

İTÜ



DERİN ÖĞRENME İLE GERÇEK ZAMANLI GÖRÜNTÜLERDEN ATIK TESPİTİ UYGULAMASI

Hazırlayan

: **Alp Doğan FİDAN**

Öğrenci No

: **090180320**

Teslim Tarihi

: **12.01.2024**

Ders

: **MAT 4902**

Danışman

: **DR. ÖĞR. ÜYESİ Bahri GÜLDOĞAN**

İÇİNDEKİLER

İÇİNDEKİLER	1
ŞEKİLLER LİSTESİ	4
TABLolar LİSTESİ	7
KISALTMA LİSTESİ	8
BÖLÜM 1	9
GİRİŞ	9
1.1. Literatür Taraması	10
BÖLÜM 2	13
ÇEVRE ve ATIKLAR	13
2.1. Çevre ve Ekosistem	13
2.1. Çevresel Atıklar	14
2.1. Atık Yönetimi	16
BÖLÜM 3	17
BİLGİSAYARLI GÖRÜ TEKNOLOJİSİ	17
3.1. Dijital Görüntü	17
3.2. Görüntü İşleme ve Bilgisayarlı Görü	18
3.2.1. Görüntü İşleme	18
3.2.2. Bilgisayarlı Görü	19
BÖLÜM 4	21
KONVOLÜSYONEL SİNİR AĞLARI	21
4.1. Makine Öğrenmesi	21
4.1.1. Öğrenme Çeşitleri	22
Denetimli Öğrenme (Supervised Learning)	22
Denetimsiz Öğrenme (Unsupervised Learning)	23
4.1.2. Modelin Öğrenmesi	23
Kayıp Fonksiyonu (Loss Function)	23
Optimizasyon Fonksiyonu (Optimization Function)	24
4.1.3. Yetersiz Öğrenme (Underfitting) ve Aşırı Öğrenme (Overfitting)	26
4.1.4. Yapay Sinir Ağları	27
Tek Katmanlı Algılayıcı (Perceptron)	28
Çok Katmanlı Algılayıcı (Multi Layer Perceptron)	30
4.2. Derin Öğrenme	32
4.3. Konvolüsyonel Sinir Ağları	33
4.3.1. Konvolüsyonel Sinir Ağları Mimarisi	36
Girdi Katmanı	36

Konvolüsyon (Filtre) Katmanı	36
Toplu Normalizasyon (Batch Normalization) Katmanı.....	40
Rektifiye Edilmiş Doğrusal Birim Katmanı (ReLU: Rectified Linear Units Layer)	40
Ortaklama Katmanı (Pooling Layer)	42
Düzleştirme Katmanı (Flattening)	43
Tam Bağlı Katman	44
Sönümleme (Dropout) Katmanı.....	44
Sınıflandırma Katmanı.....	45
Kayıp Fonksiyonu	45
4.3.2. Konvolüsyonel Sinir Ağları'nda Yaygın Karşılaşılan Zorluklar.....	46
4.3.3. ResNet, DenseNet ve CSPDenseNet.....	47
4.4. CNN ile Nesne Tespiti	50
4.4.1. İki Aşamalı (Two-Shot) Nesne Tespiti.....	51
4.4.2. Tek Aşamalı (One-Shot) Nesne Tespiti	51
BÖLÜM 5.....	52
MATERYAL-METHOD.....	52
5.1. YOLO Algoritması	52
5.1.1. YOLO Genel Yapısı.....	56
5.1.2. YOLO Tarihi	58
5.2. Uygulama.....	60
5.2.1. Veri Seti	61
Veri Önişleme (Data Preprocessing)	63
Veri Arttırma (Data Augmentation)	63
Eğitim-Test-Validasyon Ayırımı.....	64
Veri Setinin Hazır Hale Getirilmesi.....	65
5.2.2. Başarı Metriği	66
5.2.3. Öğrenimin Transferi (Transfer Learning)	67
5.2.4. Modelin Eğitimi	68
5.3. Araştırma Sonuçları	73
5.3.1. Model Kayıpları	73
5.3.2. mAP ve F1 Skorları	75
5.3.3. Sınıf Tespiti.....	76
5.3.4. Karmaşıklık Matrisi	77
BÖLÜM 6.....	79
SONUÇ	79
6.1. Tartışma	80

ŞEKİLLER LİSTESİ

Şekil 1: İnsan ve doğa	13
Şekil 2: Çöple dolu doğa ve insan resmi	14
Şekil 3: İnşaat, organik, batarya, pet, tıbbi, raddrasyon atıkları (Orak, Tarih yok).....	15
Şekil 4: Farklı tür atıkların geri dönüşüm kutuları	15
Şekil 5: Sosyal çevre bilinci.....	16
Şekil 6: Elektromanyetik spektrum (Kurban, Tarih yok).....	17
Şekil 7: Analog sinyalin dijital sinyale dönüşümü (Kurban, Tarih yok)	17
Şekil 8: RGB renk bantları (Kurban, Tarih yok)	18
Şekil 9: Girdi görüntüye uygulanan yumuşatma ve keskinleştirme işlemleri (Barreto, 2023)	19
Şekil 10: Girdi görüntüye görüntü işleme ve bilgisayarlı görünün uygulanması (Barreto, 2023)	20
Şekil 11: Yapay Zeka, Makine Öğrenmesi, Derin Öğrenme ve Yapay Sinir Ağlarına ilişkin şema (Carpenter, Cohen, Jarrell, & Huang, 2018).....	21
Şekil 12: Küçük, uygun ve aşırı büyük öğrenme oranlarının seçilmesi (Cayla, 2021)	25
Şekil 13: Sırayla Underfitting, Düzgün Öğrenme ve Overfitting karşılaştırması (Forjan, 2021)	26
Şekil 14: İnsan sinir hücresinin yapısı (Agatonovic-Kustrin & Beresford, 2000)	27
Şekil 15: Biyolojik Sinir Ağları ve Yapay Sinir Ağları bileşenlerinin denklikleri (Artificial Neural Network Tutorial)	28
Şekil 16: Tek Katmanlı Algılamacı modeli (Güzel, 2018)	28
Şekil 17: Çok Katmanlı Algılamacı modeli (Lee, Kim, & Lee, 2020)	30
Şekil 18: Geleneksel Makine Öğrenmesinde ve Derin Öğrenmede özellik seçimi karşılaştırması (Alzubaidi, ve diğerleri, 2021)	32
Şekil 19: ILSVRC-2010 verisetine ait 8 adet test görüntüsü ve AlexNet modelinin görüntüler için sunduğu yüksekten küçüğe sıralanmış en olası 5 etiket tahmini (Krizhevsky, Sutskever, & Hinton, 2017)	33
Şekil 20: İnsana kıyasla derin öğrenme performansı (Alzubaidi, ve diğerleri, 2021)	34
Şekil 21: 3x3 kernel'e sahip 13 adet Konvolusyon katmanı ile 3 adet Tam Bağlı Katman'dan oluşan VGG-16 mimarisi (Sharma, 2020)	34
Şekil 22: Otonom sürüslerde nesne tespiti (Barla, 2023)	35
Şekil 23: CNN modelinin örnek gösterimi (Anding, Haar, Polte, Walz, & Notni, 2019)	36
Şekil 24: Resim üzerinde filtre etkileri (Doğan, 2020)	37
Şekil 25: CNN'de farklı katmanlarda meydana gelen nesne temsilleri (İnik & Ülker, 2017) ...	38
Şekil 26: Receptive field ile Kernel'in nokta çarpımı (Singh, Meitei, & Majumder, 2020)	39
Şekil 27: Örnek 96 Adet Konvolusyon Kernel'i (Krizhevsky, Sutskever, & Hinton, 2017)	40
Şekil 28: 4x4 boyutundaki matrisin ReLU fonksiyonuna girmeden önceki ve girdikten sonraki durumu (Taşhan, 2017)	41
Şekil 29: Giriş görüntüsünün sırasıyla konvolusyon ve ReLU katmanlarından geçtikten sonraki çıktı görüntüleri (İnik & Ülker, 2017)	41
Şekil 30: Stride değeri 2 iken 2x2 boyutlu max pooling ve average pooling işlemleri (Raitoharju, 2022).....	42
Şekil 31: ReLU katmanından çıkan görüntünün Pooling katmanında işlem gördükten sonraki çıkan görüntüsü (İnik & Ülker, 2017).....	43
Şekil 32: Matrisin Düzleştirme (Flattening) Katmanı ile vektör haline getirilmesi (Keskin, 2022)	43
Şekil 33: Basit bir Tam Bağlı Katman yapısı (Keskin, 2022).....	44

Şekil 34: Solda 2 gizli katmandan oluşan bir sinir ağı ve sağda Dropout katmanının uygulanmasından sonra oluşan ağ (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014)	44
Şekil 35: Solda normal CNN blok ve sağda Residual blok (Dive Into Deep Learning, Tarih yok).	47
Şekil 36: VGG-19 ve ResNet (He, Zhang, Ren, & Sun, 2015)	48
Şekil 37: DenseNet örneği (Wang, ve diğerleri, 2019).....	49
Şekil 38: CSPDenseNet örneği (Wang, ve diğerleri, 2019).....	49
Şekil 39: Classification, Localization ve Detection farkı (Singh, 2020)	50
Şekil 40: Tek ve çift aşamalı yöntemlerin işleyışı (Sharma, 2022).....	50
Şekil 41: YOLO modeline genel bakış (Redmon, Divvala, Girshick, & Farhadi, 2016)	52
Şekil 42: YOLOv1'in mimarisi (Redmon, Divvala, Girshick, & Farhadi, 2016)	53
Şekil 43: SxS grid, sınırlayıcı kutular, sınıf olasılık haritası ve tespitler (Redmon, Divvala, Girshick, & Farhadi, 2016).....	53
Şekil 44: Bir nesne için tahmin edilen ve kesin referans sınırlayıcı kutular	54
Şekil 45: Non-maximal Suppression (Singh, 2020).....	55
Şekil 46: Dedektörlerin yapısı (Bochkovskiy, Wang, & Liao, 2020)	56
Şekil 47: PANet (Liu, Qi, Qin, Shi, & Jia, 2018)	56
Şekil 48: YOLOv5 mimarisi için basitleştirilmiş bir gösterim (Nepal & Eslamiyat, 2022).	59
Şekil 49: YOLOv8 mimarisi için detaylı bir gösterim (King, 2023)	59
Şekil 50: Uygulamanın genel akış diyagramı	60
Şekil 51: YOLO model spektrumu (Jocher, Chaurasia, & Qiu, 2023)	60
Şekil 52: Verisetindeki örnek görseller	61
Şekil 53: Veriseti için hazırlanan görüntülerin sağlık durumu	62
Şekil 54: Verilerin etiketlenmesi.....	62
Şekil 55: Etiketlenen görsele ait txt uzantılı dosya içeriği.....	63
Şekil 56: Veriseti preprocessing adımı.....	63
Şekil 57: Verisetinde kullanılan veri artırma teknikleri	64
Şekil 58: Veriseti train-valid-test ayrimı	64
Şekil 59: Eğitim setinin çoğlanması	65
Şekil 60: Verisetinin modele uygun formatının seçilmesi.....	65
Şekil 61: Çalışmada kullanılan YAML dosyasının içeriği.....	68
Şekil 62: Çalışma Ortamına Drive'in bağlanması ve verisetinin açılması	68
Şekil 63: Çalışma Ortamına Github'ın klonlanması ve GPU'nun bağlanması	69
Şekil 64: Eğitimde kullanılan GPU özellikleri	69
Şekil 65: YOLOv5m modelinin eğitimi	70
Şekil 66: YOLOv5m modelinin genel yapısı	70
Şekil 67: YOLOv5m eğitimi anından bir görüntü	71
Şekil 68: Eğitim sırasında kullanılan kaynakların durumu	72
Şekil 69: Modelin değerlendirdiği validasyon görüntüleri ve güven skorları	72
Şekil 70: YOLOv5 Modelinin 125 Epokluk Eğitim Performans Çıktıları.....	74
Şekil 71: YOLOv8 Modelinin 125 Epokluk Eğitim Performans Çıktıları.....	74
Şekil 72: YOLOv5 modelinin Test kümesi için Precision-Recall Eğrisi ve F1-Confidence Eğrisi Çıktıları	75
Şekil 73: YOLOv8 modelinin Test kümesi için Precision-Recall Eğrisi ve F1-Confidence Eğrisi Çıktıları	75
Şekil 74: YOLOv5 modelinin normalized edilmiş karmaşıklık matrisi.....	77

Şekil 75: YOLOv8 modelinin normalized edilmiş karmaşıklık matrisi..... 78

Şekil 76: Gerçek videoya ait farklı açılardan tespit edilen nesnenin güven skorları 79

TABLOLAR LİSTESİ

Tablo 1: YOLO'nun farklı sürümleri ve değişen yapısı (Nepal & Eslamiat, 2022).....	58
Tablo 2: Karmaşıklık matrisi	66
Tablo 3: Modellerin katman ve ağırlık miktarları	71
Tablo 4: Modellerin optimizer ve hiperparametreleri	71
Tablo 5: Sınıfların YOLOv5m modeline göre test başarısı	76
Tablo 6: Sınıfların YOLOv8m modeline göre test başarısı	76

KISALTMA LİSTESİ

- TÜİK** : Türkiye İstatistik Kurumu
RGB : Red Green Blue
MSE : Mean Squared Error
ANN : Artificial Neural Network
CNN : Convolutional Neural Network
ReLU : Rectified Linear Unit
R-CNN : Region Based Convolutional Neural Network
RFCN : Region-based Fully Convolutional Networks
SSD : Single Shot Multibox Detector
YOLO : You Look Only Once
AP : Average Precision
mAP : Mean Average Precision
IOU : Intersection over Union

BÖLÜM 1.

GİRİŞ

Bu giriş bölümünde, tasarımın tanımı, amacı ve kapsamı hakkında bilgi verilecek ve yapılan çalışmaların genel bir özeti yer verilecektir.

Önceki dönem; günümüzde, makine öğrenmesi, yapay sinir ağları ve derin öğrenme gibi konseptler bilgisayarlı görüp sektöründe önemli bir etkiye sahiptir. 2010'lu yılların başından itibaren obje tanıma, obje sınıflandırma ve örüntü tanıma gibi bilgisayarlı görüp görevlerindeki göze çarpan başarısı nedeniyle derin öğrenme alanında önemli bir yere sahip olan Konvolüsyonel Sinir Ağları (CNN: Convolutional Neural Network) hakkında kapsamlı bir literatür araştırması yapılmıştır.

