmasında önemli bir rol oynar. Çünkü model doğru tahmin ettiği sınıfı yanlış pozitif tahmin olarak algılaması modelin kesinlik değerinin yükselmesini engeller ve modelin ağırlık parametrelerinin değişmesine sebebiyet verebilir. Sınıflandırma hatalarından kaçınmak için verilerin doğru etiketlenmesi önemlidir.

Model eğitilirken çok sayıda eğitim denemesi yapılmıştır. Eğitim sonucu elde edilen kesinlik ve duyarlılık eğrileri ile hata matrisi detaylı bir şekilde analiz edilmiştir. Kesinlik ve duyarlılık eğrileri modelin performansının ne kadar istrarlı olduğu konusunda bilgi edinmemizi sağlamıştır. Hata matrisi sınıflandırma hatalarını gösteren bir grafikdir. Eğitim sonucunda yaşanan problemlere çözüm bulmak için hata matrisi detaylı bir şekilde incelenmiş, hangi sınıfların ne oranda ne sıklıkla hatalı sınıflandırma yaptığını öğrenmemize olanak sağlamıştır. Hata matrisi her eğitim sonucu tekrar değerlendirilerek hatalı sınıflandırma yapan bazı sınıftaki görüntü verileri tek sınıfta birleştirilmiş ve tekrar eğitim yaptırılmıştır. Yeni yapılan eğitimde modelin doğruluk oranlarında kayda değer oranında yükselişlerin olduğu gözlemlenmiştir.

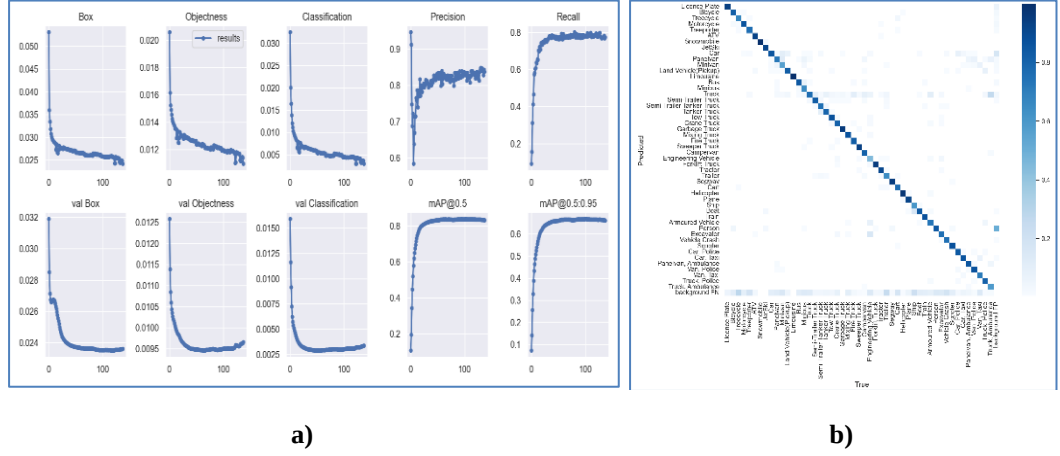
5.2.3. 49 Sınıftan Oluşan Veri Setinin Eğitilmesi ve Değerlendirilmesi

Model eğitilirken YOLOv7x modeli tercih edilmiştir. Modelin eğitiminde kullanılan parametreler tablo 5.5’ de belirtilmiştir.

Tablo 5.5 : Modelin eğitiminde kullanılan parametre değerleri

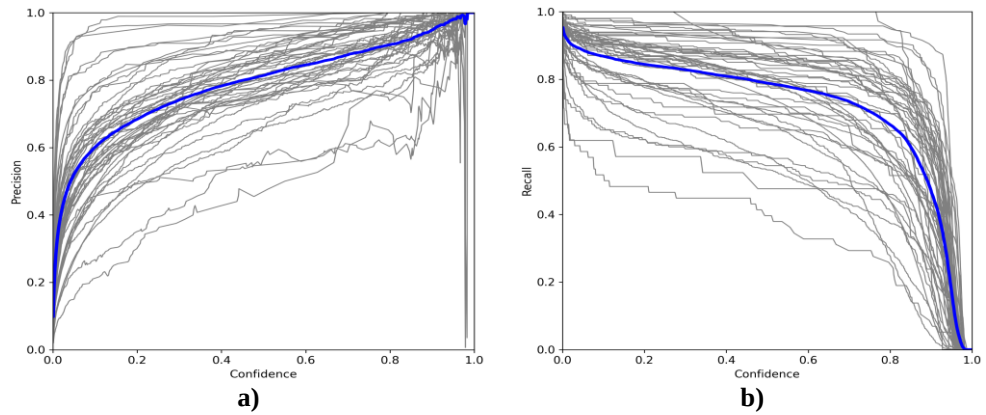
Optimizasyon algoritması	Stokastik Gradyan İnişi	Hsv_h	0.015
Weights	Öğrenme Transferi Var	Hsv_s	0.7
Batch-size	7	Hsv_v	0.4
Image_size	448	Degrees	0.0
Image_weights	Kullanılmadı	Translate	0.2
Workers	8	Scale	0.9
Label_smoothing	Kullanılmadı	Shear	0.0
Freeze	0	Perspective	0.0
Lr0	0.001	Flipud	0.0
Lrf	0.01	Fliplr	0.5
Momentum	0.937	Mosaic	1.0
Weight_decay	0.0005	Mixup	0.0
Iou_t	0.2	Copy_paste	0.0
Loss_ota	0	Paste_in	0.0

Çalışmada 49 kategoriden oluşan araç görüntüleri YOLOv7x algoritması ile eğitime çalışılmıştır. Tablo 5.5’ deki parametre değerleri ile 137 döngülük bir eğitim yapılmıştır. Eğitim sırasında doğrulama verileri üzerinde ölçülen doğrulama kayıpları ile mAP@0.5 değerleri analiz edilmiştir. Eğitimde 0.839 mAP@0.5 değerlerine ulaşılmış ancak belirli bir döngüden sonra modelin performansında düşüşler olduğu ve doğrulama kayıplarının arttığı gözlemlenmiştir. Bu eğitimde elde edilen sonuçlar ve hata matrisi şekil 5.3’ de gösterilmiştir.



Şekil 5.3 : (a)Eğitim ve doğrulama sonuçları (b)Hata matrisi

Doğrulama sonuçları analiz edildiğinde belirli bir döngüden sonra sınıflandırma ve nesnellik kayıplarında artışlar, doğrulama değerlerinde de düşüşler yaşandığı görülmüştür. Şekil 5.3b’ de eğitim sonucu elde edilen hata matrisi gösterilmiştir. Hata matrisi sınıflandırmaların doğru yapıp yapılmadığını, nesnelerin ne kadar doğru olarak sınıflandırıldığını, sınıflandırma hatalarının daha çok hangi kategorilerde olduğu hakkında bilgi edinmemize olanak sağlar. Dolayısıyla eğitim sonrası hata matrisinin detaylı bir şekilde analiz edilmesi, model hangi nesneleri tespit ederken hatalı sınıflandırma yaptığının belirlenmesine yardımcı olacaktır. Gerçekleştirilen bu eğitimden sonra elde edilen hata matrisi incelendiğinde modelin bazı sınıflardaki nesneleri hatalı sınıflandırdığı gözlemlenmiştir. Bazı kategorideki nesnelerin benzerlik göstermesi modelin hatalı sınıflandırma yapmasına neden olduğu düşünülmüştür. Şekil 5.4’ de eğitimden sonra elde edilen kesinlik ve duyarlılık eğrilerini göstermektedir.



Şekil 5.4 : (a) Kesinlik eğrisi, (b) Duyarlılık eğrisi

Şekil 5.4a’ da eğitim sonrası güven skoru ile kesinlik değerlerinin karşılaştırma grafiği gösterilmiştir. Grafikte yüksek güven skorlarında bile kesinlik

değerlerinin çok istrarlı olmadığı gözlemlenmiştir. Çok düşük güven skorlarında yüksek kesinlik değerlerinin elde edilmesi arzu edilen bir durumdur. Yapılan eğitimde düşük güven skorlarında bile kesinlik değerlerinin düşük olduğu görülmüştür. Aynı şekilde duyarlılık ile güven skorunun karşılaştırıldığı şekil 5.4b’deki grafik analiz edildiğinde yüksek güven skorlarında nesne tespiti giderek azalmış, düşük güven skorlarında ise nesne tespitlerinde istenen istikrarın sağlanamadığı gözlemlenmiştir.

5.2.4. 42 Sınıftan Oluşan Veri Setinin Eğitilmesi ve Değerlendirilmesi

49 kategoriden oluşan veri setimizi YOLOv7x modeli ile eğitmeyi denedik. Ancak modelimizin doğruluk oranları istediğimiz seviyeye ulaşamadı. Eğitim sonrası elde edilen kesinlik ve duyarlılık eğrileri analiz edildiğinde, ağın performansının istikrarlı olmadığı anlaşılmıştır. Hata matrisi incelendiğinde kayda değer oranında hatalı sınıflandırmaların olduğu tespit edilmiştir. Dolayısıyla hatalı sınıflandırma yapan sınıfların birbirine benzer yönleri olduğundan bu sınıfları birleştirerek modelin nasıl bir performans elde edeceğini görmek istedik. Sınıflandırma hataları yapan kategorileri birleştirerek yeni sınıf sayısı 42 olmuştur.

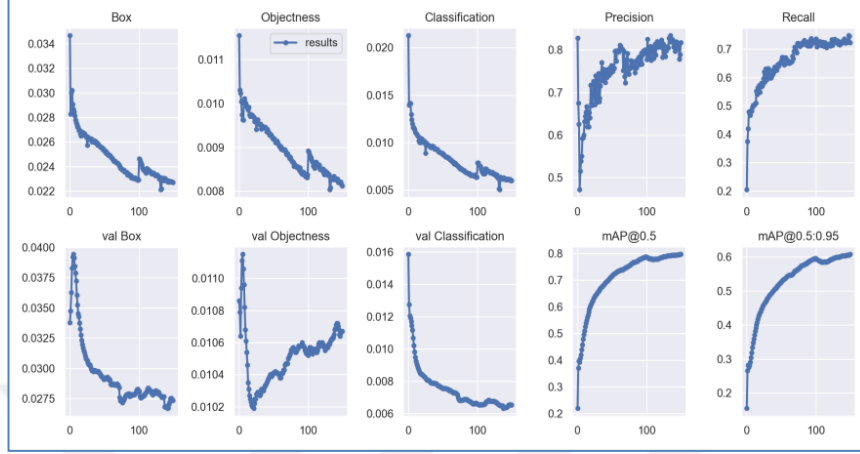
Model aynı şekilde YOLOv7x modeli ile tablo 5.6’daki parametrelerle eğitilmiştir.

Tablo 5.6: Modelin eğitiminde kullanılan parametre değerleri

Optimizasyon algoritması	Stokastik Gradyan İnişi	Hsv_h	0.015
Weights	Öğrenme Transferi Var	Hsv_s	0.7
Batch-size	4	Hsv_v	0.4
Image_size	480	Degrees	0.0
Image_weights	Kullanılmadı	Translate	0.2
Workers	8	Scale	0.9
Label_smoothing	Kullanılmadı	Shear	0.0
Freeze	0	Perspective	0.0
Lr0	0.01	Flipud	0.0
Lrf	0.2	Fliplr	0.5
Momentum	0.937	Mosaic	1.0
Weight_decay	0.0005	Mixup	0.15
Iou_t	0.2	Copy_paste	0.0
Loss_ota	1	Paste_in	0.15

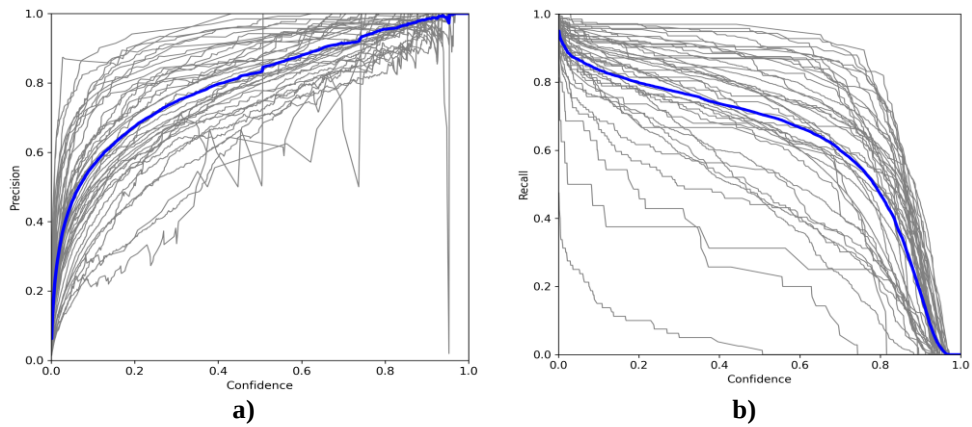
Bölüm 5.3.1.’de yapılan eğitimden farklı olarak bu eğitimde görüntü çözünürlülüğü ile öğrenme oranları artırılmıştır. Mixup veri artırma tekniği ile yeni bir kayıp fonksiyonu olan loss_ota tekniği kullanılmıştır. Görüntü çözünürlülüğün artırılmasının amacı görüntüdeki küçük detayları daha iyi yakalayıp daha fazla

nesnenin tespit edilebilmesini sağlamaktır. Ancak görüntü çözünürlüğünü arttırdıkça eğitim süreci ciddi oranda artırmıştır. Yeni kayıp hesaplama tekniği olan loss_ota tekniğinin kullanılması da eğitim sürecini uzatmıştır. Bu çalışmada elde edilen eğitim ve doğrulama sonuçları şekil 5.5’ de gösterilmektedir.