Bu dönemde de, tasarım projesinin kapsamında mevcut bilimsel makaleleri, konferans bildirileri ve diğer kaynaklar incelenerek, state-of-the-art CNN mimarisine sahip YOLO modeli seçilmiş ve bu modelin genel yapısı, mimari özellikleri ve uygulama alanları araştırılmış ve tanıtılmıştır. Atık veriseti bulunmuş olup %70-%15-%15 train-valid-test ayrimı yapılmıştır. Bu veriseti ile YOLOv5m ve YOLOv8m modelleri seçilerek gerçek zamanlı nesne algılama görevi için Google Colab ortamında eğitilmiştir. Eğitilen modellerin performans çıktıları değerlendirilmiştir ve sırasıyla %74.8 ve %74.1 mAP50 skoru elde edilmiştir. Son olarak tespit için Python uygulaması yazılmıştır.

1.1. Literatür Taraması

Derin öğrenme alanında literatür taraması yapılmıştır. Öncelikle, derin öğrenmenin tarihinde denenmiş ve başarılı sonuçlar elde edilmiş çeşitli yaklaşımı incelemek önemlidir. Ayrıca, önceki araştırmacıların deneme-yanılma yoluyla elde ettikleri bilgileri anlamak da gereklidir, çünkü bazı yaklaşımalar başarısız olmuş olabilir ve bu hatalardan öğrenmek, ileriki çalışmalara ışık tutabilir.

2021 yılında gerçekleştirilen bir çalışmada, 100 Çinli vatandaştan oluşan bir grup yardımıyla 15 çeşit Çin rakam birimini içeren toplam 15000 adet 64×64 boyutunda görüntünden oluşan bir veri seti hazırlanmıştır. Diğer yandan 256, 128 ve 64 adet 3×3 'lük filtre içeren 3 adet konvolüsyon katmanından oluşan CNN mimarisi tasarlanmıştır. Bu konvolüsyon katmanlarından sonra ise aktivasyon katmanı olarak ReLU, pooling katmanı olarak ise max pooling kullanılmıştır. Tasarlanan mimari toplamda 8,792,783 parametre içermektedir. Sonuç olarak ilgili veriseti ise bu CNN mimarisi kullanılarak eğitilmiş olup test verisinde %97 doğruluk oranı ile %13 hata oranı elde edilmiştir. (Kayalı & Omurca, 2021).

CNN'in başarısı farklı görüntü boyutlarında ve yeterli-yetersiz koşullara göre değişebilmektedir. Yine 2021 yılında yayımlanan başka bir makalede, aynı koşullar altında fakat farklı nitelikte veri setleri üzerinde geleneksel makine öğrenmesi algoritması olan Support Vector Machine (SVM) ile CNN modelleri eğitilip karşılaştırılmıştır. Veri setindeki görüntülerin boyutu sırayla 64×64 , 128×128 ve 256×256 alındığında SVM modelinin sınıflandırma başarısı yaklaşık olarak %61-%64 arasında değişirken CNN'in sınıflandırma doğruluğunu %71'den %95'e ulaştığı tespit edilmiştir. Yine aynı çalışmada COREL1000 veri setinin küçük bir örneği ile modeller eğitildiğinde SVM %86 doğruluk sağlarken CNN %83 doğruluk sağlamıştır. MNIST'den alınan büyük veri seti üzerinde ise SVM ve CNN modelleri sırayla %88 ve %98 doğruluğa sahip olmuştur (Wanga, Fan, & Wang, 2021).

Kahramanmaraş Ceyhan Havzası'nda bulunan yarıklı balık geçidine, su altı çekimlerinden alınan 400 adet gerçek veriye dayanarak, bulanık ve gürültülü sulardaki balıkları tespit etmek amacıyla YOLO-V2, YOLO-V3, YOLO-V3 Tiny ve MobileNet-SSD derin öğrenme modelleri karşılaştırılmıştır. Değerlendirme neticesinde, MobileNet-SSD modelinin eğitim verisinin yarısını kullandığında diğer modellere göre daha üstün bir performans sergileyerek %88.07 ortalama kesinlik değeri sağladığı görülmüştür (Akgül, Çalık, & Töreyin, 2020).

Ammour ve meslektaşlarının, 2017 yılında yaptıkları çalışmada İHA kaynaklarından elde edilen çok yüksek çözünürlüklü (VHR) ve yüksek detaylı gerçek görüntülerle araç tespiti yapmak için

dört temel adımı takip etmişler. İlk olarak, alınan görüntülerin araç olup olmadığına dair belirlemek için ortalama kaydırma (mean-shift) algoritmasını kullanarak farklı bölgelere bölmüşler. İkinci adımda, bu bölgeler önceden eğitilmiş VGG-16 CNN ağına besleme yaparak özellik çıkarma işlemini gerçekleştirmiştir. Üçüncü evrede de destek vektör makinesi (SVM) ile sınıflandırma yapmışlar. En sonunda, belirlenen nesneler üzerinde morfolojik işlemler için fine-tuning uygulayıp çalışmanın sonuçlarına ulaşmışlardır. Çalışmadaki uygulanan yöntemin, diğer state-of-the-art yöntemlerle karşılaşıldığında sadece daha yüksek doğruluk oranlarına ulaştığıyla kalmayıp daha hızlı olduğu gözlemlenmiştir (Ammour, ve diğerleri, 2017).

ImageNet veri kümesine göre eğitilen CNN tabanlı AlexNet, GoogleNet, Vgg16, Vgg19, ResNet50, Faster-CNN modeller oldukça iyi sonuçlar sergilemesine karşın gerçek zamanlı görüntü işleme uygulamalarında 7 FPS'e yakın değerler vererek gerçek zamanlı görevler için tespit hızı yönünden çok zayıf kaldığını göstermiştir. YOLO modeli, bu çalışmada büyük objelerde %98 başarı sağlarken, küçük objelerde %60 başarı ile zayıf bir performans sergilemiştir (Türk Bilim Araştırma Vakfı, 2020).

İtalya'nın Bari üniversitesinden Carolis ve arkadaşları Google Images Download uygulaması kullanarak toplam 2714 görüntü indirmiştir. İnternet üzerinden resimler rastgele indirildiği için duplicate deletion stratejisini kullanan bir script yazarak duplike resimlerin silinmesini sağlamışlardır. Labelimg uygulaması ile de 1230 çöp poşeti, 1062 çöp konteyniri, 1030 çöp kutusu, 2213 çöp yığını olmak üzere 5535 adet etiketleme yaparak TrashNet adını verdikleri verisetini oluşturmuşlardır. Oluşan verisetini %80 eğitim, %20 validasyon olarak ayırmışlardır. Saturation, exposure ve hue parametrelerini sırasıyla 1.5, 1.5 ve 0.1 seçerek veri artırımı yapmışlardır. YOLOv3 modelinin hiperparametreleri ise batch değeri 64, görüntü boyutu 832x832, momentumu 0.9, weight decay değerini 0.0005, anchor box miktarını da 9 seçilerek eğitim yapılmıştır. Learning rate hiperparametresi ise iterasyonlar sırasında kayıplarda salınım yaşanmaya başlandığında dinamik olarak değiştirilmiştir. Dinamik öğrenme oranı sayesinde model, 10000 iterasyona kadar çalıştırılmış olup en iyi başarının da 10000. iterasyonda sağlandığını belirtmişlerdir. Bu nedenle 4700. iterasyonda kullanılan ağırlıkları kullanmışlardır. Modelin çöp yığınlarını tespit ederken üst üste gelen (overlap) sınırlayıcı kutuların birleştirilmesi için post-detection processing adımı ekleyerek sınırlayıcı kutuların birleşmesini sağlamışlardır. Günün sonunda 10000. iterasyonda elde edilen ağırlıklar kullanılarak validasyon kümesinde %59.6 mAP50 skoru elde etmişlerdir (Carolis, Ladogana, & Macchiarulo, 2020).

Rahmi ve arkadaşlarının yaptığı çalışmada dünya ekonomisi ve iklim dengesi açısından önem arz eden geri dönüştürülebilir çöplerin sınıflandırılmasına odaklanılmıştır. Toplamda 2527 görüntü içeren kağıt, cam, plastik, metal ve karton sınıflarını kapsayan Trashnet veri kümesi üzerinde gerçekleştirilen testlerde DenseNet121, DenseNet169, INceptionResnetV2,

MobileNet ve Xception gibi öncü derin modelleri kullanılırken optimizasyon için ise Adam ve Adadelta tercih edilmiştir. Modeller, 100 ve 150 epokta eğitilmiş. Yapılan deneyler sonucunda Adam'ın Adadelta'ya göre daha etkili olduğu tespit edilmiş ve sınıflandırma doğruluğunu artırmak amacıyla görüntüyü dikey çevirme, 20 derece döndürme gibi veri artırma yöntemleri uygulanmıştır. Fine-tuning ile elde edilen en iyi sonuçlar, sırasıyla %95 ve %94 test doğruluğu oranlarına sahip olan DenseNet121 ve InceptionResNetV2 modelleri tarafından elde edilmiştir. Çalışmanın sonucunda algoritmaların potansiyeli ortaya konurken, daha başarılı sonuçlar için veri setinin boyutunun artırılması ve modellerde çeşitli iyileştirmelerin yapılmasıının önemi vurgulanmıştır (Aral, Keskin, Kaya, & Hacıömeroğlu, 2018).

Sakarya Üniversitesi'nin finansörlük desteği alarak Ergin ve arkadaşlarının atık tespiti yaptıkları çalışmada atıkların sadece türü değil konumu da bir 3D derinlik kamerası olan Intel RealSense D415 kullanılarak görüntü işleme teknikleri aracılığıyla belirlenmiştir. Araştırmada kullanmak için kağıt, metal, cam ve plastik sınıflarından toplam 1974 görüntünden %80-%20 oranlarıyla train-test kümelerinden veriseti oluşturulmuş. YOLOv4 algoritması, gerçek zamanlı nesne tespit hızı bakımından diğer algoritmala baskınlık kurduğu için çalışmada YOLOv4 ve YOLOv4-tiny algoritmaları ele alınmıştır. Pyrealsense2 ve OpenCV kütüphaneleri ile Python ortamında Nvidia Jetson AGX Xavier card ile eğitilerek model oluşturulmuştur. YOLOv4-Tiny ve YOLOv4 sırasıyla %74.56 ve %78.36 IoU ile %81.56 ve %90.23 mAP başarısı elde edilmiştir (Erin, Bingöl, & Boru, 2022).

BÖLÜM 2.

ÇEVRE ve ATIKLAR

Bu çalışma, sosyal sorumluluk ve çevre bilinci kavramlarını teknolojiyle birleştirerek yenilikçi çözümler sunmayı hedeflemektedir.

2.1. Çevre ve Ekosistem

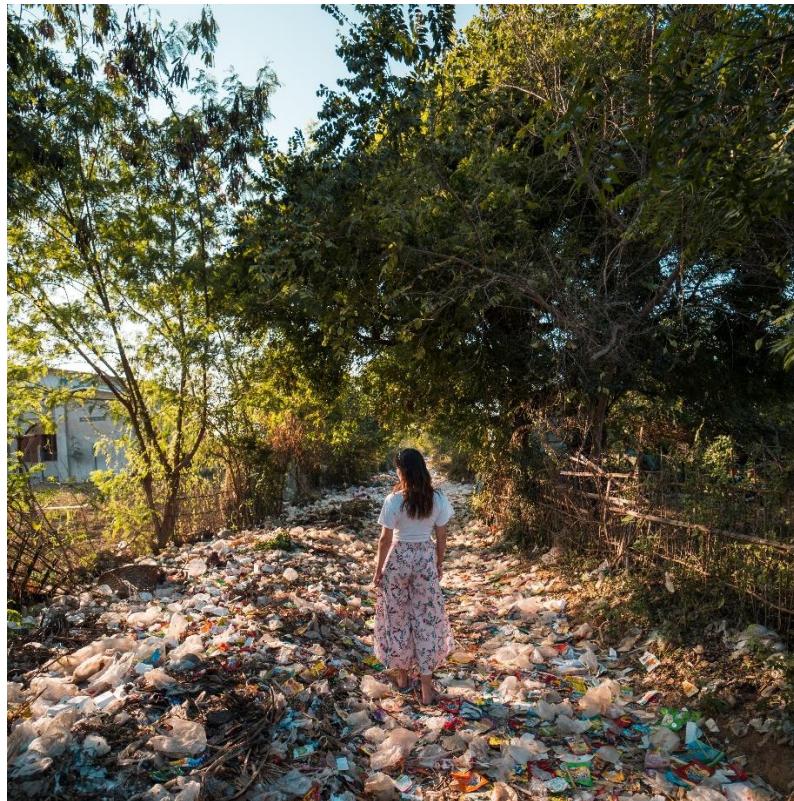
Doğa, canlı-cansız her varlığı kapsar. Doğal çevre, insanlar, hayvanlar ve bitkiler dahil tüm canlıların biyolojik, fizikal ve kimyasal olarak etkileşimde bulunduğu habitatları kapsar. Bu habitatlar, yaşamın devam etmesi için temel ihtiyaçları karşılamakla kalmaz, aynı zamanda uygun iklim ve atmosfer şartlarını temin eder. İnsanlar da dahil tüm canlılar, kendileriyle ve cansız varlıklarla etkileşim içerisinde oldukları bir ekosistemin unsurlarıdır ve kendi başına var olma kapasitesine sahip değildir. Ekosistem ise canlı ve cansız varlıkların belli bir ortamda kompleks bir ahenk içinde bulunduğu sistemdir. İnsanlar, hayvanlar, bitkiler gibi canlılar ekosistemin canlı bileşenlerini oluştururken; su, hava, toprak gibi unsurlar ise cansız bileşenlerini oluşturur. Ekosistem, canlı çeşitliliği ve enerji döngüsü gibi faktörlere bağlı olarak kendi içerisinde yeterli bir sistem oluşturur (Atık Yönetimi Vize Ders Notları, Tarih yok). Şekil 1'de insan ve doğa etkileşimi gösteren görsel paylaşılmıştır.



Şekil 1: İnsan ve doğa

Günümüzde sanayileşme, kentleşme, ve sürekli artmaya devam ederek 7 milyarı aşan dünya nüfusu yüzünden tüketimin ciddi derecede fazlalaşması çevre sorunlarına felaket şekilde neden olmaya başlamıştır. Aynı zamanda insanların yaşam stili, sanayi faaliyetleri, tarım uygulamaları ve enerji üretimi gibi etkinlikler sonucu ortaya çıkan atıklar ekosistemlere zarar vererek ekolojik sorunların meydana gelmesinde büyük bir rol oynamaktadır. Öyle ki, Türkiye

İstatistik Kurumu'nun (TÜİK) yaptığı bir araştırmaya göre 2022 yılında Türkiye'de işyerlerinden ve evlerden toplam 109,2 milyon ton atık ortaya çıkmıştır; bu miktarın 29,4 milyon tonu tehlikeli niteliktedir (Atık İstatistikleri, 2022, 2023). TÜİK'e göre 2022 yılında Türkiye'nin nüfusunun yaklaşık 85,2 milyon olduğunu düşündüğümüzde insanlar 3,511 kg/gün atık üretmektedirler (Adrese Dayalı Nüfus Kayıt Sistemi Sonuçları, 2022, 2023). Şekil 2'de etrafi çöplerle çevrili doğa içinde insan bulunmaktadır.