Şekil 5.5 : Eğitim ve doğrulama sonuçları

Bu deneyde 150 döngülük bir eğitim gerçekleştirilmiştir. Bu eğitimde veri setini 42 kategoride toplamamıza rağmen bölüm 5.3.1’ deki çalışmadan daha kötü bir performans elde edilmiştir. Modelin mAP@0.5 değeri 0.80’ in altında kalmıştır. Doğrulama sonuçlarının nesnellik kayıplarında ciddi oranda kayıplar yaşanmıştır. Eğitim sırasında model eğitim verileri üzerinde performans gösterirken doğrulama verileri üzerinde aynı performansı gösterememiştir. Eğitim sırasında model eğitim verileri üzerinde aşırı uyum içinde olup doğrulama verileri üzerinde genelleme yapabilme yeteneğini gösterememiştir. Eğitimden elde edilen kesinlik ve duyarlılık eğrileri Şekil 5.6’ da gösterilmektedir.



Şekil 5.6 : (a) Kesinlik eğrisi, (b) Duyarlılık eğrisi

Şekil 5.6’ daki kesinlik ve duyarlılık eğrileri analiz edildiğine eğitilen modelin daha da istikrarsız hale geldiği, ortalama güven skorlarında bile modelin

doğruluk seviyelerinin çok düşük kaldığı görülmüş, modelin görüntü üzerindeki nesneleri tespit edebilme yeteneğini koruyamadığı anlaşılmıştır. Şekilde 5.6b’ deki duyarlılık eğrisinden de anlaşılabacağı üzere modelin nesneleri tespit etmede sorunlarla mücadele ettiği, düşük güven skorlarında bile nesneleri tespit etmede sorunlar yaşandığı gözlemlenmiştir.

5.2.5. 35 Sınıftan Oluşan Veri Setinin Eğitilmesi ve Değerlendirilmesi

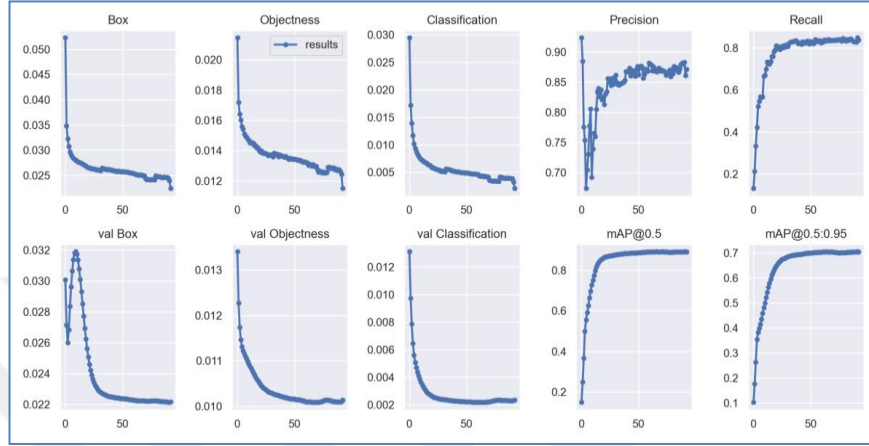
Bölüm 5.3.2’de gerçekleştirmiş olduğumuz eğitim deneyinde istenen performans alınamamış, veri setinin sınıflandırılması kısmında değişikliğe gidilmiştir. Bölüm 5.3.2’deki eğitimden elde edilen hata matrisi detaylı bir şekilde incelendiğinde, kayda değer oranda sınıflandırma hatalarının olduğu anlaşılmıştır. Sınıflandırma hatalarının çok olması modelin doğruluk oranlarının yükselmesini doğrudan etkilemiştir. Hatalı sınıflandırma yapılan sınıflar tekrar gözden geçirilip bu kategoriler veri setinde birleştirilmiş ve sınıf sayısı 35 kategoriye indirilmiştir. Model YOLOv7x modeli ile tablo 5.7’ deki parametrelerle tekrar eğitilmiştir.

Tablo 5.7: Modelin eğitiminde kullanılan parametre değerleri

Optimizasyon algoritması	Stokastik Gradyan İnişi	Hsv_h	0.015
Weights	Öğrenme Transferi Var	Hsv_s	0.7
Batch-size	4	Hsv_v	0.4
Image_size	512	Degrees	0.0
Image_weights	Kullanılmadı	Translate	0.2
Workers	8	Scale	0.9
Label_smoothing	Kullanılmadı	Shear	0.0
Freeze	0	Perspective	0.0
Lr0	0.001	Flipud	0.0
Lrf	0.1	Fliplr	0.5
Momentum	0.937	Mosaic	1.0
Weight_decay	0.0005	Mixup	0.0
Iou_t	0.2	Copy_paste	0.0
Loss_ota	0	Paste_in	0.0

Bölüm 5.3.2.’ de yapılan eğitimden farklı olarak bu eğitimde görüntü çözünürlüğü tekrar artırılmış, öğrenme oranı ise düşürülmüştür. Mixup veri artırma tekniği ile loss_ota kayıp hesaplama tekniği kullanılmamıştır. 150 döngülük bir eğitimde erken durdurma işlemi yapılarak eğitim 93’ üncü döngüden sonra sonlandırılmıştır. Loss_ota kayıp hesaplama tekniği ile paste_in veri artırma tekniğinin kullanılmaması eğitim sürecini kayda değer oranda kısaltmıştır. Eğitim yaklaşık olarak 15 gün sürmüştür. Eğitim boyunca eğitim ve doğrulama kayıpları, kesinlik, duyarlılık ve 0.5 eşikteki mAP@50 gibi ölçüm değerleri gözlemlenmiştir.

Bu ölçüm değerleri modelin ne kadar öğrendiği hakkında bilgiler verir. Eğitim ve doğrulama kayıpları başlangıçta yüksek düzeylerden başlamış, eğitim döngüsü sayısı arttıkça kayıplar yavaş yavaş azalma eğilimine girmiştir. Kesinlik, duyarlılık, mAP@0.5 ve mAP@0.5:0.95 ölçüm değerleri de döngü sayısı arttıkça daha iyi seviyelere gelmiştir. Şekil 5.7 bu eğitim deneyinden elde edilen eğitim ve doğrulama sonuçlarını göstermektedir.



Şekil 5.7 : Eğitim ve doğrulama sonuçları

Model eğitilirken öğrenme transferinden faydalanılmıştır. Öğrenme transferi sayesinde modelin doğruluğu çok kısa bir süre içerisinde yükselmiştir. Bu eğitim deneyinde mAP@0.5 değeri 0.89 seviyelerine kadar yükselmiştir. Eğitim boyunca eğitim ve doğrulama kayıpları uyum içinde kalmıştır. Bir başka deyişle model hem eğitim hem de doğrulama verileri üzerinde yaklaşık aynı performansı gösterebilmiştir. Sınıflandırma kayıpları detaylı bir şekilde analiz edildiğinde, sınıflandırma hatalarının bölüm 5.3.2.' de yapılan eğitim deneyine göre yaklaşık üç kat daha az olduğu tespit edilmiştir. Sınıflandırma hatalarının az olmasına bağlı olarak modelin doğruluk oranlarının artmasına önemli oranda katkı sağlamıştır. Ayrıca bu deney ile eğitilen model, bölüm 5.3.2.' de yapılan eğitim deneyine göre görüntü üzerinde daha fazla sınırlayıcı kutu tespit edebilmiş, kutular içerisinde daha fazla nesnellik tahmininde bulunabilmiştir. Nesnellik kayıplarının az olması, modelin görüntü üzerinden daha fazla nesne tespit edebilme yeteneğine sahip olması demektir. Buna bağlı olarak modelin doğrulama verileri üzerinde duyarlılık ölçümlerinin bir önceki deneye göre önemli oranda artmış olduğu da görülmüştür.

Bu eğitimden elde edilen kesinlik, duyarlılık ve mAP@0.50 değerleri kategorilere göre tablo 5.8' de ayrıntılı olarak gösterilmiştir.

Tablo 5.8: Kategorilere göre kesinlik, duyarlılık ve mAP değerleri

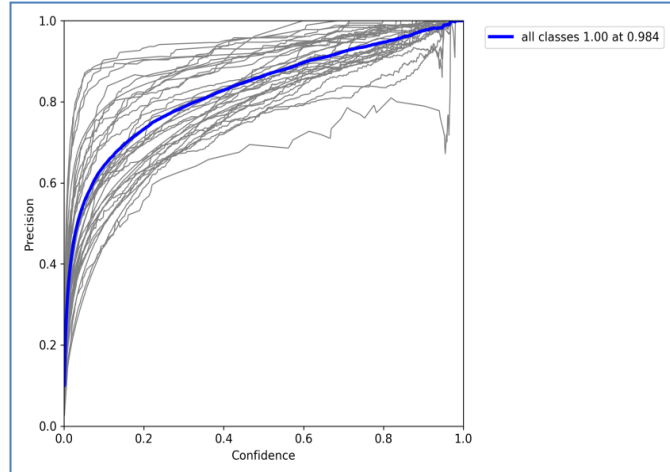
	Araç Sınıfları	YOLOv7x(512)		
		Kesinlik	Duyarlılık	mAP@0.50
	Hespi	0.871	0.838	0.891
1	Plaka Numarası	0.925	0.903	0.95
2	Bisiklet	0.9	0.808	0.897
3	Üç Tekerlekli Bisiklet-Tripörtör	0.809	0.615	0.746
4	Motosiklet	0.886	0.79	0.885
5	ATV	0.924	0.937	0.961
6	Kar Aracı	0.975	0.962	0.991
7	Jetski	0.888	1	0.995
8	Araba	0.831	0.784	0.875
9	Van	0.825	0.702	0.809
10	Limuzin	0.958	0.952	0.982
11	Otobüs – Minibüs	0.871	0.794	0.885
12	Kamyon	0.85	0.757	0.856
13	Yarı Römorklu Kamyon	0.829	0.862	0.909
14	Tankerli Kamyon	0.872	0.861	0.869
15	Çöp Kamyonu	0.922	0.929	0.936
16	Beton Mikserli Kamyon	0.925	0.846	0.913
17	İtfaie Aracı	0.787	0.862	0.845
18	Yol Süpürme Kamyonu	0.964	0.927	0.967
19	İş Makinesi	0.835	0.762	0.837
20	Çatalı Kaldıraç(Forklift)	0.948	0.925	0.96
21	Traktör	0.861	0.882	0.926
22	Segway	0.938	0.894	0.957
23	Fayton	0.807	0.682	0.759
24	Helikopter	0.961	0.962	0.983
25	Uçak	0.874	0.895	0.925
26	Gemi – Tekne	0.798	0.778	0.844
27	Tren	0.933	0.869	0.905
28	Yaya	0.872	0.745	0.863
29	Trafik Kazası	0.842	0.661	0.774
30	Scuter	0.912	0.881	0.926
31	Araba - Polis	0.776	0.858	0.879
32	Araba - Taksi	0.783	0.804	0.86
33	Van - Ambulans	0.864	0.859	0.915
34	Van - Polis	0.859	0.856	0.917
35	Van - Taksi	0.69	0.74	0.675

Tablo 5.8’de gösterilen ölçüm değerleri gözlemlendiğinde, birkaç araç dışında çoğu aracın mAP@0.50 değeri %80’ in yukarısında olduğu tespit edilmiştir. Kategorilerin duyarlılık değerleri de analiz edildiğinde, eğitilen modelin görüntü üzerindeki nesneleri tespit edebilme yeteneğinin arttığı görülmüştür.

5.2.5.1. Kesinlik Eğrisinin Değerlendirilmesi

Kesinlik, modelin doğrulama verileri üzerinde yaptığı pozitif tahminlerini ne kadar doğru yaptığını belirtir. Güven skoru arttıkça model görüntü üzerindeki nesneleri daha istikrarlı bir şekilde tahmin eder, tahmin edilen nesnelerin kesinlik değerleri de yükselir. Ancak güven skoru arttıkça modelin nesneleri görüntü üzerinde tespit edebilme yeteneğinde düşüşler yaşanabilir. Şekil 5.8’ de gösterilen grafik

eğitim sonucu elde edilen kesinlik değerlerinin güven skorları ile karşılaştırılması gösterilmektedir.



Şekil 5.8 : Kesinlik değerlerinin güven skoru ile karşılaştırılması

Şekil 5.8’ deki grafikte gösterilen gri çizgiler modeldeki her bir sınıfın güven skorlarına göre kesinlik değerlerini temsil ederken, grafikteki mavi çizgi ise güven skorlarına göre sınıfların ortalama kesinlik değerlerini temsil etmektedir. Her bir sınıf için gri çizgiler değişiklik gösterebilmektedir. Grafikte de görüldüğü gibi güven skorlarına göre bazı kategorilerin kesinlik değerleri daha yüksek iken bazılarının kesinlik değerleri ise düşüktür. Grafik detaylı şekilde analiz edildiğinde, güven skoru arttıkça modelin tahmin ettiği kesinlik değerleri de yükselmektedir. Bir başka deyişle güven skoru arttıkça modelin tahmin edemediği nesneler azalmaktadır. Grafikte güven skorunun 0.3 değerinde modelin nesneleri tahminlemesindeki minimum kesinlik değeri yaklaşık %60’ ın üzerinde olduğu, ortalama kesinlik değerinin ise %70 civarında olduğu; güven skorunun 0.5 değerinde ise minimum kesinlik değerleri bir kategori dışında %70’ in üzerinde olduğu ortalama kesinlik değerinin ise %80’ in üzerinde olduğu görülmektedir.