Şekil 2: Çöple dolu doğa ve insan resmi

2.1. Çevresel Atıklar

Atık, bir sürecin sonucunda ortaya çıkan, kullanım durumunu yitirmiş, doğrudan veya dolaylı olarak doğaya salındığında çevre sağlığına sorun teşkil eden artık maddeler olarak nitelendirilir. Şekil 3'deki gibi kaynağına göre ise farklı atık türleri bulunmaktadır (Orak, Tarih yok):

- Evsel atıklar,
- Tıbbi atıklar,
- Endüstriyel atıklar,
- Tarım ve Hayvancılık atıklar,
- İnşaat atıkları,
- Radyoaktif Atıklar



Şekil 3: İnşaat, organik, batarya, pet, tıbbi, raddrasyon atıkları (Orak, Tarih yok)

Bazı atıklar doğaya ve çevreye zararsızken bazıları oldukça tehlikeli olabilir. Doğaya kontrolsüz şekilde salınan atıklar en sonunda ekosistemin bozulmasına ve insanoğlu da dahil diğer canlıların sağlığının olumsuz yönde etkilenmesine neden olur. İklim bozulması, kuraklaşma, arazi bozulmaları, hava kirliliği, ozon tabakasının delinmesi, içilebilir suların azalması, biyolojik çeşitlenmede azalma, bazı türlerin neslinin tükenmesi gibi problemlerin son zamanlarda global ölçekte devasa boyutlara ulaşmıştır. Bu aşamada, çevresel zorluklarla başa çıkarken sürdürülebilir tüketim alışkanlıklarını ve atık yönetiminin ne denli hayatı önem taşıdığını fark edebiliriz (Atık Yönetimi Vize Ders Notları, Tarih yok). Şekil 4'te farklı tür atıklar için renklendirilmiş çöp kutuları gösterilmektedir.



Şekil 4: Farklı tür atıkların geri dönüşüm kutuları

İnsanlar, çevreyi koruma hassasiyetiyle enerji tasarrufuna yönelme, geri kazanma teşvik etme ve doğal kaynakları koruma konusunda daha bilinçli adımlar atarak çevresel zorluklara karşı

çözümler üretebilirler. Özellikle sanayi devriminden sonra yaşanan teknolojik ilerlemeler insanların doğa üzerindeki hakimiyetini arttırmıştır. Teknolojiyi etkili kullanarak çevresel sorunların önüne geçmek mümkündür. Şekil 5'de sosyal çevre bilinci için görsel paylaşılmıştır.



Şekil 5: Sosyal çevre bilinci

2.1. Atık Yönetimi

Atık yönetimi süreçleri 3 ana aşamaya ayrılmıştır. İlk aşamada atıklar toplanır, toplanan atıklar taşınır, taşınan atıklar depolanır. İkinci aşamada atıklar işlenir. İşleme süreci de 3'e ayrılır: Tekrar kullanım, geri dönüşüm ve geri kazanım. Tekrar kullanımda, atıklar temizlik dışında hiçbir işleme tabii tutulmaz ve aynı veya farklı bir amaç uğruna tekrar kullanıma hazır hale getirilir. Geri dönüşüm, toplanan atıkların işlenerek yeni ürünlere dönüştürülmesine denir. Örneğin toplanan kauçuk, metal, kağıt, cam veya plastik atıkların geri dönüşümüyle yeni kauçuk, metal, kağıt, cam veya plastik ürünlerinin oluşturulmasıdır. Geri kazanımda ise yağlar, pil, akü, ve diğer elektronik atıklardan enerji elde edilir. Üçüncü aşamada ise bertaraf işlemi gerçekleştirilir. Bertaraf aşamasında, ikinci aşamada uygulanan yöntemler sonunda arta kalan maddelerin kompostlama, düzenli/düzensiz depolama yöntemleriyle sağlığa olumsuz etkisi giderilir (Atık Yönetimi Vize Ders Notları, Tarih yok).

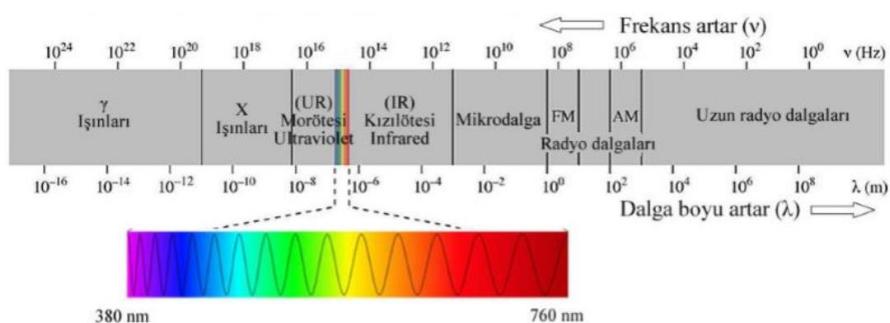
Bu bölümde anlatılanlara özet olarak, dünya ekosisteminin değeri ve yaşanabilir bir dünya için olan kritik rolü, bugünün dünyasında göz ardı edilemez bir gerçekdir. Bu noktada atık malzemelerin geri dönüşümünün tespiti, insanlığın ve medeniyetin sürdürülebilmesi için büyük bir öneme sahiptir. Bu çalışma kapsamında, cam, plastik, alüminyum ve karton gibi materyallerin geri dönüşümünün sınıflandırılması hedeflenmektedir.

BÖLÜM 3.

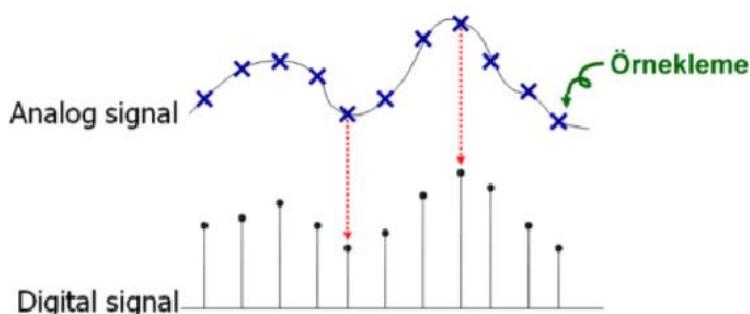
BİLGİSAYARLI GÖRÜ TEKNOLOJİSİ

3.1. Dijital Görüntü

İşinler farklı dalga boylarında insanların çiplak gözle göremeyeceği sinüs şeklinde yayılan elektromanyetik dalgalarıdır. Kütlesi olmayan ve ışık hızında ilerleyen bu dalgalar enerji yığını (photon) taşımaktadır. Photonlar, taşıdıkları enerji miktarına göre gruplanmıştır. Şekil 6'de elektromanyetik spektruma yer verilmiştir.



Dünya'ya gelen Güneş ışınları kamera içerisindeki filmde kimyasal tepkimelerle neden olarak kamera gibi donanımlar aracılığıyla analog sinyal elde edilir. Analog sinyaller, sürekli fonksiyon olarak temsil edilebilmektedir. Analog-Dijital dönüştürücüler sayesinde analog sinyallerin belirli zaman aralıklarındaki örneklemleri alınarak Şekil 7'deki gibi 0 veya 1 değerini alan kesikli yapıdaki dijital sinyaller oluşturulur (Kurban, Tarih yok).



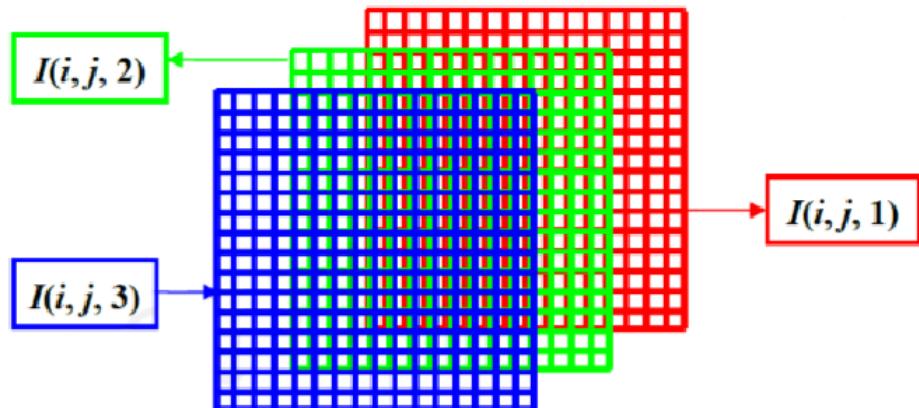
Şekil 7: Analog sinyalin dijital sinyale dönüşümü (Kurban, Tarih yok)

Dijital görüntüler, satır ve sütunlardan oluşan iki boyutlu bir dizi olarak temsil edilebilir, renk uzayına bağlı olarak değişen boyutlarda matris şeklinde saklanabilir. Dijital görüntü dizisindeki her bir elemanına yani temel birimine ise piksel (pixel) denir. Piksellerde renk bantları 0 ile 255

arasında toplam 256 değer almaktadır. Renkli görüntüler için kırmızı-yeşil-mavi (RGB) renkleri olmak üzere 3 kanaldan oluşmaktadır. Öyle ki, RGB formatındaki renkli görüntülerin piksellerinde $256 \times 256 \times 256 = 16.777.216$ çeşit renk değeri alabilmektedir. Denklem (*)’de $X \times Y$ büyüklüğünde ve 2D boyutlu RGB formatındaki sayısal görüntünün $\{I(i, j, k) | i = 1, 2, \dots, X; j = 1, 2, \dots, Y; k = 1, 2, 3\}$ I matrisi için matris temsiline yer verilmiştir. Şekil 8’de ise, Denklem (1)'de belirtilen I matrisinin RGB bantları gösterilmiştir.

Denklem

$$\begin{aligned} \text{Kırmızı (R) bandına ilişkin matris} &= I(i, j, 1); i = 1, 2, \dots, X \text{ ve } j = 1, 2, \dots, Y \\ \text{Yeşil (G) bandına ilişkin matris} &= I(i, j, 2); i = 1, 2, \dots, X \text{ ve } j = 1, 2, \dots, Y \\ \text{Mavi (B) bandına ilişkin matris} &= I(i, j, 3); i = 1, 2, \dots, X \text{ ve } j = 1, 2, \dots, Y \end{aligned} \quad (1)$$



Şekil 8: RGB renk bantları (Kurban, Tarih yok)

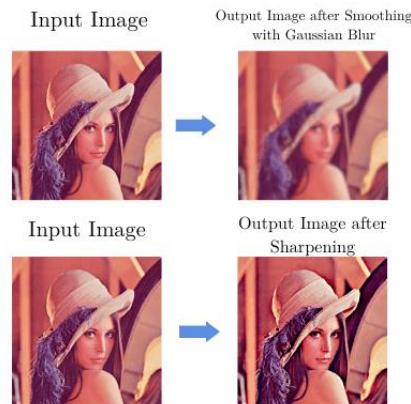
3.2. Görüntü İşleme ve Bilgisayarlı Görü

Bilgi teknolojisinin evriminde, bilgisayarlı görüp göründü İşleme alanları, yapay zeka ve desen tanıma tekniklerini kullanarak görsel girdilerden anlamlı bilgiler elde etmeyi amaçlayan iki yakından ilişkili alandır (Department of Electrical and Computer Engineering, Tarih yok). Dolayısıyla bu iki kavram iç içe geçip karıştırılabilir için birbiri yerine de kullanılabilir.

3.2.1. Görüntü İşleme

Görüntü işleme, endüstri 4.0 olarak adlandırılan teknik devrimde başrolü oynayarak, görsel veri işleme ve nesne tanıma teknolojilerine yeni ufuklar açmaktadır. Bu alandaki algoritmalar,

görsel verileri manipülasyon ederek onarmak, depolayabilmek için sıkıştırmak, insanların daha iyi anlayabilmesi için görüntüleri düzenlemek, sanal ve artırılmış gerçeklikte yeni deneyimler oluşturmak gibi amaçlar doğrultusunda kullanılır (Department of Electrical and Computer Engineering, Tarih yok). Şekil 9'da girdi ve çıktıının görüntü olduğu yumusatma ve keskinleştirme işlemeye örnek verilmiştir.

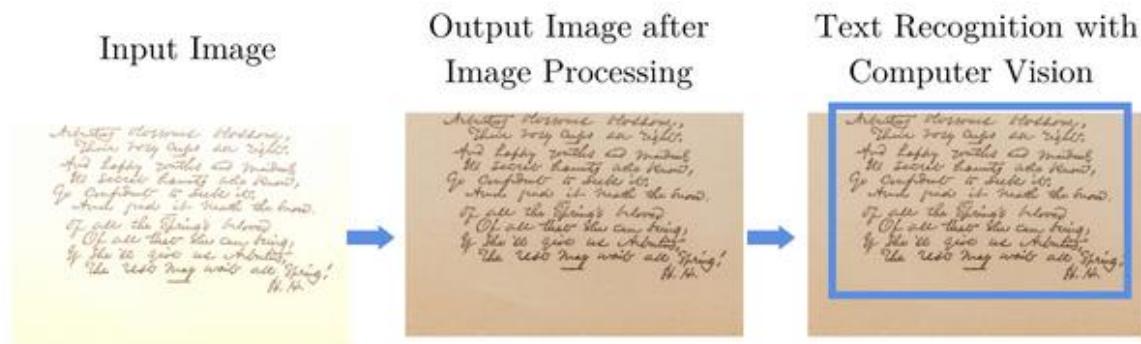


Şekil 9: Girdi görüntüye uygulanan yumusatma ve keskinleştirme işlemleri (Barreto, 2023)

3.2.2. Bilgisayarlı Görü

Bilgisayarlı görü (Computer Vision), dijital formattaki görüntülerin bilgi çıkarımı gibi amaçlar doğrultusunda belirli algoritmalarla bilgisayar aracılığıyla işlenmesidir. Çeşitli operasyonların kullanılmasıyla görüntüden nesne tanıma, el yazısının tanınması gibi girdinin görüntü olup çıktıının ise görüntü içeriğiyle alakalı bilginin elde edilmesidir.

Ayrıca görüntü işleme, bilgisayarlı görünün önemli yapıtaşlarındanandır. Bilgisayarlı görü uygulamalarında algoritmalar çalıştırılmadan önce yaygın biçimde önişleme (preprocessing) adımı olarak görüntü işlemeden faydalанılır (Barreto, 2023). Şekil 10'da kullanıma örnek verilmiştir.



Şekil 10: Girdi görüntüye görüntü işleme ve bilgisayarlı görünün uygulanması (Barreto, 2023)

Bilgisayarlı görü bilimi, mühendislik bakış açısıyla insanın görüş sistemini bilgsayara uygulayarak insan etkileşimini O'a indirmeyi yani diğer bir deyişle otonomlaşmayı amaçlamaktadır (Sonka, Hlavac, & Boyle, 1993).

Bu teknoloji, otomotivden sağlığa, medyadan güvenliğe kadar birçok sektörde kullanılmaktadır. Tıbbi cihazlarla hastalıkların teşhisleri, futbolda oyuncuların ve topun takibi, tarımda gelişmiş kameralara sahip drone'lar aracılığıyla tarım ürünlerinin sağlığını izlemek gerçek hayatı kullanımına örnek sayılır (Department of Electrical and Computer Engineering, Tarih yok).

Günümüzde Türkiye'de arama kurtarma çalışmalarında da bilgisayarlı görüden faydalılmaktadır. Örnek olarak, 6 Şubat 2023 tarihinde aynı gün yaşanan Kahramanmaraş merkezli 7.7 ve Hatay merkezli 7.6 şiddetindeki 2 yıkıcı depremden sonra uydudan alınan deprem öncesi ve deprem sonrası görüntüler yardımıyla yıkılan yapıların analizleri yapıldı. Askeri alanda kullanılan Drone'lar deprem sahasında gözcü rolüyle kullanılarak yıkıntı düzeyi tespit edildi. Duvar arkası radar ile enkaz altında mahsur kalan insanlar tespit edilip ekipler yönlendirildi (Memiş, 2023).

Bilgisayarlı görü disiplininde nesne tanıma için nesne segmentasyonu, nesne sınıflandırma ve nesne tespiti olmak üzere 3 ana başlık bulunmaktadır. Bu çalışmada ise nesne tespiti üzerine uygulama geliştirileceğinden sadece nesne tespiti açıklanacaktır.

BÖLÜM 4.

KONVOLÜSYONEL SİNİR AĞLARI

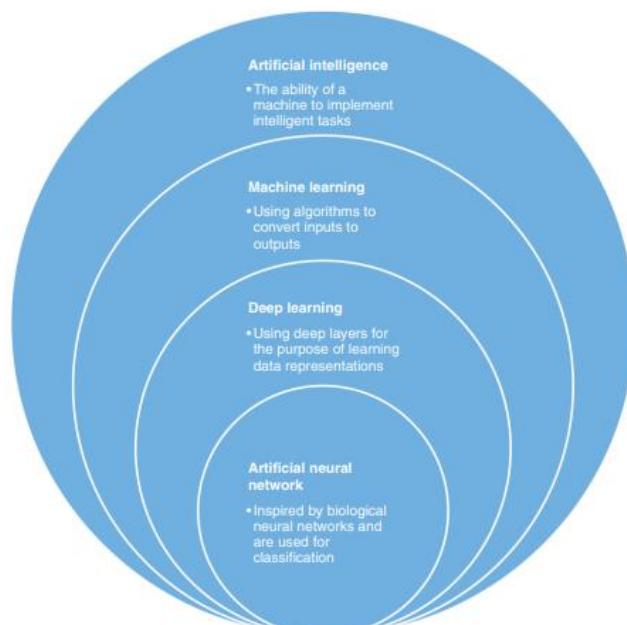
Bu bölümde Makine Öğrenmesi, Yapay Sinir Ağları ve Derin Öğrenme kavramlarına kısaca değinilip ardından sırasıyla CNN'in kısa tarihi, CNN'in ana mimarisi ve son olarak CNN'den türeyen başarısı kanıtlanmış mimarilerden bahsedilecektir.

4.1. Makine Öğrenmesi

Yapay zeka, insanların sahip olduğu mantık yürütübilme yeteneğinin makinelere kazandırılarak makinelerin adeta bir insan gibi davranışmasını ve insan gibi düşünerek karar vermesini sağlamaya çalışan bir disiplindir.

1950 yılında "Makineler Düşünebilir mi?" sorusunu soran Alan Turing, kendisinin geliştirdiği Turing Test'i ile camiada büyük bir yankı uyandırmıştır. Turing Test'i, Yapay Oyun (The Imitation Game) olarak da bilinir ve makinenin insanı kandırabilme yeteneğini ölçer. Eğer bir makine veya bir yazılım insan gibi davranışarak başka bir insanı kandırabiliyorsa test başarılı anlamına gelmektedir (French, 2000).

Şekil 111'de şemada Yapay Zeka, Makine Öğrenmesi, Derin Öğrenme ve Yapay Sinir Ağları kavramlarının hiyerarşisi ve ilişkisi gösterilmektedir.



Şekil 11: Yapay Zeka, Makine Öğrenmesi, Derin Öğrenme ve Yapay Sinir Ağlarına ilişkin şema
(Carpenter, Cohen, Jarrell, & Huang, 2018)

Teknolojinin gelişmesi ile birlikte artan internet ve sosyal medya kullanıcıları sonucunda her gün sayısız miktarda veri meydana gelmektedir. Bu veriler, yapay zekanın bir alt dalı olan makine öğrenmesi alanında kullanılarak anlamlı desenler ve özellikler keşfedilebilir. Makine öğrenmesi disiplini yüksek matematik, istatistik ve programlama bilgisi gerektirmektedir.

Bir makine öğrenmesi modeli, veri kümeleri üzerinden eğitilerek anlamlı desenler ve özellikler keşfeder. Eğitilen modele gelecekte yeni bir girdi verildiğinde daha önce analizini gerçekleştirek anlamlı desenler ve özelliklerin çıkarımını yaptığı veri kümesine dayalı bir tahmin çıktısı ortaya koyar.

Modeller oluşturulurken seçilen veri kümesine ve problemin türüne özel yaklaşımalar dikkate alınmaktadır. Bu başlık altında bazı bilinen problem türleri ve birkaç öğrenme yöntemi ele alınmıştır.

4.1.1. Öğrenme Çeşitleri

Makine öğrenmesi dünyasında tarih boyunca sınıflandırma, regresyon ve kümeleme gibi başka birçok probleme çözüm getirebilmek adına farklı öğrenme yöntemleri geliştirilmiştir. Bu teknikler sayesinde veri setlerindeki neden-sonuç ilişkilerinin anlaşılmaları ve makine öğrenmesi modellerine tahmin etme kapasitesi kazandırılması sağlanmıştır. Bu öğrenme stratejilerinden en popüler olanları denetimli öğrenme ve denetimsiz öğrenmedir.

Denetimli Öğrenme (Supervised Learning)

Denetimli öğrenme tekniğindeki amaç makine öğrenmesi modellerini, etiketlenmiş veriler üzerinden eğitmektir. Veri kümesi, eğitim (train) ve test olmak üzere iki kümeye ayrıılır ve iki kümeye de etiketlenmiş verilerden oluşmaktadır. Model, eğitim sürecinde train kümesindeki öznitelik değişkenleri ile hedef (target) değişkenlerin arasındaki ilişkiyi öğrenir. Modelin başarısı train ve test kümesine göre verdiği tahminler üzerinde çeşitli metrikler aracılığıyla hesaplanır. Bu öğrenme yöntemi regresyon ve sınıflandırma problemlerinde kullanılmaktadır. Regresyon, sürekli (continuous) çıktıyı tahmin etme problemidir. Sınıflandırma ise ayrık (discrete) kategorik çıktıların tahmin edilmesi problemidir. Aslında sınıflandırma da bir regresyon çeşididir; çıktı (response) eğer kategorikse bu tip regresyon, lojistik regresyon olarak adlandırılır ve regresyonla elde edilen değer kategorik değere yuvarlanır.

Denetimli öğrenme metodunu kullanılarak literatürde birçok alanda çalışmalar yürütülmüştür ve yürütülmeye de devam edilmektedir. Bu yöntemin başlıca kullanıldığı alanlar aşağıdaki maddelerde belirtilmiştir (Das, Dey, Pal, & Roy, 2015):

- E-mail spam filtreleme,
- El yazısı tanıma,
- Yüz tanıma,

- Konuşma tanıma,
- Doğal dil işleme,
- Anomali tespiti,
- Saldırı tespit sistemi,
- Epilepsi nöbet sistemi.

Denetimsiz Öğrenme (Unsupervised Learning)

Denetimsiz öğrenme stratejisinde modeller bir rehber yardımcı olmadan etiketsiz veriler üzerinde eğitilir. Hedef bilgisi, bu öğrenmede kullanılan veri setinde bulunmaz. Verinin kendisinden anlam çıkarılır. Benzer özelliklere sahip veriler gruplandırılır. Kümeleme (clustering) problemlerinde denetimsiz öğrenme tercih edilmektedir.

Denetimsiz öğrenme metodunun başlıca kullanıldığı alanlar aşağıdaki maddelerde belirtilmiştir (Das, Dey, Pal, & Roy, 2015):

- DNA sınıflandırma,
- Sosyal medya analizi,
- Piyasa bölünmesi (Market Segmentation)
- Astronomik veri analizi,
- Kanser teşhisi,
- Konuşma etki tespiti.

4.1.2. Modelin Öğrenmesi

Makine öğrenmesi, verilerden öğrenme yoluyla modele karar verme yeteneği kazandırmayı amaçlayan bir alandır. Modeller bir veya daha fazla parametrelerden oluşmaktadır. Modelin başarısını bu parametreler belirler ve bu parametrelerin düzgün ayarlanması modelin performansını oldukça yükseltir. Modelin öğrenmesinde en önemli etkenlerden ikisi kayıp fonksiyonları ve optimizasyondur.

Kayıp Fonksiyonu (Loss Function)

Kayıp fonksiyonu, modelin tahmin ettiği değerler ile gerçekde beklenen değerler arasındaki hata farkının hesaplayarak uygun optimizasyon yöntemi ile modelin iyileştirilmesini sağlar. Maliyet (Cost Function) olarak da bilinir. Bilinen kayıp fonksiyonlarından biri olan ortalama hata karesi (MSE), Denklem (2)'de gösterilmiştir.

$$J(\theta) = MSE = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 \quad (2)$$

Denklemde \hat{y}_i modelin tahmin değerini, y_i gerçek değeri, N gözlem miktarını ifade etmektedir.

Optimizasyon Fonksiyonu (Optimization Function)

Optimizasyon, bir fonksiyonun parametrelerini en iyilemeyi sağlayan bir süreçtir. Makine öğrenmesinde temel optimizasyon yöntemleri, birinci dereceden optimizasyon yöntemleri (First-Order Methods), yüksek dereceden optimizasyon yöntemleri (High-Order Methods) ve türevsiz optimizasyon yöntemleri (Derivative-Free optimization methods) olarak üç gruba ayrılabilir. Birinci dereceden alınan türev metotları optimizasyon süreçlerinde ise en yaygın kullanılan metotlardandır (Sun, Cao, Zhu, & Zhao, 2019).

Dereceli azalma (gradient descent) birinci dereceden türeve dayanan, minimizasyon problemlerinde kullanılan bir optimizasyon algoritmasıdır. Literatürde bulunan en popüler dereceli azalma çeşitlerinden 2 tanesi şunlardır:

- Toplu Dereceli Azalma (Batch Gradient Descent),
- Rastgele Dereceli Azalma (Stochastic Gradient Descent).

Batch gradient descent yönteminin fikri, değişkenlerin kayıp fonksiyonunun gradyanlarının (tersi) yönünde iteratif olarak güncellenmesidir. Güncelleme, amaç fonksiyonunun optimal değerine adım adım yavaşça yaklaşmak için yapılır (Sun, Cao, Zhu, & Zhao, 2019).

Diyelim ki η öğrenme oranımız, $J(\theta)$ kayıp fonksiyonumuz ve iyileştirilecek θ parametreli $f_\theta(x)$ fonksiyonumuz olsun. Fonksiyonlar sırasıyla Denklem (3) ile Denklem (4)'de verilmiştir.

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^N (y^i - f_\theta(x^i))^2 \quad (3)$$

$$f_\theta(x) = \sum_{j=1}^D \theta_j x_j \quad (4)$$

Burada N eğitimdeki gözlem sayısını, D girdi öznitelikleri, $i = 1, \dots, N$ için $x^i = (x_1^i, \dots, x_D^i)$ olmak üzere x^i bağımsız değişkeni ve y^i hedef çıktıyi temsil etmektedir. Dereceli azalma yöntemi Denklem (5) ve Denklem (6)'da verilen 2 adımı yerel optimuma yakınsayana kadar devam etmektedir (Sun, Cao, Zhu, & Zhao, 2019).

İlk adımda $J(\theta)$ fonksiyonun θ_j 'e göre 1.dereceden kısmi türevi alınır:

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{N} \sum_{i=1}^N (y^i - f_\theta(x^i)) x_j^i \quad (5)$$

İkinci adımda yeni değer $\hat{\theta}_j$ olmak üzere her bir θ_j parametresi, gradyenin negatif yönünde maliyet fonksiyonunun minimize edilmesi için güncellenir:

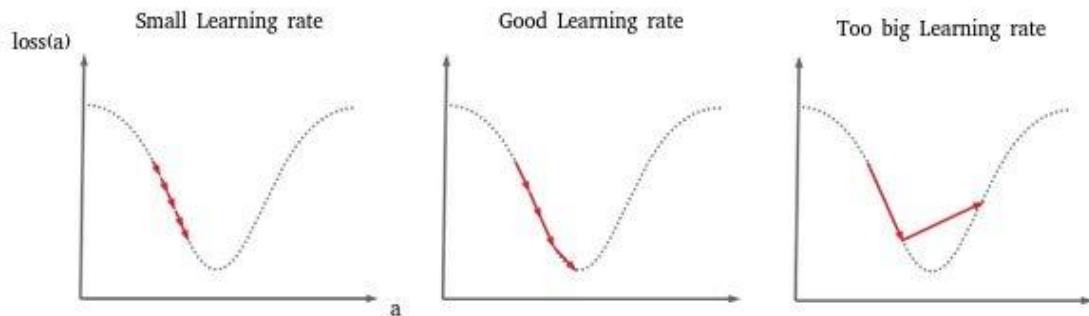
$$\hat{\theta}_j = \theta_j + \eta \cdot \frac{1}{N} \sum_{i=1}^N (y^i - f_\theta(x^i)) x_j^i \quad (6)$$

Batch Gradient Descent'nin çalışma karmaşıklığı her iterasyon için $O(D.N)$ 'dir. Hacimli verilerle uğraşırken yüksek matematiksel hesaplamalar gerektirmesi yüzünden çalışma karmaşıklığı yine her iterasyon için $O(D)$ olan Stochastic Gradient Descent (SGD)'in doğmasına sebep olmuştur. SGD, veri kümesindeki örnekleri rastgele seçer. Bir önceki yönteme göre daha hızlı olsa da gürültü yaratabilmektedir. SGD, Denklem (7)'de verilmiştir (Sun, Cao, Zhu, & Zhao, 2019).

$$\hat{\theta} = \theta + \eta \cdot (y^i - f_\theta(x^i)) x^i \quad (7)$$

Bu başlık altında dikkat edilmesi gereken en önemli hususlardan bir tanesi, öğrenme katsayısı η dediğimiz hiperparametrenin uygun seçilmesidir. η 'nin küçük seçilmesi lokal minimuma küçük adımlarla gidilmesine ve böylece modelin çok yavaş öğrenmesine neden olur. Tam tersi, öğrenme katsayısının yüksek seçilmesi ise genellikle lokal minimumdan sapmaya neden olur (Ruder, 2017).