Modelin tahminlemesinde optimum bir güven skorunun seçilmesi gerçekten çok önemlidir. Seçilen güven skorunda modelin görüntü üzerinde hem daha fazla nesneyi tahmin edebilmesi, hem de tahminlerin daha yüksek kesinlik oranlarına ulaşması arzu edilen bir durumdur. Aslında modelin tahminlemesinde güven skorunun seçilmesi yapılan çalışmanın önemine göre değişebilmektedir. Yapılan çalışmada yüksek kesinlik oranları çok önem arz ediyor ise kesinlik-güven skoru karşılaştırmasını gösteren grafik detaylı bir şekilde analiz edilip istenen kesinlik oranlarına göre bir güven skoru seçilmelidir.

kategorilerde ise duyarlılık değerleri düşüktür. Dolayısıyla ortalama bir güven skorunda model görüntü üzerindeki bazı kategorideki sınıfları tespit etmede yüksek performans gösterirken, bazı kategorideki nesneleri tespit etmede aynı yeteneği gösterememiştir.

Grafik detaylı bir şekilde analiz edildiğinde, güven skoru arttıkça duyarlılık değerleri düşmüştür. Modelin tahminlemesinde yüksek güven skorlarının seçilmesi modelin tahmin ettiği nesnelerin doğruluk oranlarının daha kesin olmasına, yanlış pozitif tahminlerin de azalmasına katkı sağlar. Ancak yüksek güven skorları genellikle yanlış negatiflerin artmasına neden olur. Bir başka deyişle yüksek güven skorunda model görüntü üzerindeki nesneleri tespit edebilme yeteneğindeki performans azalır.

Şekil 5.9' daki duyarlılık eğrisi optimum güven skorunun belirlenmesine yardımcı olur. Güven skoru azaldıkça model görüntü üzerinden daha fazla nesne tespit edebilme yeteneği gösterir. Ancak tespit edilen nesnelerin kesinlik değerlerinde düşüşler görülebilir. Dolayısıyla güven skorunun kesinlik ve duyarlılık eğrisi ile birlikte değerlendirilmesi gerekir. Amaç, modelin görüntü üzerinden hem daha fazla nesne tespit edebilmesini hem de tespit edilen nesnelerin daha kesin değerlere ulaşabilmesini sağlamaktır.

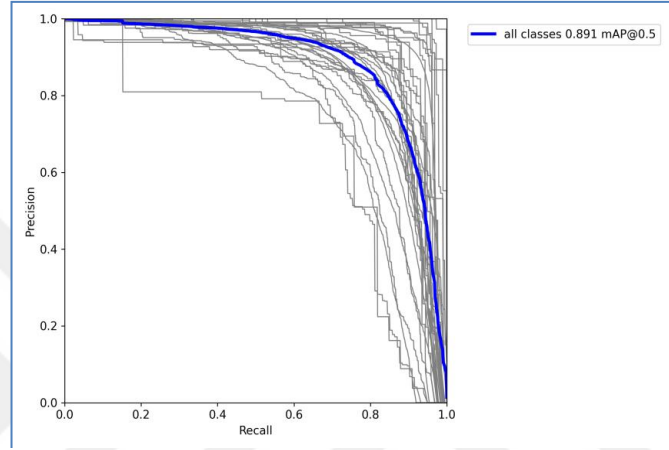
Duyarlılık ölçümü bazı kritik uygulamalarda önem arz edebilir. Özellikle tıp ve güvenlik sistemlerinde görüntü üzerindeki nesnelerin tespit edilememesi bazı problemlere neden olabilir. Bu gibi sistemlerde görüntüdeki her bir nesne ayrı bir öneme sahip olduğundan düşük güven skorları ile görüntü üzerindeki bütün nesneleri tespit edebilmek önem arz eder. Bir nesnenin tespit edilememesi kötü sonuçlara neden olabilir.

Çalışmada sadece duyarlılık eğrisi analiz edilerek optimum bir güven skorunu belirlemek yeterli değildir. Tespit edilen nesnelerin kesinlik değerlerinin de yüksek olması gerekir. Belirlenen %30 veya %35' lik bir güven skorunda modelin duyarlılık ölçümünün minimum %70' in üzerinde olduğu, ortalama duyarlılık ölçümünün de %90 civarında olduğu görülmektedir. Bir başka deyişle %30 luk bir güven skoru ile model görüntüdeki nesnelerin ortalama %90 tahminleyebilme yeteneğine sahiptir.

5.2.5.3. Kesinlik – Duyarlılık Eğrisinin Değerlendirilmesi

Modelin performansını değerlendirmek için kesinlik – duyarlılık eğrisinden de faydalanılır. Kesinlik ve duyarlılık eğrisi modelin kesinliği ve duyarlılığı arasında

bir dengeyi gösterir. Duyarlılık değerleri arttıkça tahmin edilen nesnelerin kesinlik değerleri genellikle düşer. Yüksek kesinlik ve duyarlılık değerleri modelin performansının iyi olduğunu gösterir. Bu eğri detaylı bir şekilde analiz edilerek optimum güven skoru belirlenebilir. Çalışmada yüksek duyarlılık değerleri önem arz ediyor ise daha düşük kesinlik skorları seçilebilir. Eğer yüksek kesinlik oranları daha önemli ise daha yüksek kesinlik skorları seçilebilir. Şekil 5.10' da gösterilen grafik eğitim sonucu elde edilen duyarlılık değerlerinin kesinlik değerleri ile karşılaştırılmasını göstermektedir.



Şekil 5.10 : Kesinlik ve duyarlılık değerlerinin karşılaştırılması

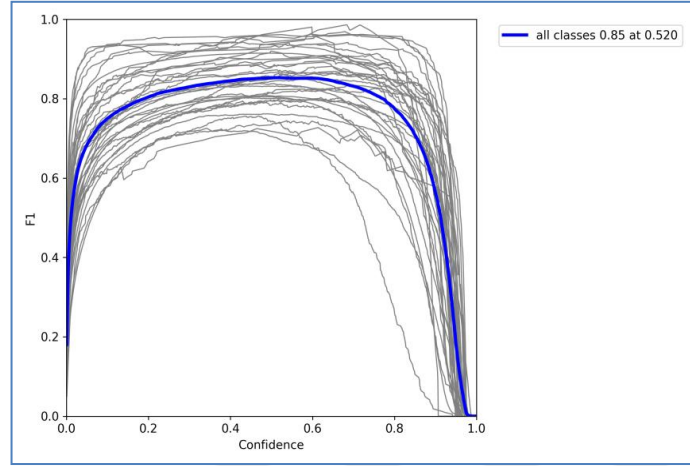
Şekil 5.10 gösterilen grafik analiz edildiğinde, ortalama duyarlılık değerleri %80 iken ortalama kesinlik değerlerinin %80'in üzerinde olduğu gözlemlenmiştir.

5.2.5.4. F1 Skor Eğrisinin Değerlendirilmesi

Kesinlik pozitif tahminlerin doğruluğunu ölçerken, duyarlılık görüntü üzerindeki bütün nesneleri tespit edebilme yeteneğini ölçer. F1 skoru ise kesinlik ve duyarlılık değerlerinin harmonik ortalamasıdır. F1 skorunun kullanılmasının nedeni yanlış negatif ile yanlış pozitif hatalarını birlikte ölçülmesine izin vermektir. Hem yanlış negatifleri hem de yanlış pozitiflerini dikkate alarak modelin dengeli bir şekilde değerlendirilebilmesine olanak verir. Sınıf dengesizliği olan durumlarda sadece kesinlik ölçümü modelin performansını ölçmede yanıltıcı olabilir. Model çoğunlukta olan sınıf örneklerini daha kolay tahmin edebildiğinden F1 skoru duyarlılık ve kesinlik değerlerini birlikte değerlendirerek modelin azınlık sınıfları üzerindeki performansını daha iyi bir şekilde değerlendirmesini sağlar.

F1 skoru yanlış pozitif ile yanlış negatif değerlerinin sayısının yüksek olmasının sorun olduğu sistemlerde önem arz eder. Hem kesinlik hem de duyarlılık ölçümleri bu sistemlerde kritik rol oynar. F1 skoru kesinlik ve duyarlılık arasındaki

dengeyi gözeterek optimum güven skorunun belirlenebilmesine yardımcı olur. Şekil 5.11’ de gösterilen grafik eğitim sonucu elde edilen F1 skorunun güven skoru ile karşılaştırılmasını göstermektedir.



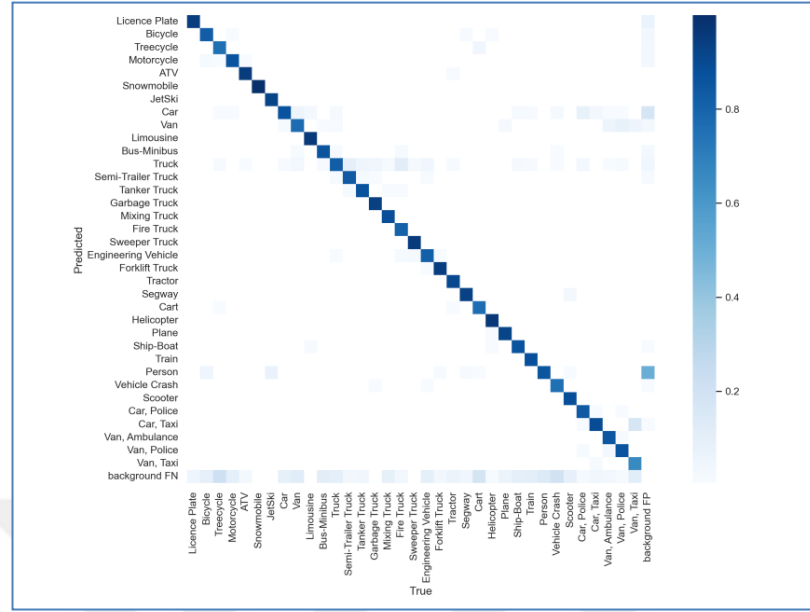
Şekil 5.11 : F1 skorunun güven skoru ile karşılaştırılması

Şekil 5.11’ deki F1 skorunun güven skoruna göre karşılaştırılma grafiği dikkatli bir şekilde analiz edildiğinde gri çizgiler her bir kategorideki F1 skorunu, mavi çizgiler ise modelin ortalama F1 skorunu temsil etmektedir. Güven skorunun 0.4 olarak seçilmesi durumunda modelin ortalama F1 skorunun %80’ in üzerinde olduğu anlaşılmıştır.

5.2.5.5. Hata Matrisinin Değerlendirilmesi

Hata matrisi modelin farklı sınıflar arasında ne kadar iyi ayırım yaptığı ve hataları ne kadar iyi tanımladığı hakkında bilgi edinilmesini sağlar. Hata matrisi, sınıflandırmaların doğru yapıp yapılmadığını, nesnelerin ne kadar doğru olarak sınıflandırıldığını, nesnelerin hangi kategorilere yanlış olarak sınıflandırıldığını gösterir. Dolayısıyla, eğitim sonrası hata matrisini detaylı inceleyerek modelin nerelerde yanlış sınıflandırma yaptığı tespit edip modelimizdeki kategorileri tekrar gözden geçirebilme olanağını elde ederiz. Hata matrisini oluşturmak için $N \times N$ şeklinde bir matris tanımlanır. Burada N sınıf sayısını temsil eder. Eğitim öncesinde modelin tahminlerini gerçek değerlerle karşılaştırabilmek için bir eşik değeri tanımlanır. Bu eşik değeri IoU değeridir. Bir başka deyişle IoU gerçek sınırlayıcı kutunun konumu ile tahmin edilen kutunun konumu arasındaki kesişimdir. Bu eşik değerinin altında kalan tahminler yanlış negatif tahmin olarak tanımlanır. Bu eşik değerinin üstünde olan ve doğru tahmin edilen nesneler doğru pozitif, eşik değerinin üstünde ve hatalı sınıflandırma yapılan tahminler ise yanlış pozitif olarak

değerlendirilir. Şekil 5.12’ de eğitim sonucu elde edilen hata matrisi gösterilmektedir.



Şekil 5.12 : Hata matrisi

Şekil 5.12’ deki hata matrisi detaylı bir şekilde analiz edildiğinde hala çok azda olsa sınıflandırma hatalarının olduğu görülmektedir. Sınıflandırma hatalarını en aza indirebilmek için YOLO mimarisinin katman sayısı ile kanal derinliği artırılarak sınıflandırma hatalarının önüne geçilebilir. Ancak karmaşık bir mimarinin tasarımı eğitim sürecini ciddi oranda arttırabilir. Karmaşık bir mimariye sahip bir modeli eğitebilmek için çok GPU’ lara ihtiyaç vardır. Kısıtlı kaynaklar ile bu tarz modelleri eğitebilmek ve optimum bir sonuca ulaşabilmek için çok uzun bir süreç gerektirecektir.

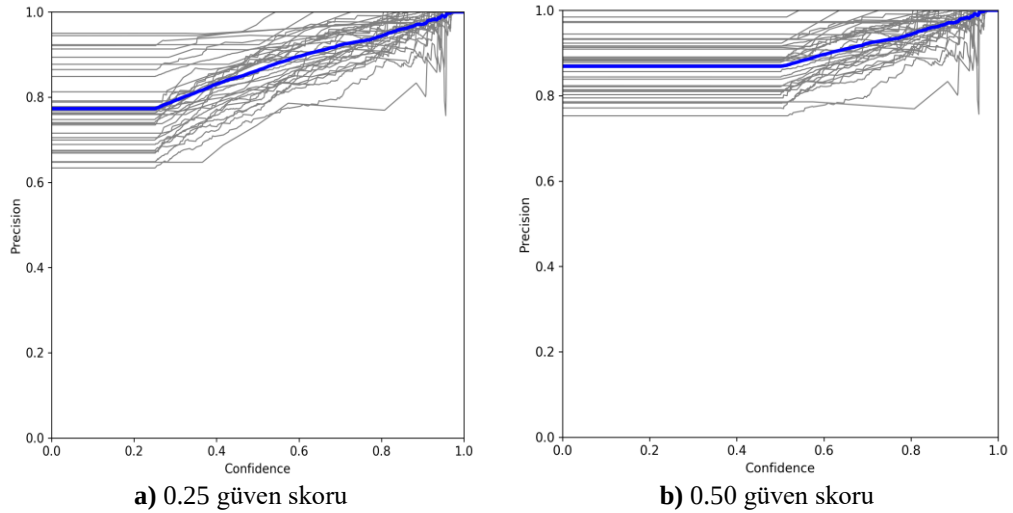
Hata matrisini detaylı bir şekilde analiz ederek eğitmiş olduğumuz modelin nerelerde yanlış sınıflandırma yaptığını tespit etmeye çalıştık. Bazı kategorilerin benzerlik göstermesi nedeni ile modelin yanlış sınıflandırma yaptığını gördük. Modelimizdeki kategorileri tekrar gözden geçirip benzerlik gösteren sınıfları birleştirip 35 kategoriye indirmeye karar verdik. Modelin yanlış sınıflandırma yapması mAP@0.50 değerini doğrudan etki ettiğinden mAP@0.50 değerinin yükselmesini engellemektedir. Ayrıca, yanlış sınıflandırma belirtilen bölgede sınırlayıcı kutuların ve nesnelerin tespit edilmesini de etkileyebilmektedir. Bu da duyarlılık ölçümünü direkt olarak etkiler.

5.3. Modelin Test Verisi Üzerinde Test Edilmesi

Geliştirilen modelin performansını değerlendirebilmek için görülmemiş veriler üzerinde test edilerek modelin genelleme yeteneğinin ne kadar iyi olduğunu öğrenmemiz önemlidir. Modelin nesneleri doğru bir şekilde tespit etmesi ve tespit edilen nesneleri doğru bir şekilde sınıflandırması gerekir. Modelin daha önce görülmemiş test verileri üzerinde test edilmesi, modelin farklı ortamlara uyum sağlayıp sağlamadığı, görüntü üzerindeki nesneleri doğru bir şekilde yerleştirip yerleştirmede ve doğru şekilde sınıflandırıp sınıflandıramadığı hakkında değerlendirme yapmamıza yardımcı olur. Modelin daha çok nerelerde hata yapma eğiliminde olduğunun tespiti modelin iyileştirilebilmesine katkı sağlar.

5.3.1. Modelin Değişik Güven Skorları İle Test Edilmesi

Modelin eğitimi tamamlandıktan sonra, test verileri üzerinde gerekli testler yapılmıştır. Model 0.25 ve 0.50 güven skorları ile test edilmiştir. Modeli farklı güven skorları ile test edilmesi, modelin farklı kriterler altında nesneleri doğru bir şekilde tespit edemediği konusunda bilgi edinmemizi sağlar. IoU eşik değeri, kesinlik ve duyarlılık arasındaki dengeyi etkiler. Şekil 5.13’ de modelin 0.25 ve 0.50 güven skorları ile test edilmesinden sonra elde edilen kesinlik eğrilerinin karşılaştırılması gösterilmektedir.

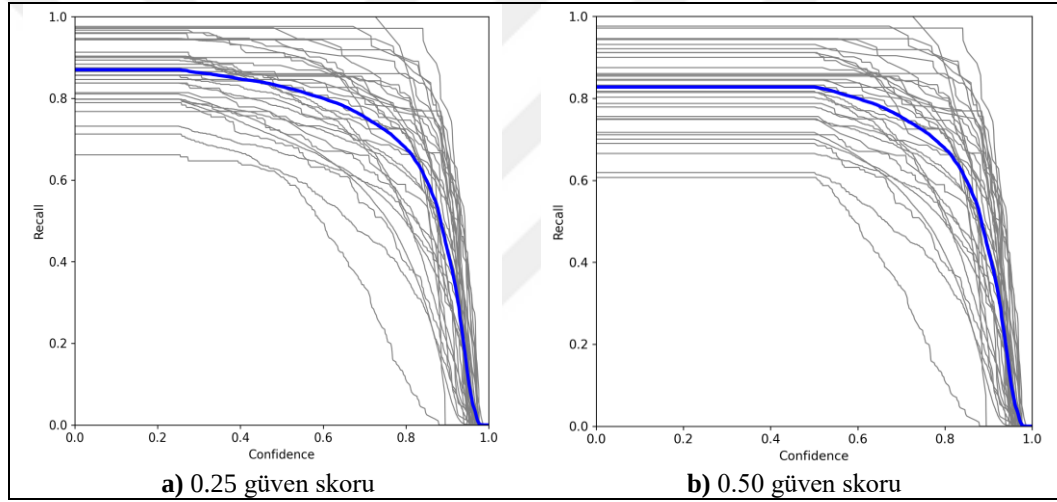


Şekil 5.13 : 0.25 ve 0.50 güven skoru ile elde edilen kesinlik eğrilerinin güven skorları ile karşılaştırılması

Şekil 5.13a’ da modelin 0.25 güven skoru ile şekil 5.13b de ise modelin 0.50 güven skoru ile test edilmesi sonucu elde edilen kesinlik ve güven skoru karşılaştırılmasını göstermektedir. Şekil 5.13a’ daki grafikteki ortalama kesinlik oranlarının %80 değerine yaklaştığı, şekil 5.13b’ deki grafikteki ortalama kesinlik

oranlarının ise %90 değerine yaklaştığı gözlemlenmiştir. Sonuç olarak modelin %50 güven skoru ile test edilmesi modeli daha istikrarlı hale getirmiş, yapılan tahminlerin doğruluk oranları daha sağlam ve daha doğru olduğu tespit edilmiştir. Yapılan çalışmanın önemine göre modelin güven skorları arttırıldığında modelin tahminlerinin doğruluk oranları da artacaktır.

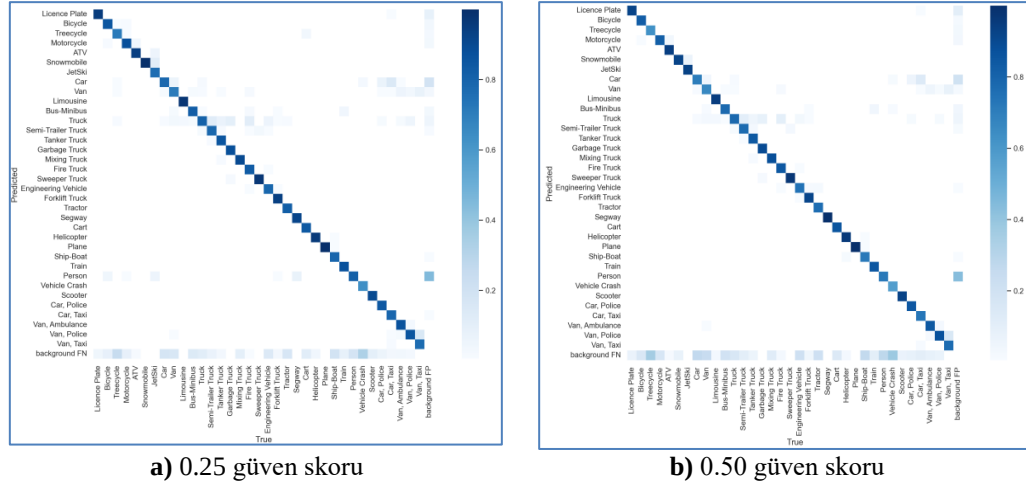
Model test verileri üzerinde farklı eşik değerleri ile test edildikten sonra elde edilen duyarlılık eğrilerinin de değerlendirilmesi önem arz etmektedir. Modelin görüntü üzerinden daha fazla nesneyi tespit edebilmesi çalışmanın türüne göre kritik önem taşıyabilir. Bu gibi uygulamalarda görüntü üzerindeki nesnelerin eksik tespit edilmesi bazı problemlere neden olabilir. Şekil 5.14' de modelin 0.25 ve 0.50 güven skorları ile test edilmesinden sonra elde edilen duyarlılık eğrilerinin karşılaştırılması gösterilmektedir.



Şekil 5.14 : 0.25 ve 0.50 güven skoru ile elde edilen duyarlılık eğrilerinin güven skorları ile karşılaştırılması

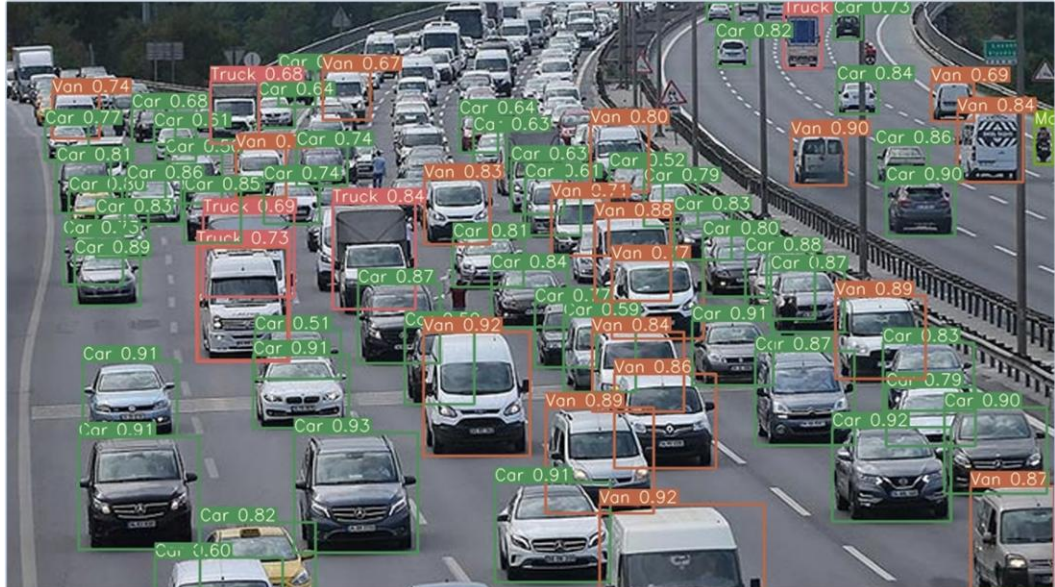
Şekil 5.14a' da modelin 0.25 güven skoru ile şekil 5.14b de ise modelin 0.50 güven skoru ile test edilmesi sonucu elde edilen duyarlılık ve güven skoru karşılaştırılmasını göstermektedir. Modelin 0.25 güven skoru ile daha çok nesne tahminlemesinde bulunabildiği gözlemlenmektedir. Güven skoru azaldıkça model görüntü üzerinden daha çok nesne tespit edebilmektedir.

Modelin test verileri üzerindeki performansını değerlendirebilmenin bir diğer yolu hata matrisinin analiz edilmesidir. Hata matrisi analiz edilerek modelin tahminlerini ne kadar doğru yapabildiğini değerlendirebilmemizi sağlar. Şekil 5.15' de modelin 0.25 ve 0.50 güven skorları ile test edilmesinden sonra elde edilen hata matrislerinin karşılaştırılması gösterilmektedir.



Şekil 5.15 : 0.25 ve 0.50 güven skoru ile elde edilen kesinlik ve duyarlılık karşılaştırma eğrileri

Şekil 5.15’ de gösterilen hata matrisleri detaylı bir şekilde analiz edildiğinde modelin 0.25 güven skoru ile daha çok sınıflandırma hataları yaptığı, 0.50 güven skoru ile modelin görüntü üzerindeki nesneleri daha doğru tahmin ettiği anlaşılmıştır. Güven skorları azaldıkça model daha fazla sınıflandırma hataları yapabilir, güven skoru azaldıkça sınıflandırma hataları daha az görülebilir. Yüksek doğruluk değerlerinin önemli olduğu sistemlerde güven skorunun artırılması sınıflandırma hatalarının azalmasına katkı sağlayabilir. Şekil 5.16, modelin test görüntüsü üzerindeki nesneleri tespit etmesini ve sınıflandırmasını grafiksel olarak göstermektedir.



Şekil 5.16 : Trafikteki araçların tespit edilip sınıflandırılması

Şekil 5.16’ da modelin görüntü üzerindeki nesneleri yüksek güven skorları ile tespit ettiği gözlemlenmiştir. Yüksek güven skorları ile araçların tespit edilmesi

sınıflandırma hatalarının azalmasına katkı sağlar. Tablo 5.9’ da modelin test görüntüleri üzerinde test edilmesinden sonra elde edilen bazı sınıflardaki araçların tespit edilmesini ve sınıflandırmasını göstermektedir.

Tablo 5.9 : Farklı sınıfta araçların tespit edilip etiketlenmesi

 Semi-Trailer Truck 0.96	 Cart 0.94	 Car 0.93
a) Yarı römorklu kamyon	b) Fayton	c) Araba
 Fire Truck 0.87	 Bicycle 0.49	 Motorcycle 0.88
d) İtfaiye aracı	e) Bisiklet	f) Motorsiklet
 Truck 0.92	 Mixing Truck 0.92	 Van, Ambulance 0.90
g) Tankerli kamyon	h) Mikserli kamyon	i) Ambulans
 Truck 0.92	 Car, Taxi 0.95	 Licence Plate 0.82
j) Kamyon	k) Taksi	l) Plaka numarası

Tablo 5.9’ da sınıflandırılan çoğu aracın güven skorlarının yüksek olduğu, Tablo 5.9e’ deki bisikletin yüzde 49 güven skoru ile doğru bir şekilde tespit edildiği anlaşılmıştır.

5.3.2. Test Verilerinden Elde Edilen Sonuçların Değerlendirilmesi

Eğitilen model test verileri üzerinde 0.25 ve 0.50 güven skorları ile test edilmiştir. Elde edilen sonuçlar analiz edildiğinde, düşük güven skoru ile test verileri üzerinde daha çok nesnenin tespit edildiği ancak yapılan nesne tespitlerinin kesinlik değerlerinin yüksek güven skoru ile test edilen verilerden elde edilen sonuçlara göre daha düşük olduğu görülmüştür. Güven skoru arttıkça tespit edilen nesnelerin kesinlik değerlerinin arttığı, sınıflandırma hatalarının ise azaldığı anlaşılmıştır. Düşük güven skoru ile modelin test verileri üzerinde test edilmesi halinde modelin hata yapma olasılığı da artmaktadır. Model düşük güven skorları ile görüntü üzerinde çok sayıda nesnenin tespit edilebilmesine olanak sağlarken, aynı bölgede birden fazla kategoride hatalı sınıflandırma tahminlemesi gösterebilir, iki aracı bir araçmış gibi hatalı bir şekilde tespit edilebilmesine sebebiyet verebilir.