Şekil 12'de öğrenme oranına göre lokal minimuma yaklaşma biçimleri gösterilmektedir. Optimum öğrenme oranında lokal minimuma başarıyla yaklaşığı, düşük öğrenme oranında adımların çok yavaş ilerlendiği, yüksek öğrenme oranında ise lokal minimumdan sapabileceği anlatılmaktadır.

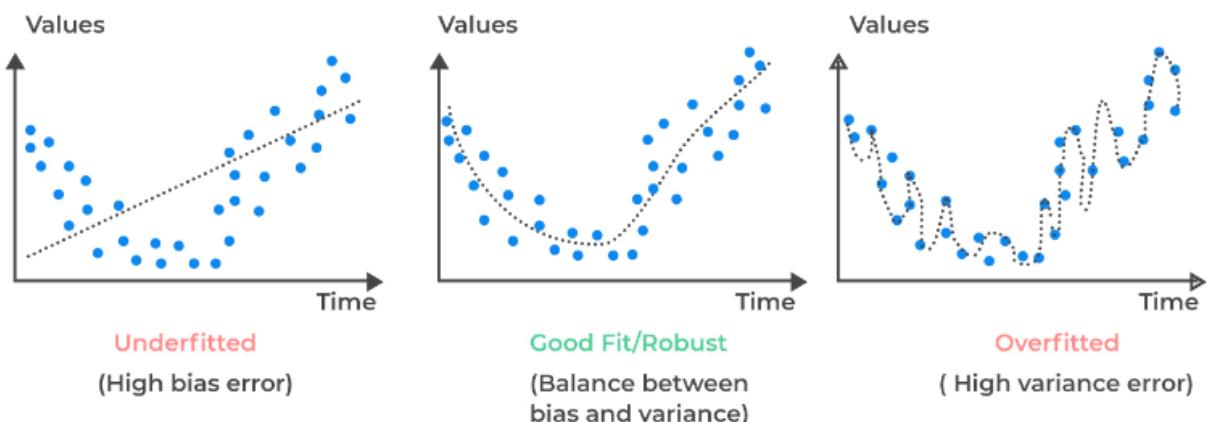


Şekil 12: Küçük, uygun ve aşırı büyük öğrenme oranlarının seçilmesi (Cayla, 2021)

Yapay sinir ağlarında gradient descent yöntemleri kullanılarak ileri yönlü yayılım işleminin sonucunda elde edilen kayıp fonksiyonunun 1. dereceden türevinin alınıp rastgele adımlarla lokal minimuma yakınsaması sağlanır. Ağırlıklar bu işlem sayesinde güncellenmektedir (Ruder, 2017).

4.1.3. Yetersiz Öğrenme (Underfitting) ve Aşırı Öğrenme (Overfitting)

Modelin düzgün öğrenebilmesi için hem düşük varyans hem de düşük taraflılığa sahip olması gerekmektedir. Genel olarak makine öğrenmesinde modeller bazen yetersiz bazen de aşırı öğrenebilmektedir. Yetersiz öğrenme yüksek yanılık (bias) sonucunda gerçekleşir. Aşırı öğrenme ise yüksek varyans sebebiyle yaşanır. Bahsedilen iki durum da modelin başarısız performans göstermesine neden olur. Örneğin, aşırı öğrenmiş bir makine öğrenmesi modeli gürültülerin etkisiyle barındırmasından ötürü daha önce gözlemlenmemiş bir veri için çıktıyı yanlış tahmin etme eğilimindedir. Aşırı öğrenme durumu, eğitim setinin doğruluğunun test setinin doğruluğundan bir hayli yüksek çıkmasıyla farkedilebilir. Şekil 13'te regresyon problemini içeren bir örnek üzerinden underfitting, overfitting ve düzgün öğrenme farkı gösterilmiştir.

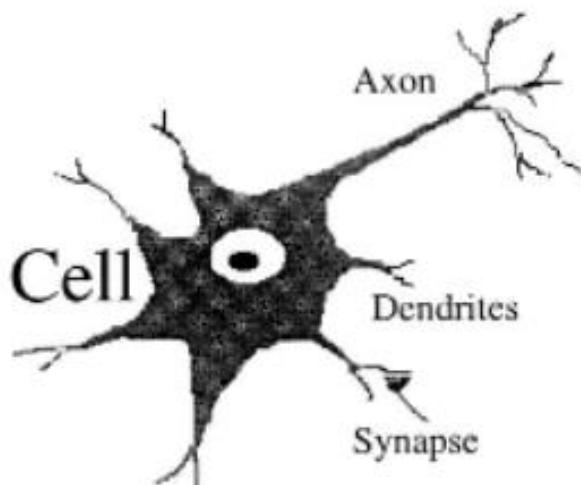


Şekil 13: Sırayla Underfitting, Düzgün Öğrenme ve Overfitting karşılaştırması (Forjan, 2021)

4.1.4. Yapay Sinir Ağları

Yapay Sinir Ağları (ANN: Artificial Neural Network), gelişmiş varlıklar olan insanlar ve hayvanların bilgi işleme şeklini taklit etmek amacıyla sinir sisteminin önemli parçası olan beyin organının biyolojik yapısından esinlenerek matematiksel modellenmesi ile ortaya çıkan bir bilgisayar kavramıdır. Tıpkı insanlarda olduğu gibi deneyim yoluyla öğrenirler. ANN'leri iyi anlamak için ilk olarak beyindeki işleyiş hakkında bir bilgi sahibi olmak gerekmektedir.

Beyin, örüntü tanıma, tahmin gibi konularda son derece etkileyici bir araçtır ve sonsuz çeşitlilikteki girdi örüntülerini öğrenip tanayıbilir. Beyinde ortalama olarak 100 milyar tane nöron adı verilen sinir hücreleri bulunmaktadır. Her bir nöronun da diğer nöronlarla sayısı yaklaşık olarak 1000 ile 10000 arasında değişen bağlantılarla birbirlerine bağlı olduğu düşünülmektedir (Agatonovic-Kustrin & Beresford, 2000). Bilgi ve deneyimin tümü, nöronların aralarındaki bağlantılar tarafından kodlandığına inanılır, ancak bu konu hakkında hala tartışmalar sürmektedir (Noriega, 2005). Bir biyolojik nöronun yapısı Şekil 14'deki modelde gösterilmiştir.



Şekil 14: İnsan sinir hücresinin yapısı (Agatonovic-Kustrin & Beresford, 2000)

Beynin temel yapı taşıları olan nöronlar, hücre aktivitesini kontrol eden çekirdek içeren bir hücre gövdesi, bilgiyi hücreye taşıyan birçok ince tel olan dendritler ve sinyali uzaklaştırın daha uzun bir tel olan aksondan oluşmaktadır. Bir nöron diğer nöronlarla sinaps adı verilen bir bağlantı noktasında birleşir. Rezeptörlerden herhangi bir uyarıma sonucunda oluşan elektrokimyasal sinyaller (impuls), bir sinir hücresine dentritlerden girer. Dentritlerden gelen tüm sinyaller hücre gövdesinde toplanır. Eylem potansiyeli akson boyunca sinapslara ulaşılır. Hücrede gerçekleştirilen eşik kontrolü sonucunda bu sinyaller diğer nöronlara ya tamamen iletilebilir ya da hiçbir şekilde iletilemez.

Nöronlar tamamen bağlı (fully-connected) bir ağda düzenlenir ve uyarıları alıp göndererek adeta bir mesajcı gibi davranışlarılar. Sinir ağlarının bağlantısının sonucunda öğrenme, tahmin ve tanıma gibi yeteneği olan zeki bir beyin ortaya çıkarır (Agatonovic-Kustrin & Beresford, 2000).

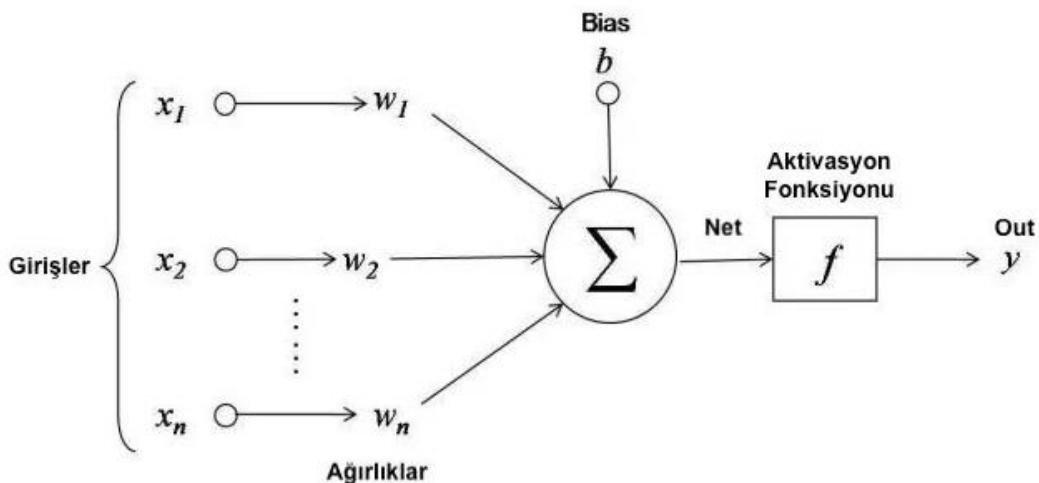
Tek Katmanlı Algılayıcı (Perceptron)

Yapay sinir ağlarının temel yapıtaşısı yapay nöron hücreleridir. 1943 yılında nörobilimci Warren S. McCulloch ve mantık bilimci Walter Pitts, birçok sinir hücresinin bir araya bağlanmasıyla beynin nasıl karmaşık desenler üretebileceğini anlamaya çalışmışlar ve sundukları MCP nöron modeli ile yapay sinir ağlarının gelişimine önemli katkıda bulunmuşlardır (Marsalli). Daha sonra 1957 yılında Frank Rosenblatt tarafından yapay nöron modeli olarak bilinen tek katmanlı Perceptron tanıtılmıştır (Rosenblatt, 2017). Perceptron, insan ve hayvan beynindeki gerçek nöronlara benzer şekilde girdi sinyallerini alır, bu sinyalleri çarpar ve toplar, ardından da aktivasyon işlemlerine tabi tutar. Tek katmanlı perceptron yapısı Şekil 15'de gösterilmiştir.

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

Şekil 15: Biyolojik Sinir Ağları ve Yapay Sinir Ağları bileşenlerinin denklikleri (Artificial Neural Network Tutorial)

Yapay nöron bileşenlerinin biyolojik nörondaki hangi bileşenlere karşılık geldiği de Şekil 16'de gösterilmiştir.



Şekil 16: Tek Katmanlı Algılayıcı modeli (Güzel, 2018)

Algılayıcıya giren x_1, \dots, x_n girdi değerleri olsun, her bir girdi değerine denk düşen w_1, \dots, w_n ağırlık bulunmaktadır. Girdiler ile ağırlıkların çarpımına skalar bir değer olan yanlışlık b değerinin de eklenmesiyle Denklem (8)'de verilen net değer z oluşmaktadır.

$$z = \sum_{i=1}^n w_i * x_i + b \quad (8)$$

Denklem (*Error! Reference source not found.*)'den elde edilen z değeri bir aktivasyon fonksiyonuna sokulur ve çıkan sonuca göre nöronun ateşlenip ateşlenmeyeceğinin kararı verilir. Aktivasyon fonksiyonlarından biri olan adım fonksiyonu z değerini girdi alacak şekilde Denklem (9)'de ifade edilmektedir. Aktivasyon fonksiyonun sonucu olan y değeri 0 ise nöron ateşlenmez, 1 ise nöron ateşlenir.

$$y = f(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases} \quad (9)$$

Perception'ın öğrenme sürecinde yaygın olarak delta kuralı kullanılır. Delta kuralında, gerçekleşen değer ve beklenen değerin hata kareler yöntemine göre türevinden faydalananlarak ağırlıklar güncellenir. Ağırlıklar, hatanın türevine göre hatayı minimize etmeye çalışan gradient descent yöntemine göre iyileştirilir. Modele verilen girdilere göre, modelin tahmin ettiği değer y_{tahmin} , gerçek değer $y_{gerçek}$ ve modele tanımlanan öğrenme oranı α olsun. Denklem (10)'de delta kuralı tanıtılmıştır (Noriega, 2005).

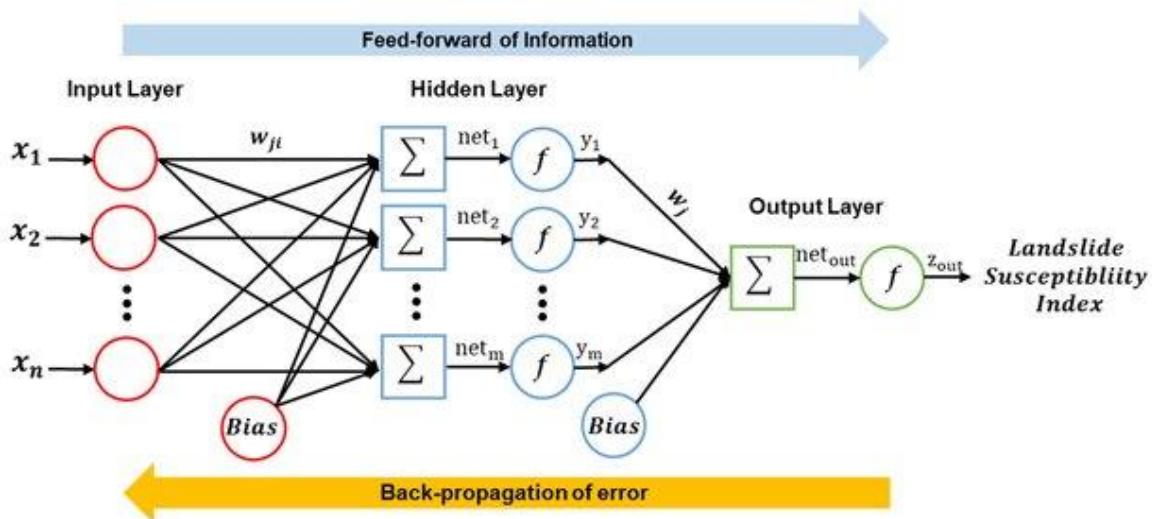
$$\Delta w_i = 2 * \alpha * x_i * (y_{gerçek} - y_{tahmin}) \quad (10)$$

Model, Denklem (*Error! Reference source not found.*), Denklem (*Error! Reference source not found.*) ve Denklem (*Error! Reference source not found.*) iterasyonlarca tekrar edilerek hatanın minimize edilmesi sağlanana kadar eğitilir.

Tek katmanlı perceptronlar ile doğrusal ayrılabilir (linearly separable) olan mantıksal OR kapısı ve mantıksal AND kapısı gibi sınıflandırma problemleri başarıyla çözülebilmiştir. Fakat bu tür bir sistem XOR gibi doğrusal ayrılabilir olmayan sınıflara çözüm bulmakta sınıfta kalmaktadır (Noriega, 2005).