6. TARTIŞMA

Çalışmamızdaki amaç hareket halinde olan araçları hem tespit etmek hem de bu araçları birçok sınıfa ayırmaktır. Karayollarındaki tüm araçları farklı görünüş ve açılarla tespit edip bu araçları kategorilere ayırmak gerçekten çok zor bir çalışmadır. Araçları çok sayıda sınıfa ayırmak için çok büyük bir veri setine ihtiyaç duyulur. Modelin iyi bir şekilde eğitilebilmesi için her türlü hava şartlarında(karanlık, aydınlık, yağmurlu, karlı vb) her bir kategori için araçların farklı görünüşlerini içeren görüntülerin veri seti içinde yer alması gerekir. Ayrıca her bir kategorideki görüntü verilerinin dengeli olması modelin sağlamlığı açısından oldukça önemlidir.

Model geliştirilirken dikkat edilmesi gereken konulardan biri modelin aşırı öğrenmesidir. Modelin aşırı öğrenmesini önlemek için çalışmada çok büyük bir veri setine ihtiyaç duyulur. Veri setinin küçük olması halinde model eğitim verileri üzerinde aşırı uyum gösterip doğrulama verileri üzerinde aynı performansı gösteremeyebilir. Eğitimde kullanılan veri setinin hem çok büyük olması hem de bu verilerinin oldukça çeşitli olması gerekir. Eğitim verilerinin yüksek oranda çeşit içermesi modelin bu verileri öğrenebilmesine ve modelin genelleme yapabilme yeteneğinin gelişmesine katkı sağlayacaktır. Ancak, çok büyük veri setini eğitmek başka bir problemi beraberinde getirecektir. Modelin performansını yükseltmek için

çok sayıda model döngüsüne, eğitime başlamadan önce öğrenme oranının optimum değerinin seçilmesine ihtiyaç duyulur. Öğrenme oranının çok düşük seçilmesi modelin en uygun performansa ulaşmasını sağlayacak ancak çok sayıda eğitim döngüsüne ihtiyaç duyulacağından modelin eğitim süresini de uzatacaktır. Dolayısıyla çalışmamızda kaynaklar kısıtlı olduğu için öğrenme oranını makul bir seviyede tutarak modelin en iyi performansa yakalamasını sağladık. Çok büyük veri setini eğitmek için çok yüksek hesaplama gücüne sahip GPU' lara ihtiyaç vardır. Ayrıca, modelin aşırı öğrenmesini engel olmanın bir diğer yolu erken durdurma işlemidir. Modelin eğitimi sırasında, sonuçlara ara ara bakılmalı, eğitim ve doğrulama kayıpları detaylı bir şekilde analiz edilmeli, doğrulama kayıplarında yer alan sınırlayıcı kutu, nesne ve sınıf kayıplarında belirli bir noktadan sonra düzelmeyip artış devam ediyor ve modelin doğruluk oranı da aşırı bir şekilde düşüyor ise modelin belirli bir döngüden sonra erken durdurma işlemi yapılması gerekecektir. Aksi takdirde, modelden istenen performans alınamayabilir.

Çalışmamızda yer alan bir diğer konu öğrenme transferidir. Hem eğitim süresini kısaltmak hem de performans açısından en iyi sonuca ulaşmak için çalışmada öğrenme transferinden yararlanılmıştır. Seçmiş olduğumuz YOLOv7 algoritması bize önceden eğitilmiş modeller sunmaktadır. Önceden eğitilmiş bu modeller çok büyük veri seti ile eğitildiğinden, hem modelin en iyi performansa ulaşmasını hem de eğitim süresi açısından bize zamandan tasarruf etmemizi sağlayacaktır. Ayrıca, öğrenme transferi olmadan kısıtlı veri seti ile modeli eğitmek, farklı test veri kümelerinde modelin iyi bir performans gösterememesine yol açabilecektir.

Çalışmamızda, karayollarında hareket halinde olan araçları birçok kategoriye ayırmaya çalıştığımızı söylemiştik. Ancak araçları birçok kategoriye ayırmak başka bir problemin ortaya çıkmasına da neden olmuştur. Çalışma süresi içerisinde modelimizde önce 49 kategoriden oluşmakta idi. Ne kadar çok deneme yapılmış ise de, modelde istenilen performans yakalanamamıştır. Eğitim sonrası oluşturulan hata matrisi incelendiğinde bazı kategorideki araçların yüksek sınıflandırma hataları yaptığı gözlemlenmiştir. Sınıflandırma hatalarını anlamak için YOLO' nun çalışma prensibini iyi anlamamız gerekir. YOLO algoritması, iki aşamalı algoritmaların aksine tek aşamada görüntüyü ızgara hücrelerine böler ve bu ızgara hücrelerinden sınırlayıcı kutular üretir. Dolayısıyla kategorilerin birbirine çok benzerlik göstermesi sınırlayıcı kutular içerisinde nesnelerin tespit edilememesine ve yanlış sınıflandırma

yapılmasından dolayı modelin mAP değerlerinin istenilen seviyelere ulaşamamasına neden olacaktır. Eğitim sonrası oluşturulan hata matrisi detaylı bir şekilde analiz edilerek çok sınıflandırma hatası yapan kategoriler birleştirilmiş, model 49 kategoriden 35 kategoriye indirilmiştir. Yeni oluşturulan 35 kategorili veri seti ile tekrar eğitim yapılmış, eğitim sonrası oluşturulan sonuç ve grafikler incelendiğinde modelin görüntülerde daha çok nesneyi tespit ettiği, modelin mAP değerlerinde kayda değer artış olduğu gözlemlenmiştir.

Model geliştirilirken veri seti ile algoritmanın mimari yapısının karmaşıklığının dengeli olması gerekir. Veri setinin büyüklüğüne ve çeşitliliğine göre modelin karmaşıklık yapısında değişiklikler yapılabilir. Modeldeki evrimsel katmanların az olması ve modelin daha az derinliğe yani daha az ağırlık ve sapma parametresine sahip olması modelin eğitim verilerini öğrenmesinde yetersiz kalabilir. Çalışma YOLOv7-tiny modeli ile eğitime çalışılmış ancak yetersiz öğrenme problemi ile karşılaşmıştır. Çalışmada oldukça büyük bir veri seti kullanılmıştır. Modelin performansını artırabilmek ve genelleme yapabilme yeteneğini iyileştirebilmek için farklı metotlar denenerek modelin doğruluğu %89' lara kadar çıkarılabilmektedir. Daha fazla doğruluk elde edebilmek için modelin karmaşıklık yapısı biraz daha artırılabilir. Modelin mimari yapısındaki evrimsel katmanlar ile kanal derinliğinin artırılması ve modelin bu karmaşıklık yapısının veri seti ile uyumunun iyi bir şekilde ayarlanması modelin daha fazla doğruluk elde edilebilmesine katkı sağlayabilir. Ancak modelin mimari yapısında yapılan değişiklik öğrenme transferinden yararlanılmasını engelleyecektir. Dolayısıyla daha karmaşık bir tasarıma sahip algoritmadan faydalanılarak modeli kısıtlı kaynaklar ile eğitebilmek birtakım sorunların oluşmasına neden olacaktır. GPU hafızasının yetersiz olması karmaşık bir modeli eğitilebilmesinde sorunlara neden olacak, eğitim süresini de ciddi oranda uzatacak ve modelin eğitiminin değerlendirilebilmesini olumsuz yönde etki edecektir. Modelin mimari yapısının çok karmaşık hale gelmesi yani modelin çok fazla evrimsel katmana ve çok fazla kanal derinliğine sahip olması, modelin eğitim verileri üzerinde aşırı derece uyum göstermesine, eğitim verilerini daha iyi öğrenmesine neden olacak, doğrulama verileri üzerinde genelleme yapabilme yeteneğini kaybetmesine neden olabilecektir. Dolayısıyla eğitilecek olan veri seti ile modelin karmaşıklık yapısının dengeli olması gerekir.

Modelin eğitimi gerçekleştirilirken dikkat edilmesi gereken bir diğer konu ise veri kümesinde yer alan görüntülerin yüksek çözünürlüklü olmasıdır. YOLO

algoritması yukarıda da bahsedildiği gibi görüntüyü ızgara hücrelerine böler. Algoritma, bu ızgara hücrelerini kullanarak çok sayıda sınırlayıcı kutu(bounding box) üretir. Dolayısıyla, görüntünün yüksek çözünürlüklü olması modelin çok sayıda sınırlayıcı kutu üretebilmesine olanak sağlayacak, görüntüde yer alan küçük nesnelerin de tespit edilebilmesini kolaylaştıracaktır. Düşük çözünürlüğe sahip görüntülerin aksine yüksek çözünürlük içeren görüntülerde küçük nesneler daha fazla ayrıntıya sahip olacağından, küçük nesnelerin tespiti ve nesnelerin daha fazla doğruluk oranına sahip olmasına olanak sağlayacak, hatalı sınıflandırma yapılmasına da engel olacaktır.

7. SONUÇ

Çalışmamızdaki amaç karayollarında hareket halinde olan araçları görüntü üzerinden tespit edip tespit edilen bu araçları birçok kategoriye ayırmaya çalışmaktır. Hareket halinde olan araçları kategorilere ayırmak birçok avantaj sağlayabilir. Örneğin karayollarında seyretmekte olan tehlikeli madde taşıyan tankerli araçların görüntü üzerinden tespit edilip yasaklı olan tünellerden geçişi önlenebilir. Karayollarını gözetleyen kameralardan elde edilen görüntüler bu model ile işlenip trafik akışını yavaşlatan ağır yük taşıyan yarı römorklu kamyonların görüntü üzerinden tespitinin yapılarak belirlenen saatlerde şehre girişi önlenebilir ya da istenilen karayoluna yönlendirilmesi yapılabilir. Geliştirmiş olduğumuz model sayesinde hem araçların plakaları tespit edilip hem de araçların sınıflandırılarak karayollarında seyreden araçlar veritabanında kayıt altında tutulması da sağlanabilir.

Çalışmamızda modelin performansını iyileştirmek ve modelde en iyi sonucu yakalayabilmek için çok sayıda yöntem ve analiz yapılarak çok sayıda eğitim denemesi yapılmak zorunda kalınmıştır. Eğitim sonrası oluşturulmuş olan hata matrisi, kesinlik ve duyarlılık gibi grafikler detaylı bir şekilde analiz edilerek eğitim sırasında oluşmuş olan ve modelin mAP ve recall değerlerinin yükselmesini engel olan hatalar tespit edilip modelin performansını yeteri kadar yükseltebilmek için farklı yöntemler kullanılmak zorunda kalınmıştır. Modelin eğitimine başlamadan önce optimum öğrenme oranı tespit edilip kısıtlı kaynaklarla uygun olan zaman dilimi içerisinde modelin en iyi performansa ulaşması sağlanmıştır.

Çalışmaya başlamadan önce birçok literatür taraması yapılmış, nesne tespiti algoritmalarındaki süreç detaylı bir şekilde incelenerek uygun algoritma seçimi yapılmaya çalışılmıştır. Algoritma olarak YOLOv7 yapmak istediğimiz çalışmaya uygun olacağı anlaşılmıştır. YOLOv7, sahip olduğu veri artırma teknikleri ile modelin performansının artmasında önemli oranda katkı sağlamıştır. Ayrıca, algoritma içerisinde yer alan farklı ölçekteki modelleri ile kısıtlı kaynaklarda gerçek zamanlı olarak geliştirilen modelin çalışabilmesine olanak vermiştir.

Yapmış olduğumuz çalışmada en iyi sonuca ulaşabilmek için YOLO algoritmasının çalışma prensibini, algoritma içerisinde kullanılan yöntemlerin iyi bir şekilde bilinmesi gerekir. Algoritmanın çalışma prensipleri detaylı bir şekilde analiz edilmeden çalışmanın yapılması sırasında istenmeyen durumlarla karşılaşıldığında ne çeşit önlemler ve tedbirler alınabileceği kestirilemeyebilir. Dolayısıyla, veri seti hazırlamadan modelin eğitime kadar olan sürecin çok iyi bir şekilde analiz edilmesi, karşılaşılan problemlerin neden kaynaklandığını bulabilmek ve oluşan problemlere çözüm getirebilmek için nesne tespiti algoritmaların nasıl çalıştığı konusunda çok iyi bir bilgi birikimine sahip olmak gerekmektedir. Çünkü görüntü üzerindeki nesnelerin tespiti, tespit edilen nesnelerin doğru bir şekilde sınıflandırılması, tespit edilen nesnelerin yüksek doğruluk oranlarına ulaştırılması ve görüntü üzerindeki tüm nesnelerin doğru bir şekilde tespit edilmeye çalışılması gerçekten çok zorlu bir çalışmadır. Görüntü üzerindeki nesnelerin her ortamda tespit edilmeye çalışılması hala üzerinde çalışılan konular içerisinde yer almaktadır.

8. KAYNAKLAR

1. Alzubaidi, L., et al., *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*. J Big Data, 2021. **8**(1): p. 53.
2. Ghosh, A., et al., *Fundamental concepts of convolutional neural network*. Recent trends and advances in artificial intelligence and Internet of Things, 2020: p. 519-567.
3. Hubel, D.H. and T.N. Wiesel, *Receptive fields and functional architecture of monkey striate cortex*. The Journal of physiology, 1968. **195**(1): p. 215-243.
4. Fukushima, K., *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*. Biological cybernetics, 1980. **36**(4): p. 193-202.
5. LeCun, Y., et al., *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 1998. **86**(11): p. 2278-2324.
6. Krizhevsky, A., I. Sutskever, and G.E. Hinton, *Imagenet classification with deep convolutional neural networks*. Advances in neural information processing systems, 2012. **25**.
7. Amjoud, A.B. and M. Amrouch, *Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review*. IEEE Access, 2023.
8. Tripathi, A., et al. *Object detection using YOLO: A survey*. in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*. 2022. IEEE.
9. Li, S., et al., *Yolo-firi: Improved yolov5 for infrared image object detection*. IEEE access, 2021. **9**: p. 141861-141875.
10. Uijlings, J.R., et al., *Selective search for object recognition*. International journal of computer vision, 2013. **104**: p. 154-171.
11. Dalal, N. and B. Triggs. *Histograms of oriented gradients for human detection*. in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. 2005. Ieee.
12. Lowe, D.G., *Distinctive image features from scale-invariant keypoints*. International journal of computer vision, 2004. **60**: p. 91-110.
13. Bay, H., T. Tuytelaars, and L. Van Gool. *Surf: Speeded up robust features*. in *Computer Vision—ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. 2006. Springer.
14. Cortes, C. and V. Vapnik, *Support-vector networks*. Machine learning, 1995. **20**: p. 273-297.
15. Girshick, R., et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.
16. Girshick, R. *Fast r-cnn*. in *Proceedings of the IEEE international conference on computer vision*. 2015.
17. Ren, S., et al., *Faster r-cnn: Towards real-time object detection with region proposal networks*. Advances in neural information processing systems, 2015. **28**.
18. Redmon, J., et al. *You only look once: Unified, real-time object detection*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
19. Purwono P., M.a.A., Rahmaniar W. , *Understanding of convolutional neural network*. International Journal of Robotics and Control Systems 2.4 (2022), 2022. **739-748**.

20. Sun, Y., et al., *Evolving deep convolutional neural networks for image classification*. IEEE Transactions on Evolutionary Computation, 2019. **24**(2): p. 394-407.
21. KAYALI, N.Z. and S.İ. OMURCA, *Konvolüsyonel Sinir Ağları (CNN) ile Çin Sayı Örüntülerinin Sınıflandırması*. Computer Science, 2021(Special): p. 184-191.
22. Goodfellow I, B.Y., Courville A, Bengio Y., *Deep Learning*. Vol. vol. 1. 2016, Cambridge: MIT press.
23. Tüfekçi, M. and F. Karpat. *Derin Öğrenme Mimarilerinden Konvolüsyonel Sinir Ağları (CNN) Üzerinde Görüntü İşleme-Sınıflandırma Kabiliyetininin Arttırılmasına Yönelik Yapılan Çalışmaların İncelenmesi*. in *International Conference on Human-Computer Interaction, Optimization and Robotic Applications*. 2019.
24. LeCun, Y., et al., *Backpropagation applied to handwritten zip code recognition*. Neural computation, 1989. **1**(4): p. 541-551.
25. Tareen, S.A.K. and F.K. Tareen, *Convolutional neural networks for beginners*. 2018.
26. Ramos-Michel, A., et al., *Image Classification with Convolutional Neural Networks*, in *Metaheuristics in Machine Learning: Theory and Applications*. 2021, Springer. p. 445-473.
27. Ahmed, W.S. *The impact of filter size and number of filters on classification accuracy in CNN*. in *2020 International conference on computer science and software engineering (CSASE)*. 2020. IEEE.
28. Nguyen, A., et al., *Explaining Convolutional Neural Networks by Tagging Filters*. arXiv preprint arXiv:2109.09389, 2021.
29. He, K., et al. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. in *Proceedings of the IEEE international conference on computer vision*. 2015.
30. Glorot, X. and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks*. in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010. JMLR Workshop and Conference Proceedings.
31. Sahu, M. and R. Dash. *A survey on deep learning: convolution neural network (CNN)*. in *Intelligent and Cloud Computing: Proceedings of ICICC 2019, Volume 2*. 2021. Springer.
32. Kumar, A., S. Sarkar, and C. Pradhan, *Malaria disease detection using cnn technique with sgd, rmsprop and adam optimizers*. Deep learning techniques for biomedical and health informatics, 2020: p. 211-230.
33. Sultan, H.H., N.M. Salem, and W. Al-Atabany, *Multi-classification of brain tumor images using deep neural network*. IEEE access, 2019. **7**: p. 69215-69225.
34. Albawi, S., T.A. Mohammed, and S. Al-Zawi. *Understanding of a convolutional neural network*. in *2017 international conference on engineering and technology (ICET)*. 2017. Ieee.
35. Nielsen, M.A., *Neural networks and deep learning*. Vol. 25. 2015: Determination press San Francisco, CA, USA.
36. Werbos. *Backpropagation: Past and future*. in *IEEE 1988 International Conference on Neural Networks*. 1988. IEEE.
37. Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning Internal Representations by Error Propagation*, *Parallel Distributed Processing*,

- Explorations in the Microstructure of Cognition*, ed. DE Rumelhart and J. McClelland. Vol. 1. 1986. Biometrika, 1986. **71**: p. 599-607.
38. Ruder, S., *An overview of gradient descent optimization algorithms*. arXiv preprint arXiv:1609.04747, 2016.
 39. Bottou, L. *Large-scale machine learning with stochastic gradient descent*. in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics Paris France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. 2010. Springer.
 40. Hinton, G., N. Srivastava, and K. Swersky, *Lecture 6a overview of mini-batch gradient descent*. Coursera Lecture slides <https://class.coursera.org/neuralnets-2012-001/lecture>, [Online, 2012.
 41. Hinton, G., N. Srivastava, and K. Swersky, *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent*. Cited on, 2012. **14**(8): p. 2.
 42. Khirirat, S., H.R. Feyzmahdavian, and M. Johansson. *Mini-batch gradient descent: Faster convergence under data sparsity*. in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017. IEEE.
 43. Qian, N., *On the momentum term in gradient descent learning algorithms*. *Neural networks*, 1999. **12**(1): p. 145-151.
 44. Zhang, Z. *Improved adam optimizer for deep neural networks*. in *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*. 2018. Ieee.
 45. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980, 2014.
 46. Duchi, J., E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*. *Journal of machine learning research*, 2011. **12**(7).
 47. Zeiler, M.D., *Adadelta: an adaptive learning rate method*. arXiv preprint arXiv:1212.5701, 2012.
 48. Dozat, T., *Incorporating nesterov momentum into adam*. 2016.
 49. Gustineli, M., *A survey on recently proposed activation functions for Deep Learning*. arXiv preprint arXiv:2204.02921, 2022.
 50. Feng, J. and S. Lu. *Performance analysis of various activation functions in artificial neural networks*. in *Journal of physics: conference series*. 2019. IOP Publishing.
 51. Szandała, T., *Review and comparison of commonly used activation functions for deep neural networks*. *Bio-inspired neurocomputing*, 2021: p. 203-224.
 52. Hara, K., H. Kataoka, and Y. Satoh. *Learning spatio-temporal features with 3d residual networks for action recognition*. in *Proceedings of the IEEE international conference on computer vision workshops*. 2017.
 53. Alkhoully, A.A., A. Mohammed, and H.A. Hefny, *Improving the performance of deep neural networks using two proposed activation functions*. *IEEE Access*, 2021. **9**: p. 82249-82271.
 54. Sharma, O. *A new activation function for deep neural network*. in *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. 2019. IEEE.
 55. Cox, S.J. and J.S. Bridle. *Unsupervised speaker adaptation by probabilistic spectrum fitting*. in *International Conference on Acoustics, Speech, and Signal Processing*. 1989. IEEE.

56. Kouretas, I. and V. Paliouras. *Simplified hardware implementation of the softmax activation function*. in *2019 8th international conference on modern circuits and systems technologies (MOCASST)*. 2019. IEEE.
57. Nair, V. and G.E. Hinton. *Rectified linear units improve restricted boltzmann machines*. in *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010.
58. Glorot, X., A. Bordes, and Y. Bengio. *Deep sparse rectifier neural networks*. in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011. JMLR Workshop and Conference Proceedings.
59. Maas, A.L., A.Y. Hannun, and A.Y. Ng. *Rectifier nonlinearities improve neural network acoustic models*. in *Proc. icml*. 2013. Atlanta, GA.
60. Clevert, D.-A., T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus)*. arXiv preprint arXiv:1511.07289, 2015.
61. Klambauer, G., et al., *Self-normalizing neural networks*. Advances in neural information processing systems, 2017. **30**.
62. Carlile, B., et al., *Improving deep learning by inverse square root linear units (ISRLUs)*. arXiv preprint arXiv:1710.09967, 2017.
63. Nicolae, A., *PLU: The piecewise linear unit activation function*. arXiv preprint arXiv:1809.09534, 2018.
64. Zheng, H., et al. *Improving deep neural networks using softplus units*. in *2015 International joint conference on neural networks (IJCNN)*. 2015. IEEE.
65. Hendrycks, D. and K. Gimpel, *Gaussian error linear units (gelus)*. arXiv preprint arXiv:1606.08415, 2016.
66. Elfving, S., E. Uchibe, and K. Doya, *Sigmoid-weighted linear units for neural network function approximation in reinforcement learning*. Neural networks, 2018. **107**: p. 3-11.
67. Ramachandran, P., B. Zoph, and Q.V. Le, *Searching for activation functions*. arXiv preprint arXiv:1710.05941, 2017.
68. Tanaka, M., *Weighted sigmoid gate unit for an activation function of deep neural network*. Pattern Recognition Letters, 2020. **135**: p. 354-359.
69. Terven, J., et al., *Loss Functions and Metrics in Deep Learning*. arXiv preprint arXiv:2307.02694, 2023.
70. Goodman, R., J.W. Miller, and P. Smyth, *Objective functions for neural network classifier design*. 1991.
71. Jadon, S. *A survey of loss functions for semantic segmentation*. in *2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB)*. 2020. IEEE.
72. Nadaraya, E.A., *On estimating regression*. Theory of Probability & Its Applications, 1964. **9**(1): p. 141-142.
73. Rusiecki, A., *Trimmed categorical cross-entropy for deep learning with label noise*. Electronics Letters, 2019. **55**(6): p. 319-320.
74. Müller, R., S. Kornblith, and G.E. Hinton, *When does label smoothing help?* Advances in neural information processing systems, 2019. **32**.
75. Szegedy, C., et al. *Rethinking the inception architecture for computer vision*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
76. Lin, T.-Y., et al. *Focal loss for dense object detection*. in *Proceedings of the IEEE international conference on computer vision*. 2017.

77. Hossain, M.S., J.M. Betts, and A.P. Paplinski, *Dual Focal Loss to address class imbalance in semantic segmentation*. Neurocomputing, 2021. **462**: p. 69-87.
78. He, T., et al. *Instance-aware embedding for point cloud instance segmentation*. in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX* 16. 2020. Springer.
79. Luo, Z., et al. *Rethinking the heatmap regression for bottom-up human pose estimation*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.
80. Wang, S., et al. *Training deep neural networks on imbalanced data sets*. in *2016 international joint conference on neural networks (IJCNN)*. 2016. IEEE.
81. Boulila, W., et al., *Weight Initialization Techniques for Deep Learning Algorithms in Remote Sensing: Recent Trends and Future Perspectives*. CoRR abs/2102.07004 (2021). arXiv preprint arXiv:2102.07004, 2021.
82. Peng, A.Y., et al. *Using supervised pretraining to improve generalization of neural networks on binary classification problems*. in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I* 18. 2019. Springer.
83. Koturwar, S. and S. Merchant, *Weight initialization of deep neural networks (DNNs) using data statistics*. arXiv preprint arXiv:1710.10570, 2017.
84. Li, H., M. Krček, and G. Perin. *A comparison of weight initializers in deep learning-based side-channel analysis*. in *Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings* 18. 2020. Springer.
85. Santos, C.F.G.D. and J.P. Papa, *Avoiding overfitting: A survey on regularization methods for convolutional neural networks*. ACM Computing Surveys (CSUR), 2022. **54**(10s): p. 1-25.
86. Nitish, S., *Dropout: a simple way to prevent neural networks from overfitting*. J. Mach. Learn. Res., 2014. **15**: p. 1.
87. Farhadi, Z., H. Bevrani, and M.-R. Feizi-Derakhshi. *Combining Regularization and Dropout Techniques for Deep Convolutional Neural Network*. in *2022 Global Energy Conference (GEC)*. 2022. IEEE.
88. do Santos, C.F.G., et al. *MaxDropout: deep neural network regularization based on maximum output values*. in *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021. IEEE.
89. Wan, L., et al. *Regularization of neural networks using dropconnect*. in *International conference on machine learning*. 2013. PMLR.
90. Ghiasi, G., T.-Y. Lin, and Q.V. Le, *Dropblock: A regularization method for convolutional networks*. Advances in neural information processing systems, 2018. **31**.
91. Zhu, H. and X. Zhao. *Targetdrop: A targeted regularization method for convolutional neural networks*. in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022. IEEE.