Çok Katmanlı Algılayıcı (Multi Layer Perceptron)

Çok katmanlı algılayıcı birden fazla algılayıcının bir araya gelmesiyle oluşan yapay sinir ağları modelidir. Tek katmanlı algılayıcıda girdi katmanı ve çıktı katmanı bulunurken bu sistemde girdi katmanı, gizli katman ve çıktı katmanı olmak üzere 3 katman bulunmaktadır. Gizli katman, bir veya daha fazla katman içerebilmektedir. Lineer olmayan problemlere tahmin üretebilmek amacıyla tanıtılmıştır. Tek katmanlı algılayıcıyla çözülemeyen doğrusal olmayan XOR kapısı problemi, çok katmanlı algılayıcı modeli ile çözülebilmiştir (Noriega, 2005). Şekil 17'de modelin yapısı gösterilmektedir. Ayrıca bu modelde aynı katmanlarda bulunan düğümler arasında bir bağlantı bulunmamaktadır.



Şekil 17: Çok Katmanlı Algılayıcı modeli (Lee, Kim, & Lee, 2020)

Girdi katmanından dış dünyadan alınan veriler sisteme girer. Girdi değerler, sırasıyla gizli katman ve çıktı katmanlarında bulunan algılayıcılarla çarpım ile toplama operasyonlarına dahil olarak net değerler hesaplanır. Hesaplanan net değerler üzerinde de lineer veya lineer olmayan aktivasyon fonksiyonları uygulanarak çıktı değer(ler) üretilmiş olur. Bu akışa ileri yönlü yayılım adı verilir.

$$y_i = f(\text{net}_j) = f\left(\sum_{i=1}^n w_{ji}x_i + b_i\right) \quad (11)$$

$$z_{out} = f(\text{net}_{out}) = f\left(\sum_{j=1}^n w_jx_j + b_j\right) \quad (12)$$

Denklem **(11)** ve Denklem **(12)**'de hesaplamaların yapıldığı fonksiyonlar verilmiştir. Burada x_i girdi değerini, y_i ve z_{out} sırayla gizli ve çıktı katmanından çıktı değerlerini, w_{ij} ve w_j sinaptik ağırlıkları, b_i ve \dot{b}_j yanlılık değerlerini, n ve m değerleri sırayla girdi katmanındaki ve gizli katmandaki nöron sayılarını, ve f operatörü ise aktivasyon fonksiyonunu temsil etmektedirler (Lee, Kim, & Lee, 2020).

Modelin öğrenme süreci 2 aşamadan oluşur:

- İleri yönlü yayılım (Forward Propagation),
- Geri yönlü yayılım (Backward Propagation).

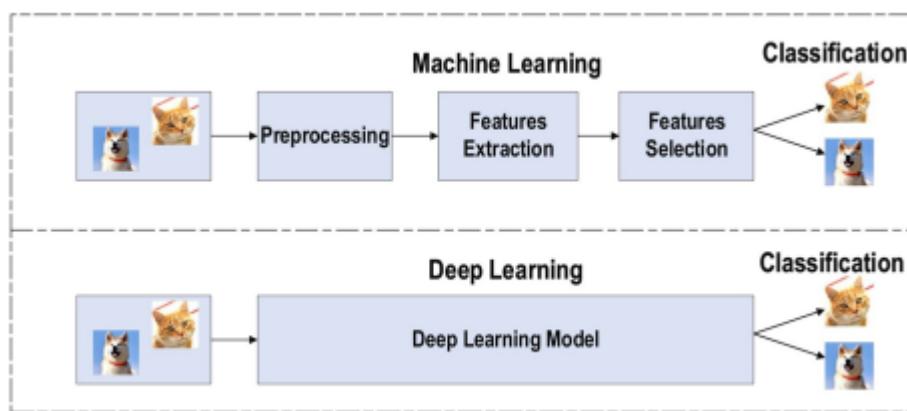
Süreçte ileri yönlü yayılım akışıyla girdi katmanından başlanılarak çıktı katmanına gidilir ve akış sonra erer. Sonrasında geri yönlü yayılım başlar. Bu algoritma, başlangıçta rastgele atanan ağırlıkları (Dongare, Kharde, & Kachare, 2012) ağın tahmin ettiği değerler ile gerçek değerlerinin kullanılmasıyla hesaplanan MSE kayıp fonksiyonu üzerinden gradient descent yöntemi ile güncellenir. Ağırlıkların güncellenmesi işlemi, MSE hatası belli bir eşeğe ulaşana devam eder (Lee, Kim, & Lee, 2020).

Bishop'a (1995) göre geriye doğru yönelim algoritması adımları aşağıdaki gibidir (Gardner & Dorling, 1998):

1. Ağ ağırlıkları başlatılır (initialization),
2. Eğitim verilerinden giriş vektörü ağa girdi olarak sağlanır,
3. Giriş vektörü ağ üzerinden ileriye doğru iletilerek çıktı tahmini elde edilir,
4. Tahmin ile olması beklenen çıktıının hata sinyali hesaplanır,
5. Hata sinyali ağ üzerinden geriye doğru iletilir,
6. Toplam hatanın minimize etmek için ağırlıklar ayarlanır,
7. Toplam hata kabul edilebilir düzeye düşene kadar 2'den 7'ye kadar olan adımlar tekrarlanır.

4.2. Derin Öğrenme

Makine Öğrenmesi alanında verideki özelliklerin seçilmesi modellerin başarısını önemli ölçüde etkileyen bir adımdır. Makine Öğrenmesinde genellikle uzman bilgisine göre analiz edilerek el yordamıyla özellikler seçilirken Derin öğrenmede eğitim sürecinde özellikler otomatik olarak tespit edilmektedir (Aslan, 2018). Derin öğrenme bu sayede hem manuel özellik seçme esnasında meydana gelebilecek hatalardan kurtarır hem de alan uzmanlığı gerektirmeden otomatik özellik seçimi sağlar. Şekil 18'de geleneksel makine öğrenmesin ve derin öğrenmedeki işleyiş farklılığı gözler önüne serilmiştir.



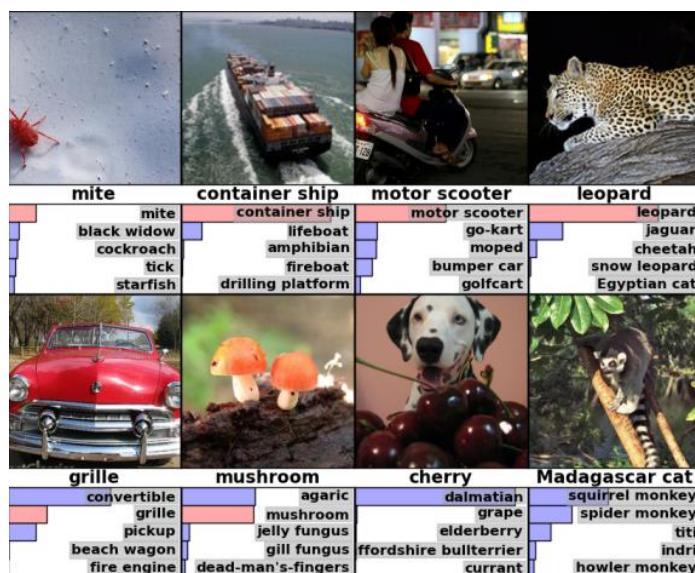
Şekil 18: Geleneksel Makine Öğrenmesinde ve Derin Öğrenmede özellik seçimi karşılaştırması (Alzubaidi, ve diğerleri, 2021)

Derin öğrenme algoritmaları, özellikle çok katmanlı sinir ağları, genel olarak makine öğrenmesine görece eğitim sürecinde daha fazla veriye ihtiyaç duymaktadır ve ayrıca gerek gerçekleştirilen kompleks hesaplamalar, gerekse veri tipinin çoğunlukla çok boyutta olması eğitim sürelerinin uzun sürmesine sebebiyet verebilmektedir.

4.3. Konvolüsyonel Sinir Ağları

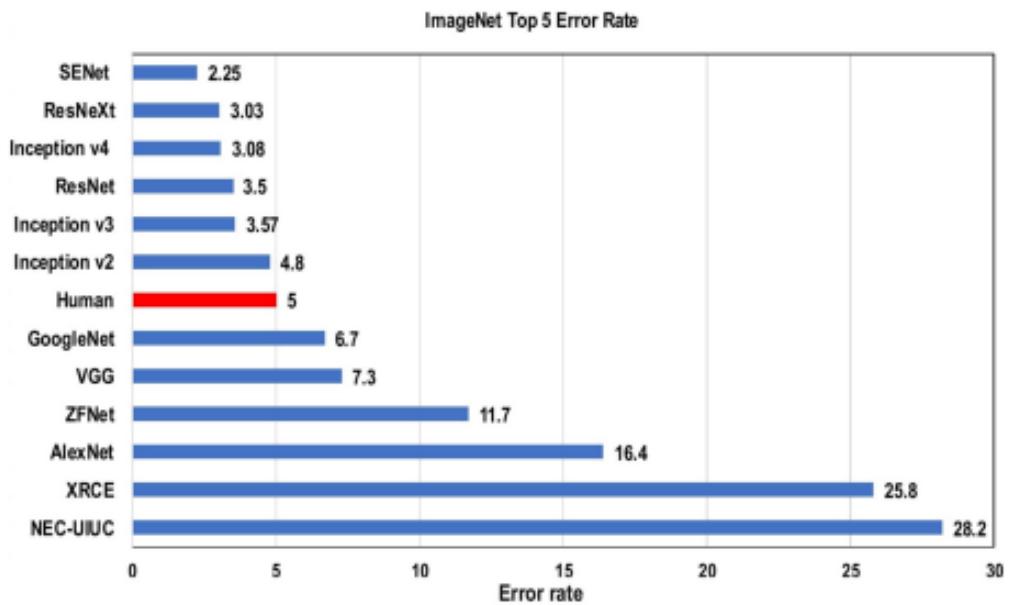
Tarihte CNN kavramı ilk olarak LeCun tarafından ortaya atılmıştır (1989). LeCun, 1998'de görüntü işleme kapsamında MNIST el yazısı rakamlarını tanımayı amaçladığı çalışmayla LeNet adını verdiği CNN mimarisini kullanarak yüksek bir başarı elde etmeyi başarmıştır. Yine de o dönemlerde büyük verilerin neredeyse hiç bulunmadığı ve günümüzde yaygınca kullanılan hesaplama kapasitesi yüksek GPU'ların henüz geliştirilmemiş olması nedeniyle pek kullanılan bir yöntem değildi. Sürekli gelişen teknoloji ile beraber veri kaynaklarının her saniye artması ve sürekli daha da iyileştirilen GPU'lar sayesinde bu engeller ortadan kalkmıştır.

Pascal Visual Object Challenge kapsamında ilk olarak 2010'da başlayan ve yıllık olarak gerçekleştirilmeye devam eden ImageNet Büyük Ölçekli Görsel Tanıma yarışmasının 2012 senesinde düzenlenen karşılaşmasında, ILSVRC-2012 veri seti üzerinde daha önceden %26,1 olan Top-5 hatasının bir CNN modeli olan AlexNet ile birlikte %15.3'e düşürülmesiyle performansını gözler önüne sermiştir. Şekil 19'da ILSVRC-2010 verisetinden alınan 8 adet test görüntüsü ve aynı model tarafından bu görüntüler için belirlenen en olası 5 etiket gösterilmektedir (Krizhevsky, Sutskever, & Hinton, 2017).



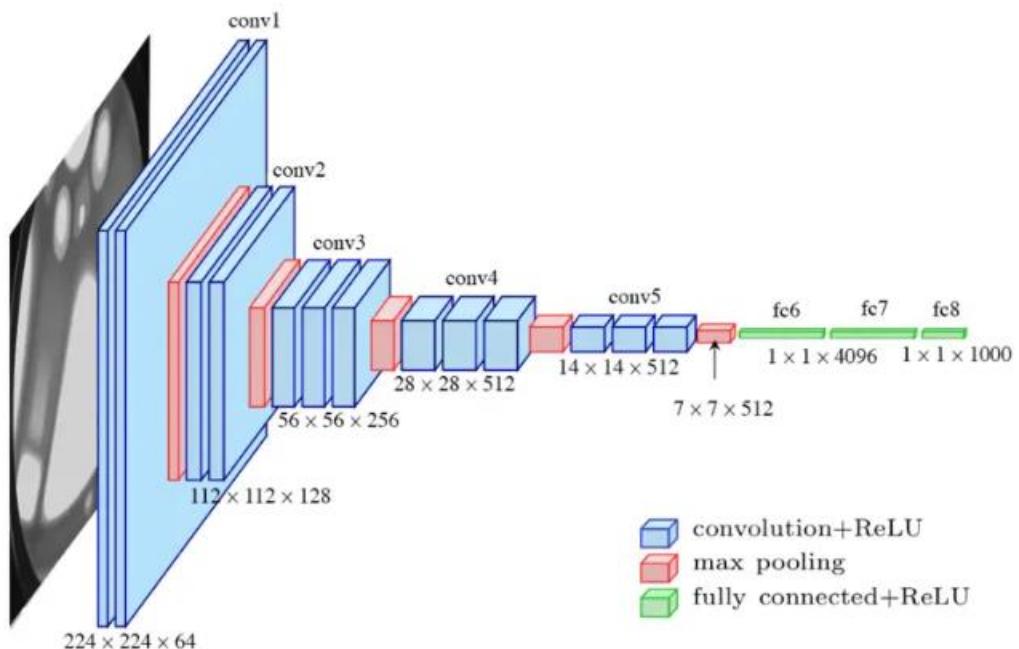
Şekil 19: ILSVRC-2010 verisine ait 8 adet test görüntüsü ve AlexNet modelinin görüntüler için sunduğu yüksekten küçüğe sıralanmış en olası 5 etiket tahmini (Krizhevsky, Sutskever, & Hinton, 2017)

Dahası, yine farklı bir yılda düzenlenen ImageNet yarışmasına ait ILSVRC 2015 veri kümesine göre insanlar %5.1 Top-5 hata skoru elde ederken (Russakovsky, ve diğerleri, 2015), ResNet modeli %3.57 Top-5 hata skorunu yakalayarak bir başarıya imza atmıştır (He, Zhang, Ren, & Sun, 2016). Şekil 20'de başarısı kanıtlanmış (state-of-art) mimarilerle insan başarısının karşılaştırılması göz önüne konulmuştur.



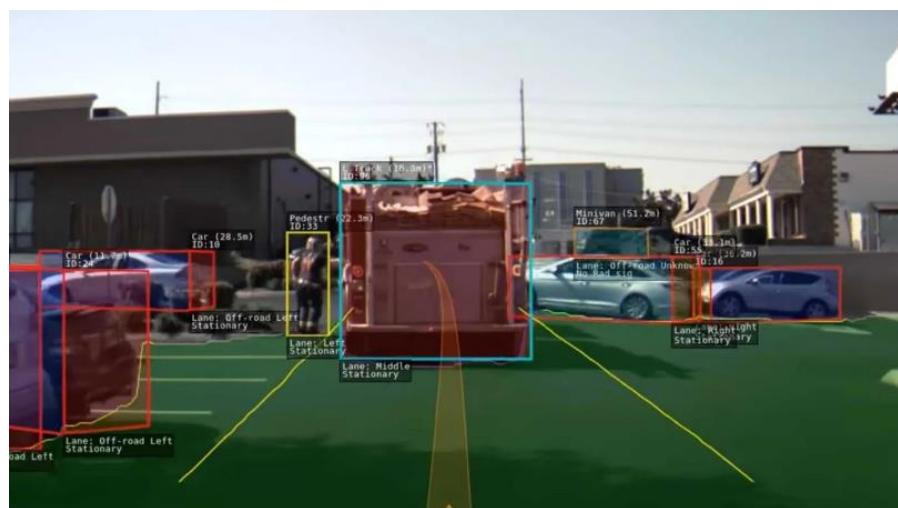
Şekil 20: İnsana kıyasla derin öğrenme performansı (Alzubaidi, ve diğerleri, 2021)

Burada AlexNet, ZFNet, GoogleNet, ResNet mimarileri sırayla 2012, 2013, 2014 ve 2015 ImageNet şampiyonlarıdır. Ayrıca 2014 yılında GoogleNet'in ardından 2.liği elde eden VGG (VGG-16) modelinin mimarisi Şekil 21'de verilmiştir (Sharma, 2020).



Şekil 21: 3x3 kernel'e sahip 13 adet Konvolüsyon katmanı ile 3 adet Tam Bağlı Katman'dan oluşan VGG-16 mimarisi (Sharma, 2020)

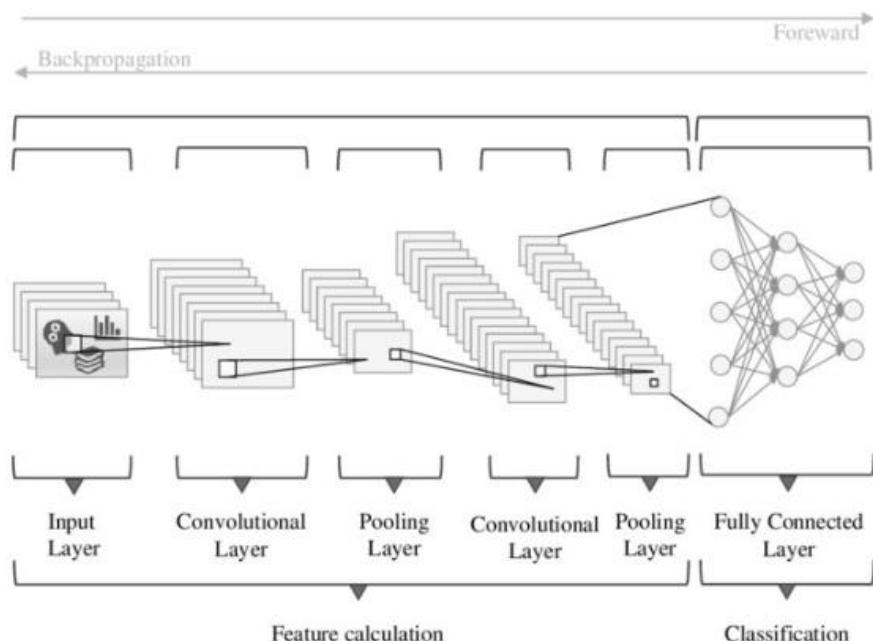
CNN, günümüzde popüler olan Google, Facebook, Pinterest ve Instagram gibi sosyal medya platformları tarafından yaygın bir biçimde kullanılmaktadır. CNN'ler genel olarak görsel tanıma amacıyla medikal, otomatik sürüs, biyometrik doğrulama, doküman analizi gibi bir çok farklı sektörde yer bulmuştur. Kalp hastalıkları, MR ile beyin tümörü ve kanser tespiti, kişinin yüzünden kimliğini tanıma ve araba destek uygulamaları (yolun, engellerin, arabaların, trafik levhalarının belirlenebilmesi) bu çalışmalara örnek olarak verilebilir (Gandharv, 2022). Otonom sürüs sektöründe Tesla'nın HydraNet'inde, Google Waymo'nun ChauffeurNet'inde ve Nvidia'nın otonom sürüs arabalarında CNN tekniklerinden faydalılmaktadır (Barla, 2023). Otonom sürüslerde nesnelerin tespitini belirten örnek bir görüntü Şekil 22'de gösterilmektedir.



Şekil 22: Otonom sürüslerde nesne tespiti (Barla, 2023)

4.3.1. Konvolüsyonel Sinir Ağları Mimarisi

Basit bir CNN mimarisi genel haliyle Girdi Katmanı, Konvolüsyon Katmanı, Havuzlama Katmanı, Tam Bağlı Katman ve Sınıflandırma Katmanından oluşmaktadır. Bu katmanlara ek olarak Batch Normalization, Rektifiye Edilmiş Doğrusal Birim Katmanı ve Dropout Katmanı gibi bir çok başka katmanlar da eklenip çıkarılarak birbirinden farklı başarılı mimariler elde edilebilmiştir. Şekil 23'de örnek bir CNN mimarisi gösterilmiştir.



Şekil 23: CNN modelinin örnek gösterimi (Anding, Haar, Polte, Walz, & Notni, 2019)

Girdi Katmanı

CNN'ler başta görüntü olmak üzere video, ses ve diğer sinyalleri girdi olarak alabilmektedir. Girdi boyutunun büyük olması işlem hacminin de artmasına sebep olmaktadır.

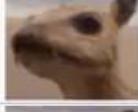
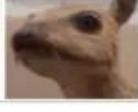
Konvolüsyon (Filtre) Katmanı

Geleneksel görüntü işleme tekniklerinde görüntüler çarpraz korelasyon (cross-correlation) ve konvolüsyon gibi bazı operasyonlarla manipüle edilebilmektedirler. Konvolüsyon katmanı, adından da anlaşılacağı üzere Konvolüsyonel Sinir Ağları'na ismini veren katmandır. Konvolüsyon katmanın ismi konvolüsyon olsa da tam tersine aslında çarpraz korelasyon operasyonu uygulanmaktadır (Rosebrock, 2021).

Bu operasyon için çekirdek(kernel) denilen filtre matrisi kullanılmaktadır. Filtre matrislerinin boyutu genellikle 3×3 , 5×5 , 7×7 , 9×9 veya 11×11 olmaktadır. Bu matris yardımıyla girdi olarak verilen verideki öznitelik matrisleri (activation map) çıkarılır. Filtrelerdeki ağırlık değerleri, CNN

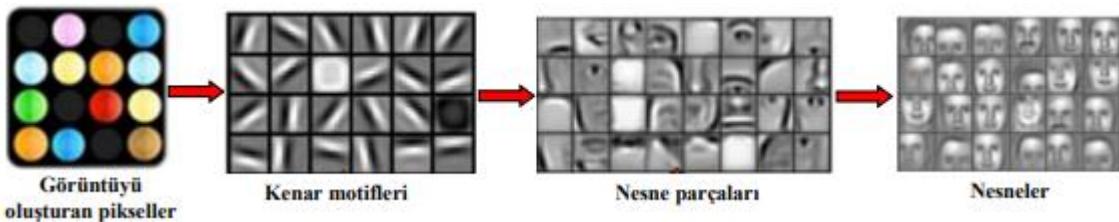
algoritmasının öğrenme sürecinde güncellenmektedir. Bu değerler güncellendikçe kenarların ve köşelerin tespit edilme kalitesi de artmaktadır. Filtrelerin ağırlık değerleri rastgele veya önceden eğitilmiş modellere göre başlatılabilmektedir.

Filtre matrisinin resim üzerindeki etkisi aşağıdaki Şekil 24'de vurgulanmıştır. Şekilde örnek bir hayvan görseli ile blurlama, keskinleştirme gibi işlemler için kullanılan bazı bilinen filtre matris çeşitleri, ve aynı görselin her bir filtre matrisi tarafından konvolüsyona uğramış son hali gösterilmektedir.

	Operation	Filter	Convolved Image
Identity		$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection		$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
		$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
Sharpen		$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
		$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)		$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)		$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Şekil 24: Resim üzerinde filtre etkileri (Doğan, 2020)

Konvolüsyon işleminin her uygulanışında farklı özellikler ortaya çıkarılmaktadır. CNN'de kullanılan ilk evrişimli katmanlar, görüntülerin düşük seviye özelliklerini (örneğin, kenarlar) yakalarken sonraki katmanlar yüksek seviye özellikleri (örneğin, şekiller ve belirli nesneler) çıkarırlar (Mostafa & Wu, 2021). Şekil 25'de motifler, nesne parçaları ve nesnelerin ilişkisi gösterilmiştir.



Şekil 25: CNN'de farklı katmanlarda meydana gelen nesne temsilleri (İnik & Ülker, 2017)

Kernel, görüntü üzerinde soldan sağa ve yukarıdan aşağı doğru görüntü boyunca hareket etmektedir. Kesikli cross-correlation operasyonunun 3 kanallı Red Green Blue (RGB) görüntüsü için matematiksel gösterimine Denklem(1)'de debynmiştir (Raitoharju, 2022):

$$Y[m, n] = \sum_{i=1}^K \sum_{j=1}^K \sum_{s=1}^C K[i, j, s] X[m - 1 + i, n - 1 + j, s] \quad (14)$$

Burada C kanal sayısına karşılık gelirken $K[i, j, s] = k_{ij}^s$ s'inci kanaldaki Kernel ağırlığını, X matrisi girdi matrisini veya 0 dolgulanmış halini, Y ise ortaya çıkan özellik matrisini temsil etmektedir.

Normal filtre sonucunda çıktı matrisinin girdi matrisinden daha küçük boyutta olması beklenir. Padding parametresi, girdi matrisinin etrafı pixeller ile doldurarak çıktı matrisinin boyutunu korumayı sağlar. Doldurulan pixelleri belirlemekte kullanılan bilinen iki adet padding teknigi vardır: Sıfır(zero) ve yansima (reflection). Zero padding çevreyi 0 pixelleri ile doldurarak keskin bir geçiş yaratır. Reflection padding teknigi ise kenardaki pixellerin yansımaları eklenerek yumuşak geçiş uygular.

Bir diğer parametre ise adım (stride) sayısıdır. Filtreler, girdi olarak verilen görüntü boyunca belirlenmiş adım sayısı kadar kaydırılarak gezdirilir ve böylece öznitelik matrisi (activation map) ede edilir.

$$H_{\text{çıktı}} = \left(\frac{H - F + 2P}{S} \right) + 1 \quad (15)$$

$$W_{\text{çıktı}} = \left(\frac{W - F + 2P}{S} \right) + 1 \quad (16)$$

Çıktı matrisinin yüksekliği ve genişliği sırayla Denklem (15) ve Denklem (16)'de verilmiştir. Girdi matrisinin yüksekliği H, genişliği W ve filtre matrisinin büyülüğu F, stride değeri S, dolgulama değeri ile P temsil edilmektedir.

Kernel, görüntü matrisi üzerinde gezerken görüntü matrisinin izdüşümündeki matrise alıcı alan (receptive field) denir. Alıcı alan ve kernelin boyutu aynıdır. Bu iki matrisin noktasal çarpımı sonucunda öznitelik matrisinin bir elemanının değeri elde edilir.

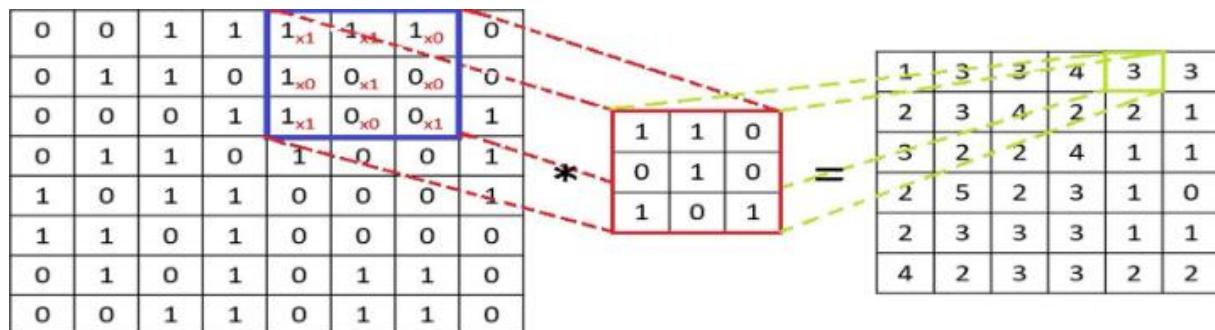
$$Receptive\ Field = A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \quad (17)$$

$$Kernel = B = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} \quad (18)$$

Denklem (17)'deki A ve Denklem (18)'deki B, 3×3 boyutunda matrisler olsun. Noktasal çarpımları Denklem (19)'de gösterilmiştir:

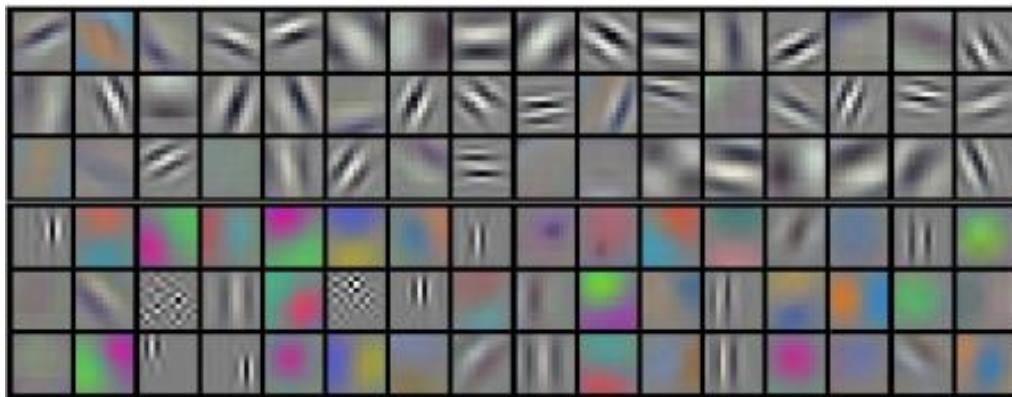
$$a_{1,1}b_{1,1} + a_{1,2}b_{1,2} + \cdots + a_{3,3}b_{3,3} \quad (19)$$

Şekil 26'de görüldüğü üzere 3×3 boyutundaki Kernel, 8×8 boyutundaki girdi matrisi üzerinde gezerek alıcı alan ile eleman bazlı çarpım uygulanmıştır ve $1 \times 1 + 1 \times 1 + 1 \times 0 + 1 \times 0 + 0 \times 1 + 0 \times 0 + 1 \times 1 + 0 \times 0 + 0 \times 1 = 3$ sonucu elde edilmiştir. Tüm bu işlemlerin sonucunda $8 - 3 + 1 = 6$, 6×6 boyutunda öznitelik matrisi oluşmuştur.