92. Pham, H. and Q. Le. *Autodropout: Learning dropout patterns to regularize deep networks*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021.
93. Ioffe, S. and C. Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. in *International conference on machine learning*. 2015. pmlr.
94. Salimans, T. and D.P. Kingma, *Weight normalization: A simple reparameterization to accelerate training of deep neural networks*. *Advances in neural information processing systems*, 2016. **29**.
95. Ba, J.L., J.R. Kiros, and G.E. Hinton, *Layer normalization*. *arXiv preprint arXiv:1607.06450*, 2016.
96. Ulyanov, D., A. Vedaldi, and V. Lempitsky, *Instance normalization: The missing ingredient for fast stylization*. *arXiv preprint arXiv:1607.08022*, 2016.
97. Nam, H. and H.-E. Kim, *Batch-instance normalization for adaptively style-invariant neural networks*. *Advances in Neural Information Processing Systems*, 2018. **31**.
98. Yan, X., et al., *A bearing fault diagnosis method based on PAVME and MEDE*. *Entropy*, 2021. **23**(11): p. 1402.
99. Khan, S., et al., *A guide to convolutional neural networks for computer vision*. Vol. 8. 2018: Springer.
100. Jabir, B. and N. Falih, *Dropout, a basic and effective regularization method for a deep learning model: A case study*. *Indonesian Journal of Electrical Engineering and Computer Science*, 2021. **24**(2): p. 1009-1016.
101. Lydia, A.A. and S. Chandrasekar, *A Comparative Study on Regularization Techniques in Convolutional Neural Networks*.
102. Rajagade, R.A. and C.K. Dewa. *Recognizing Arabic letter utterance using convolutional neural network*. in *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. 2017. IEEE.
103. Li, W., G. Dasarathy, and V. Berisha. *Regularization via structural label smoothing*. in *International Conference on Artificial Intelligence and Statistics*. 2020. PMLR.
104. Krothapalli, U. and A.L. Abbott, *Adaptive label smoothing*. *arXiv preprint arXiv:2009.06432*, 2020.
105. Shorten, C. and T.M. Khoshgoftaar, *A survey on image data augmentation for deep learning*. *Journal of big data*, 2019. **6**(1): p. 1-48.
106. Takahashi, R., T. Matsubara, and K. Uehara, *Data augmentation using random image cropping and patching for deep CNNs*. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019. **30**(9): p. 2917-2931.
107. Wang, K., et al., *Perspective transformation data augmentation for object detection*. *IEEE Access*, 2019. **8**: p. 4935-4943.
108. Yun, S., et al. *Cutmix: Regularization strategy to train strong classifiers with localizable features*. in *Proceedings of the IEEE/CVF international conference on computer vision*. 2019.
109. Zhang, H., et al., *mixup: Beyond empirical risk minimization*. *arXiv preprint arXiv:1710.09412*, 2017.
110. Zhong, Z., et al. *Random erasing data augmentation*. in *Proceedings of the AAAI conference on artificial intelligence*. 2020.

111. DeVries, T. and G.W. Taylor, *Improved regularization of convolutional neural networks with cutout*. arXiv preprint arXiv:1708.04552, 2017.
112. Yoo, J., N. Ahn, and K.-A. Sohn. *Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.
113. Prechelt, L., *Early stopping-but when?*, in *Neural Networks: Tricks of the trade*. 2002, Springer. p. 55-69.
114. Talaei Khoei, T., H. Ould Slimane, and N. Kaabouch, *Deep learning: Systematic review, models, challenges, and research directions*. Neural Computing and Applications, 2023. **35**(31): p. 23103-23124.
115. Tsuneki, M., *Deep learning models in medical image analysis*. Journal of Oral Biosciences, 2022. **64**(3): p. 312-320.
116. Pan, X., et al., *Deep learning for drug repurposing: Methods, databases, and applications*. Wiley interdisciplinary reviews: Computational molecular science, 2022. **12**(4): p. e1597.
117. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
118. Tan, C., et al. *A survey on deep transfer learning*. in *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III* 27. 2018. Springer.
119. Deng, J., et al. *Imagenet: A large-scale hierarchical image database*. in *2009 IEEE conference on computer vision and pattern recognition*. 2009. Ieee.
120. Cook, D., K.D. Feuz, and N.C. Krishnan, *Transfer learning for activity recognition: A survey*. Knowledge and information systems, 2013. **36**: p. 537-556.
121. Cao, X., et al., *Transfer learning for pedestrian detection*. Neurocomputing, 2013. **100**: p. 51-57.
122. Raghu, M., et al., *Transfusion: Understanding transfer learning for medical imaging*. Advances in neural information processing systems, 2019. **32**.
123. Fan, W.-W. and C.-H. Lee, *Classification of imbalanced data using deep learning with adding noise*. Journal of Sensors, 2021. **2021**: p. 1-18.
124. Gu, Q., et al. *Data mining on imbalanced data sets*. in *2008 International Conference on advanced computer theory and engineering*. 2008. IEEE.
125. Jeatrakul, P., K.W. Wong, and C.C. Fung. *Classification of imbalanced data by combining the complementary neural network and SMOTE algorithm*. in *Neural Information Processing. Models and Applications: 17th International Conference, ICONIP 2010, Sydney, Australia, November 22-25, 2010, Proceedings, Part II* 17. 2010. Springer.
126. Anand, R., et al., *An improved algorithm for neural network classification of imbalanced training sets*. IEEE transactions on neural networks, 1993. **4**(6): p. 962-969.
127. Ghosh, K., et al., *The class imbalance problem in deep learning*. Machine Learning, 2022: p. 1-57.
128. Johnson, J.M. and T.M. Khoshgoftaar, *Survey on deep learning with class imbalance*. Journal of Big Data, 2019. **6**(1): p. 1-54.
129. Lee, H., M. Park, and J. Kim. *Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning*. in

- 2016 IEEE international conference on image processing (ICIP). 2016. IEEE.
130. Pouyanfar, S., et al. *Dynamic sampling in convolutional neural networks for imbalanced data classification*. in *2018 IEEE conference on multimedia information processing and retrieval (MIPR)*. 2018. IEEE.
 131. Powers, D.M., *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. arXiv preprint arXiv:2010.16061, 2020.
 132. Parikh, R., et al., *Understanding and using sensitivity, specificity and predictive values*. Indian journal of ophthalmology, 2008. **56**(1): p. 45.
 133. Mishra, D.P., et al., *Various object detection algorithms and their comparison*. Indonesian Journal of Electrical Engineering and Computer Science, 2023. **29**(1): p. 330-338.
 134. Lienhart, R. and J. Maydt. *An extended set of haar-like features for rapid object detection*. in *Proceedings. international conference on image processing*. 2002. IEEE.
 135. Alsharabi, N., *Real-Time Object Detection Overview: Advancements, Challenges, and Applications*. عمران جامعة مجلة, 2023. **3**(6): p. 12-12.
 136. Calonder, M., et al. *Brief: Binary robust independent elementary features*. in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. 2010. Springer.
 137. Freund, Y. and R.E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of computer and system sciences, 1997. **55**(1): p. 119-139.
 138. Cover, T. and P. Hart, *Nearest neighbor pattern classification*. IEEE transactions on information theory, 1967. **13**(1): p. 21-27.
 139. Hakim, H. and A. Fadhil. *Survey: convolution neural networks in object detection*. in *Journal of Physics: Conference Series*. 2021. IOP Publishing.
 140. Lazebnik, S., C. Schmid, and J. Ponce. *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories*. in *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. 2006. IEEE.
 141. He, K., et al., *Spatial pyramid pooling in deep convolutional networks for visual recognition*. IEEE transactions on pattern analysis and machine intelligence, 2015. **37**(9): p. 1904-1916.
 142. Zhao, Z.-Q., et al., *Object detection with deep learning: A review*. IEEE transactions on neural networks and learning systems, 2019. **30**(11): p. 3212-3232.
 143. Simonyan, K. and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556, 2014.
 144. Dai, J., et al., *R-fcn: Object detection via region-based fully convolutional networks*. Advances in neural information processing systems, 2016. **29**.
 145. Liu, W., et al. *Ssd: Single shot multibox detector*. in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. 2016. Springer.
 146. Upulie, H. and L. Kuganandamurthy, *Real-Time Object Detection using YOLO: A review*. Real-Time Object Detection using YOLO: A review, 2021.
 147. Szegedy, C., et al. *Going deeper with convolutions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

148. Redmon, J. and A. Farhadi. *YOLO9000: better, faster, stronger*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
149. Redmon, J. and A. Farhadi, *Yolov3: An incremental improvement*. arXiv preprint arXiv:1804.02767, 2018.
150. Bochkovskiy, A., C.-Y. Wang, and H.-Y.M. Liao, *Yolov4: Optimal speed and accuracy of object detection*. arXiv preprint arXiv:2004.10934, 2020.
151. Wang, C.-Y., A. Bochkovskiy, and H.-Y.M. Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023.
152. Wang, C.-Y., H.-Y.M. Liao, and I.-H. Yeh, *Designing network design strategies through gradient path analysis*. arXiv preprint arXiv:2211.04800, 2022.
153. Peng, C., et al. *Megdet: A large mini-batch object detector*. in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018.
154. Tan, M., R. Pang, and Q.V. Le. *Efficientdet: Scalable and efficient object detection*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.
155. Tan, M. and Q. Le. *Efficientnet: Rethinking model scaling for convolutional neural networks*. in *International conference on machine learning*. 2019. PMLR.
156. Kim, K. and H.S. Lee. *Probabilistic anchor assignment with iou prediction for object detection*. in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV* 16. 2020. Springer.
157. Rezatofighi, H., et al. *Generalized intersection over union: A metric and a loss for bounding box regression*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.
158. Zheng, Z., et al. *Distance-IoU loss: Faster and better learning for bounding box regression*. in *Proceedings of the AAAI conference on artificial intelligence*. 2020.
159. Du, S., et al. *An improved bounding box regression loss function based on CIOU loss for multi-scale object detection*. in *2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML)*. 2021. IEEE.
160. Zhang, Y., et al., *Focal and efficient IOU loss for accurate bounding box regression*. arXiv 2021. arXiv preprint arXiv:2101.08158.
161. He, Y., et al. *Bounding box regression with uncertainty for accurate object detection*. in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.
162. Gefan, Y. and L. Yuchi, *Object Detection in the KITTI Dataset using YOLO and Faster R-CNN*.
163. Kachhava, R., et al., *Security system and surveillance using real time object tracking and multiple cameras*. Advanced Materials Research, 2012. **403**: p. 4968-4973.
164. Nevcihan, D., *Deep Learning Applications in Medical Image Processing*. Acta Infologica, 2021. **5**(2): p. 459-474.
165. Yao, J., et al., *High-efficiency classification of white blood cells based on object detection*. Journal of Healthcare Engineering, 2021. **2021**: p. 1-11.

166. Sun, Z., G. Bebis, and R. Miller, *On-road vehicle detection: A review*. IEEE transactions on pattern analysis and machine intelligence, 2006. **28**(5): p. 694-711.
167. Cheong, Y.Z. and W.J. Chew. *The application of image processing to solve occlusion issue in object tracking*. in *MATEC Web of Conferences*. 2018. EDP Sciences.
168. Feng, Q., X. Xu, and Z. Wang, *Deep learning-based small object detection: A survey*. Mathematical Biosciences and Engineering, 2023. **20**(4): p. 6551-6590.
169. Liang, J. and J. Ai. *Improving Accuracy of Small Object Detection in Screen based on YOLOv7*. in *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*. 2023. IEEE.
170. Vaishnavi, K., et al., *Real-time Object Detection Using Deep Learning*.
171. Tavakoli, H., et al., *Object Detection for Human–Robot Interaction and Worker Assistance Systems*, in *Artificial Intelligence in Manufacturing: Enabling Intelligent, Flexible and Cost-Effective Production Through AI*. 2023, Springer Nature Switzerland Cham. p. 319-332.
172. M Ashir, A., *A Transfer-Learning-Based Approach for Emergency Vehicle Detection*. Eurasian Journal of Science and Engineering, 2022. **8**(1).
173. Kırca, S., *Derin öğrenme yöntemi ile araç ve plaka tanıma*. 2021, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü.
174. Zhang, Y., et al., *Real-time vehicle detection based on improved yolo v5*. Sustainability, 2022. **14**(19): p. 12274.
175. Priymak, M. and R. Sinnott. *Real-Time Traffic Classification through Deep Learning*. in *2021 IEEE/ACM 8th International Conference on Big Data Computing, Applications and Technologies (BDCAT'21)*. 2021.
176. Kasper-Eulaers, M., et al., *Detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5*. Algorithms, 2021. **14**(4): p. 114.
177. Zhang, K., et al., *Research on mine vehicle tracking and detection technology based on YOLOv5*. Systems Science & Control Engineering, 2022. **10**(1): p. 347-366.
178. Mim, F.S., S.N.R. Sayam, and M.T. Amin, *Traffic Participants Detection and Classification Using YOLO Neural Network*. LC International Journal of STEM (ISSN: 2708-7123), 2022. **3**(2): p. 9-18.
179. Andhy Panca Saputra, K., *Waste Object Detection and Classification using Deep Learning Algorithm: YOLOv4 and YOLOv4-tiny*. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021. **12**(6): p. 5583-5595.
180. Qazzaz, A.A. and A.Y. Abdulkadhim, *Car Detection and Features Identification Based on YOLOV5*. International Journal of Mechanical Engineering. ISSN: p. 0974-5823.
181. Krishnapriya, S. and Y. Karuna, *A deep learning model for the localization and extraction of brain tumors from MR images using YOLOv7 and grab cut algorithm*. Frontiers in Oncology, 2024. **14**.

9. EKLER

9.1. EK A.1. Gradyanların Hesaplanması ve Ağırlıkların Güncellenmesi

Tamamen bağı katmandaki kayıpların ağırlıklara ve sapmalara göre eğimi hesaplanır, öğrenme katsayısı ile kayıp eğimleri çarpılır ve yeni ağırlıklar ve sapmalar güncellenir. Bu işlemleri basit bir örnek tanımlayarak gradyan hesaplamalarını gösterelim. İki örnekten oluşan bir X matrisi, ağırlıkları temsil eden 4x9 boyutlu bir W matrisi, sapmaları temsil eden tek boyutlu bir B matrisi ve one-hot kodlu gerçek değerleri temsil eden bir matris tanımlayarak önce çıkış değerlerini, sonra bu çıkışları softmax fonksiyonu ile sınıflandırarak değerleri normalize edelim. Sonra kayıpları hesaplayarak kaybın ağırlıklara göre gradyanlarını hesaplayarak bu işlemleri örnek hesaplamalar ile anlatmaya çalışacağız. Örnek olarak iki örnekli bir X girdi matrisi, ağırlıkları temsil eden 4x9 boyutunda bir ağırlık matrisi, 1x4 boyutlu bir sapma matrisi, 2x4 boyutlu gerçek değerleri temsil eden one-hot kodlanmış bir P matrisi olsun.

$$X = \begin{bmatrix} 0,19 & 0,35 & 0,55 & 0,70 & 0,65 & 0,15 & 0,25 & 0,95 & 0,45 \\ 0,25 & 0,15 & 0,56 & 0,75 & 0,85 & 0,90 & 0,85 & 0,45 & 0,15 \end{bmatrix}$$

$$W = \begin{bmatrix} 0,10 & 0,45 & 0,95 & 0,56 \\ 0,25 & 0,55 & 0,85 & 0,65 \\ 0,15 & 0,65 & 0,75 & 0,70 \\ 0,35 & 0,75 & 0,65 & 0,85 \\ 0,50 & 0,85 & 0,12 & 0,25 \\ 0,65 & 0,90 & 0,25 & 0,12 \\ 0,70 & 0,45 & 0,30 & 0,24 \\ 0,15 & 0,15 & 0,45 & 0,35 \\ 0,25 & 0,18 & 0,56 & 0,45 \end{bmatrix}, B = [0,001 \quad 0,002 \quad 0,005 \quad 0,004]$$

$$Sınıflar = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Araba \\ Bisiklet \\ Kamyon \\ Otobüs \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} Araba \\ Kamyon \end{bmatrix}$$

Tamamen bağı katmandaki çıkışları hesaplamak için denklem 9.1' deki formül kullanılır.

$$z_{sj} = b_j + \sum_{i=1}^n x_{is} * w_{ij}, \quad (9.1)$$

Denklem 9.1' deki formül kullanılarak Z çıktı matrisi elde edilir.

$$Z = \begin{bmatrix} 1,288 & 2,186 & 2,221 & 2,093 \\ 2,12 & 3,133 & 2,146 & 2,021 \end{bmatrix}$$

Z çıktı matrisi elde edildikten sonra bu değerler softmax fonksiyonu ile normalize edilmesi gerekir. Softmax fonksiyonu ile sınıf skorları elde edilir. Softmax fonksiyonu elde edilen bu skorların toplamının “1” olduğundan emin olur. Softmax fonksiyonu denklem 9.2’ de tanımlanmıştır.

$$y_{sj} = \text{Softmax} \left(\frac{e^{z_{sj}}}{\sum_{k=1}^k e^{z_{sj}}} \right) \quad (9.2)$$

Bu softmax fonksiyonu ile girdi değerleri hesaplanarak normalize edilir. İki adet girdi örneği olduğu için iki örneğin de çıktıları aşağıdaki şekilde hesaplanır.

$$y_{11} = \text{Softmax} \left(\frac{e^{z_1}}{\sum_{k=1}^k e^{z_1}} \right), y_{12} = \text{Softmax} \left(\frac{e^{z_2}}{\sum_{k=1}^k e^{z_2}} \right),$$

$$y_{13} = \text{Softmax} \left(\frac{e^{z_3}}{\sum_{k=1}^k e^{z_3}} \right), y_{14} = \text{Softmax} \left(\frac{e^{z_4}}{\sum_{k=1}^k e^{z_4}} \right)$$

$$y_{11} = \frac{e^{1,288}}{e^{1,288} + e^{2,186} + e^{2,221} + e^{2,093}}, y_{12} = \frac{e^{2,186}}{e^{1,288} + e^{2,186} + e^{2,221} + e^{2,093}},$$

$$y_{13} = \frac{e^{2,221}}{e^{1,288} + e^{2,186} + e^{2,221} + e^{2,093}}, y_{14} = \frac{e^{2,093}}{e^{1,288} + e^{2,186} + e^{2,221} + e^{2,093}}$$

$$y_{11} = \frac{3,626}{29,852}, y_{12} = \frac{8,9}{29,852}, y_{13} = \frac{9,217}{29,852}, y_{14} = \frac{8,109}{29,852} \Rightarrow$$

$$y_{11} \approx 0,121, y_{12} \approx 0,298, y_{13} \approx 0,309, y_{14} \approx 0,272$$

$$y_{21} = \frac{e^{2,12}}{e^{2,12} + e^{3,133} + e^{2,146} + e^{2,021}}, y_{22} = \frac{e^{3,133}}{e^{2,12} + e^{3,133} + e^{2,146} + e^{2,021}},$$

$$y_{23} = \frac{e^{2,146}}{e^{2,12} + e^{3,133} + e^{2,146} + e^{2,021}}, y_{24} = \frac{e^{2,021}}{e^{2,12} + e^{3,133} + e^{2,146} + e^{2,021}}$$

$$y_{21} = \frac{8,331}{47,369}, y_{22} = \frac{22,942}{47,369}, y_{23} = \frac{8,551}{47,369}, y_{24} = \frac{7,545}{47,369} \Rightarrow$$

$$y_{21} \approx 0,176, y_{22} \approx 0,484, y_{23} \approx 0,181, y_{24} \approx 0,159$$

Her iki girdi örneği softmax fonksiyonu ile normalize edilip sınıflandırıldıktan sonra kayıplar gerçek değerler ile karşılaştırılarak hesaplanır. Kayıplar denklem 9.3’ deki katogorik kayıp hesaplama fonksiyonu kullanılarak hesaplanır.

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C x_{ij} \log(y_{ij}) \quad (9.3)$$

Kayıp hesaplama formülü kullanılarak her iki örnek için modelin ürettiği tahmin ile gerçek değerler karşılaştırılarak L_1 ve L_2 kayıpları ve bu kayıpların ortalaması hesaplanır.

$$L_1 = -(1 * \ln(0,121) + (0 * \ln(0,298)) + (0 * \ln(0,309)) + (0 * \ln(0,272)))$$

$$\Rightarrow L_1 = 2,112$$

$$L_2 = -(0 * \ln(0,176) + (0 * \ln(0,484)) + (1 * \ln(0,181)) + (0 * \ln(0,159)))$$

$$\Rightarrow L_2 = 1,709$$

$$L = \frac{1}{N}(L_1 + L_2) \Rightarrow L = \frac{1}{2}(2,112 + 1,709) = \frac{3,821}{2} \Rightarrow L = 1,911$$

Daha sonra geri yayılım yapmak için kaybın çıktıya göre gradyanların hesaplanmasında denklem 9.4' deki formül kullanılır ve $\frac{\partial L}{\partial y_i}$ elde edilir.

$$L = -\sum_{i=1}^C x_i \log(y_i), \quad \frac{\partial L}{\partial y_i} = -\frac{x_i}{y_i} \quad (9.4)$$

Aynı şekilde, softmax fonksiyonundan elde edilen çıktıların girdi verisine göre gradyanları denklem 9.5' deki gibi hesaplanır ve $\frac{\partial y_i}{\partial z_j}$ elde edilir.

$$y_j = \left(\frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \right), \quad \frac{\partial y_i}{\partial z_j} = y_i(\delta_{ij} - y_i) \quad (9.5)$$

Daha sonra $\frac{\partial L}{\partial z_i}$ yi hesaplamak için calculusun zincir kuralından faydalanılır.

Kaybın girdi verisine($\frac{\partial L}{\partial z_i}$) göre hesaplanması denklem 9.6' daki gibidir.

$$\frac{\partial L}{\partial z_i} = \sum_{j=1}^N \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial z_i} = \sum_{j=1}^N \left(-\frac{x_i}{y_i}\right) * y_i(\delta_{ij} - y_i) \Rightarrow \frac{\partial L}{\partial z_i} = (y_i - x_i) \quad (9.6)$$

Burada, $\frac{\partial L}{\partial z_i} = (y_i - x_i)$ elde edildikten sonra elde edilen veriler yerine konur.

$$\frac{\partial L}{\partial z_i} = (y_i - x_i) \Rightarrow \begin{bmatrix} 0,121 & 0,298 & 0,309 & 0,272 \\ 0,176 & 0,484 & 0,181 & 0,159 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \Rightarrow$$

$$\frac{\partial L}{\partial z_i} = \begin{bmatrix} -0,879 & 0,298 & 0,309 & 0,272 \\ 0,176 & 0,484 & -0,819 & 0,159 \end{bmatrix}$$

Daha sonra, girdi verilerinin ağırlıklara göre gradyanları hesaplanır. Denklem 9.6' da girdi verilerini hesaplayan fonksiyon, denklem 9.7' de girdi verilerinin ağırlıklara göre gradyanını hesaplayan formül tanımlanmıştır.

$$z_i = w_{ij} x_j + b_j \quad (9.6)$$

$$\frac{\partial z_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} (w_{ij} x_j + b_j) = \frac{\partial}{\partial w_{ij}} (w_{ij} x_j) + \frac{\partial}{\partial w_{ij}} (b_j) = \frac{\partial z_i}{\partial w_{ij}} = x_j \quad (9.7)$$

$\frac{\partial L}{\partial z_i}$ ve $\frac{\partial z_i}{\partial w_{ij}}$ gradyan hesaplamaları elde edildikten sonra kayıpların ağırlıklara

göre gradyanlarını ($\frac{\partial L}{\partial w_{ij}}$) hesaplamak için calculusun zincir kuralından faydalınılır ve

hesaplanan diğer gradyanlar denklem 9.8' de yerine konur.

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}} = (y_i - x_i) x_j \Rightarrow \frac{\partial L}{\partial W} = \frac{\partial L}{\partial Z} \frac{\partial Z}{\partial W} = X^T * \frac{\partial L}{\partial Z} \quad (9.8)$$

$\frac{\partial L}{\partial W}$ yi elde etmek için girdi matrisinin transpozu ile $\frac{\partial L}{\partial Z}$ çarpılarak kaybın ağırlıklara göre gradyan matrisi elde edilir.

$$\frac{\partial L}{\partial W} = \begin{bmatrix} 0.19 & 0.25 \\ 0.35 & 0.15 \\ 0.55 & 0.56 \\ 0.70 & 0.75 \\ 0.65 & 0.85 \\ 0.15 & 0.90 \\ 0.25 & 0.85 \\ 0.95 & 0.45 \\ 0.45 & 0.15 \end{bmatrix} \begin{bmatrix} -0,879 & 0,298 & 0,309 & 0,272 \\ 0,176 & 0,484 & -0,819 & 0,159 \end{bmatrix} \Rightarrow$$

$$\frac{\partial L}{\partial W} = \begin{bmatrix} -0,123 & 0,177 & -0,146 & 0,091 \\ -0,281 & 0,177 & -0,0147 & 0,119 \\ -0,385 & 0,435 & -0,289 & 0,239 \\ -0,483 & 0,571 & -0,398 & 0,310 \\ -0,422 & 0,605 & -0,495 & 0,312 \\ 0,027 & 0,480 & -0,691 & 0,184 \\ -0,07 & 0,486 & -0,619 & 0,203 \\ -0,756 & 0,5 & -0,075 & 0,330 \\ 0,369 & 0,207 & 0,016 & 0,146 \end{bmatrix}$$

Ağırlıklara göre kayıpların gradyanları ($\frac{\partial L}{\partial W}$) elde edildikten sonra, gradyan iniş algoritması ile yeni ağırlıkların güncellemesi yapılır. Gradyan iniş algoritması denklem 9.9' da tanımlanmıştır.

$$W^* = W - \eta * \frac{\partial L}{\partial W} \quad (9.9)$$

Burada, W^* yeni ağırlıkları, W önceki ağırlıkları, η öğrenme katsayısını, $\frac{\partial L}{\partial W}$ kaybın ağırlıklara göre gradyanlarını temsil eder. $\eta = 0,001$ öğrenme oranı olarak kabul edilip elde edilen değerler denklem 9.9' daki formülde yerine konursa güncellenecek yeni ağırlıklar bulunur.

$$W^* = \begin{bmatrix} 0.10 & 0.45 & 0.95 & 0.56 \\ 0.25 & 0.55 & 0.85 & 0.65 \\ 0.15 & 0.65 & 0.75 & 0.70 \\ 0.35 & 0.75 & 0.65 & 0.85 \\ 0.50 & 0.85 & 0.12 & 0.25 \\ 0.65 & 0.90 & 0.25 & 0.12 \\ 0.70 & 0.45 & 0.30 & 0.24 \\ 0.15 & 0.15 & 0.45 & 0.35 \\ 0.25 & 0.18 & 0.56 & 0.45 \end{bmatrix} - 0,001 * \begin{bmatrix} -0,123 & 0,177 & -0,146 & 0,091 \\ -0,281 & 0,177 & -0,0147 & 0,119 \\ -0,385 & 0,435 & -0,289 & 0,239 \\ -0,483 & 0,571 & -0,398 & 0,310 \\ -0,422 & 0,605 & -0,495 & 0,312 \\ 0,027 & 0,480 & -0,691 & 0,184 \\ -0,07 & 0,486 & -0,619 & 0,203 \\ -0,756 & 0,5 & -0,075 & 0,330 \\ 0,369 & 0,207 & 0,016 & 0,146 \end{bmatrix}$$

Bu işlemin sonunda yeni ağırlıklar tespit edilir. Aynı şekilde kayıpların sapmalara göre gradyanları($\frac{\partial L}{\partial B}$) da hesaplanır. Bu işlemler geriye doğru bütün katmanlarda uygulanır. Bütün katmanlarda da ağırlık ve sapmaların gradyanlarını hesaplamak için calculusun zincir kuralından faydalanılır. Buradaki amaç türev fonksiyonundan faydalanılarak kayıpları en aza indirmektir. Her bir eğitim döngüsünde bu işlemler tekrarlanarak modelin en uygun ağırlıklara sahip olması beklenir. Model maksimum performansa ulaştığında modelin tahminlemesinde en iyi ağırlıklara ve sapmalara sahip olunur.