Şekil 26: Receptive field ile Kernel'in nokta çarpımı (Singh, Meitei, & Majumder, 2020)

Örneğin bir konvolüsyon katmanında k adet filtre kullanılırsa aynı katmanda çıktı olarak k adet bağımsız öznitelik matrisi oluşmaktadır. İlk konvolüsyon katmanında $224 \times 224 \times 3$ boyutundaki girdi görüntülerinden elde edilen 96 adet $11 \times 11 \times 3$ boyutundaki konvolüsyon kernel'leri Şekil 27'de gösterilmiştir (Krizhevsky, Sutskever, & Hinton, 2017).



Şekil 27: Örnek 96 Adet Konvolüsyon Kernel'i (Krizhevsky, Sutskever, & Hinton, 2017)

Toplu Normalizasyon (Batch Normalization) Katmanı

Bilindiği üzere, CNN ve diğer derin öğrenme mimarilerinin başarılı bir şekilde eğitebilmesi için çok miktarda veriye ihtiyaç duyulmaktadır. Bu modellere resimlerin tek tek sokulduğu düşünülürse her bir resim için ağırlıkların güncellenmesi öğrenme sürecini oldukça fazla uzatacaktır.

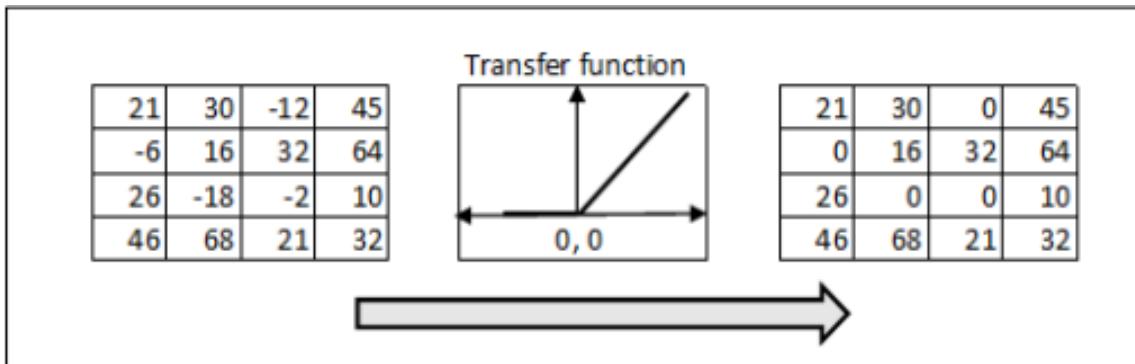
Geriye doğru yönelim yöntemiyle çıktıdan girdiye doğru bir yol izlerken katmanlarda bulunan ağırlıklar katman katman güncellenir. Yani bir katmandaki ağırlıkların güncellenmesi için önce diğer katmandaki ağırlıkların güncellenmiş olması gerekmektedir. Bu durum bazen Derin Öğrenme'de yaygın olarak ortaya çıkan İçsel Kovaryans Kayması'na (Internal Covariate Shift) neden olabilmektedir (Brownlee, 2019). Kullanılması opsiyonel olan Batch Normalization katmanı ise veriyi gruplar halinde normalleştirme yaparak eğitimi hızlandırır (Keskin, 2022). Internal Covariate Shift olsusunun önüne geçer ve eğitim döngüsü sayısını da (epoch) dramatik bir şekilde azaltır (Brownlee, 2019).

Rektifiye Edilmiş Doğrusal Birim Katmanı (ReLU: Rectified Linear Units Layer)

Rektifiye Edilmiş Doğrusal Birim, bir aktivasyon fonksiyonu çeşididir. ReLU katmanı, evrişim katmanında gerçekleşen matematiksel işlemlerin ardından doğrusallaşan ağı doğrusal olmayan ağa dönüştürür ve aynı zamanda ağıın hızını arttırır (Inik & Ülker, 2017). ReLU aktivasyon katmanı, ilk defa bir CNN mimarisi olan AlexNet'te kullanılmıştır (Abdelmalek, Ahmed, & Amine, 2019). Denklem (20)'de ReLU aktivasyon fonksiyonunun matematiksel denklemi verilmiştir.

$$f(x) = \max(0, x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (20)$$

Şekil 28'de, sol tarafta 4×4 boyutunda bir girdi matrisi, orta kısmında ReLU fonksiyonunun koordinat düzlemindeki grafiği ve sağ tarafta ise fonksiyon sonunda oluşan çıktı matrisi verilmiştir. Negatif değerlerin 0 değerini aldığı gözlenmektedir.



Şekil 28: 4×4 boyutundaki matrisin ReLU fonksiyonuna girmeden önceki ve girdikten sonraki durumu (Taşhan, 2017)

ReLU fonksiyonunun konvolüsyon katmanından gelen görüntü verisi üzerindeki etkisi Şekil 29'de gösterilmiştir.



Şekil 29: Giriş görüntüsünün sırasıyla konvolüsyon ve ReLU katmanlarından geçtikten sonraki çıktı görüntüleri (İnik & Ülker, 2017)

Bazen bu katman Konvolüsyon katmanın içinde sayılığinden nihai çıktıının eşitliği Denklem (2)'de ifade edilmektedir. Daha düzgün ifadeyle, Denklem (2)'de uygulanan * operasyonu sonrası elde edilen Y değerine yanlılık B değeri eklenip f aktivasyon fonksiyonuna sokulduğunda öznitelik matrisi \hat{Y} elde etmiş oluruz.

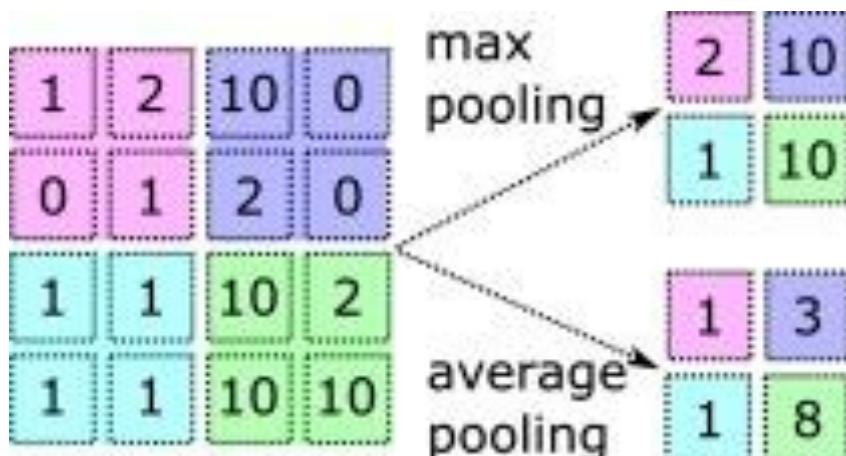
$$\hat{Y} = f(K * X + B) = f(Y + B) \quad (21)$$

Ortaklama Katmanı (Pooling Layer)

Ortaklama katmanı, verinin boyutunu düşürürken verideki özellik kaybının mümkün olduğunda az olmasını sağlar. Literatürde max pooling, average pooling ve minimum pooling gibi birçok pooling çeşidi bulunsa da CNN algoritmalarında en yaygın kullanılanı max pooling metodu olmuştur.

Max pooling işleminde bazı özelliklerin silinmesine karşın iyi eşleşmiş özelliklerin bilgisi korunmaktadır ve ayrıca görüntü boyutunu düşürmesi ağır hesaplama süresini oldukça azaltmaktadır (Inik & Ülker, 2017). Konvolüsyonel sinir ağları, pooling katmanları sayesinde translation invariant özelliği gösterir. Translation invariant özelliği, girdi üzerinde dönüşüm işlemleri (kaydırma, döndürme vb.) uygulansa dahi bu değişikliklere karşı direnç gösterilerek özelliğin yine de tespit edilebilmesidir (Goodfellow, Bengio, & Courville, 2016).

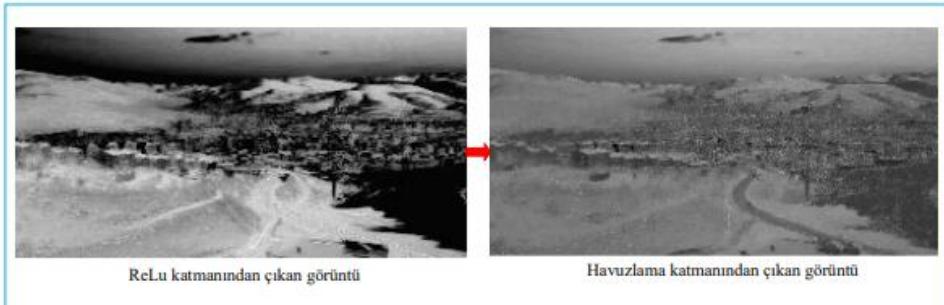
Ortaklama katmanına giren görüntünün yüksekliği ve genişliği konvolüsyon katmanındaki benzer şekilde Denklem **(15)** ve Denklem **(16)**'ye göre değişmektedir. Şekil 30'de pooling katmanına giren özellik matrisi üzerinde 2×2 boyutlu stride değeri 2 olan max pooling ve average pooling işlemlerinin uygulanışı gösterilmiştir.



Şekil 30: Stride değeri 2 iken 2×2 boyutlu max pooling ve average pooling işlemleri
(Raitoharju, 2022)

Girdi matrisi 4×4 boyutunda olup filtre matrisi ise 2×2 boyutundadır. Burada stride değeri 2 olduğundan her iki pooling işlemi sonucunda meydana gelen matrisin yükseklik ve derinliği sırasıyla Denklem **(15)** ve Denklem **(16)**'ye göre 2×2 olmaktadır.

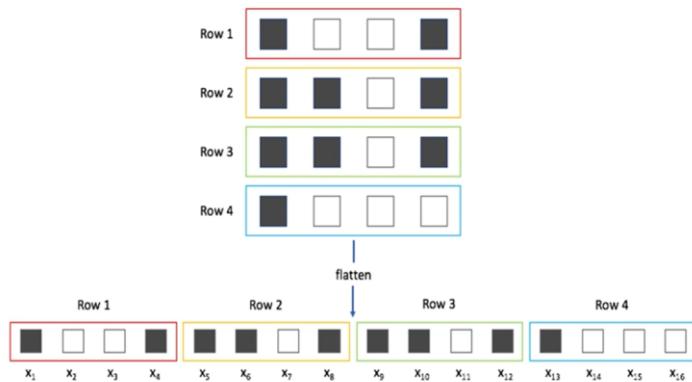
ReLU katmanından çıkan görüntünün havuzlama katmanından geçmesi sonucunda oluşan görüntüsü Şekil 31'de gösterilmiştir. Örneğin, soldaki görüntüde yollar membeyaz gözükmüşken sağdaki görüntüde yolların daha belirginleştiği anlaşılmaktadır.



Şekil 31: ReLU katmanından çıkan görüntünün Pooling katmanında işlem gördükten sonraki çıkan görüntüsü (İnik & Ülker, 2017)

Düzleştirme Katmanı (Flattening)

Önceki katmanlarda işlemlerden geçerek son hale gelen çok boyutlu veri, tam bağlı katmana girmeden önce bu katmanda tek boyuta yani vektöre indirgenir. Bunun sebebi tam bağlı katmandaki sinir ağlarının girdi değerlerini vektör olarak kabul etmesidir (Keskin, 2022). Şekil 32'de flattening katmanında örnek bir 4×4 boyutuna sahip matrisin 1×8 boyutundaki vektöre indirgendiği ifade edilmektedir.

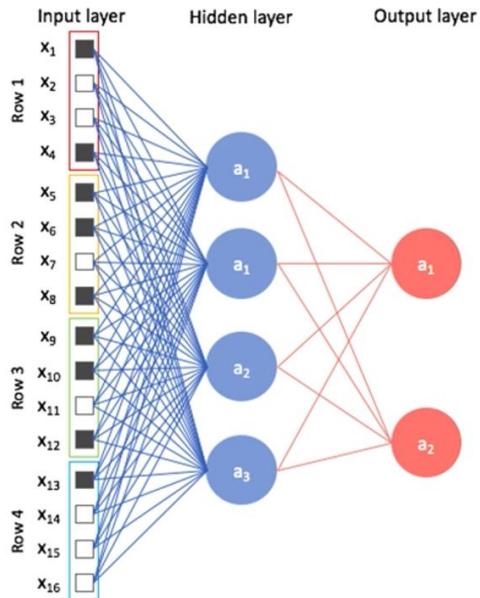


Flattening Katmanı

Şekil 32: Matrisin Düzleştirme (Flattening) Katmanı ile vektör haline getirilmesi (Keskin, 2022)

Tam Bağlı Katman

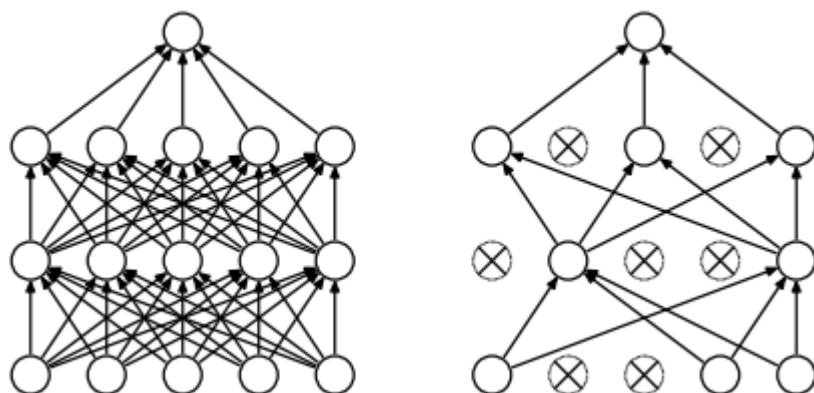
Tam Bağlı (Fully Connected) katmanlarında ise daha önceden tek boyuta indirgenmiş veri burada sinir ağları yardımıyla sınıflandırma işlemine tabii tutulur. ANN'lerdeki işlemler sonucunda sınıflandırmak üzere her sınıf için skor üretilir. Sınıf sayısı ile çıktı sayısı aynı miktardadır (Keskin, 2022). Şekil 33'de örnek basit bir tam bağlı katmanın yapısı gösterilmektedir.



Şekil 33: Basit bir Tam Bağlı Katman yapısı (Keskin, 2022)

Sönümleme (Dropout) Katmanı

Yapay sinir ağları, veriyi çok fazla öğrenmesi sebebiyle yaşanan aşırı öğrenme sorununu Dropout katmanı azaltabilmektedir. Tamamen bağlı sinir ağlarını oluşturan nöronların belli oranda rastgele sökümlenmesi ile bilinçli olarak meydana gelen hata rastgelelik sağlar. Böylece aşırı öğrenmenin önüne geçilir ve model başarısı artmış olur (Keskin, 2022). Şekil 34'de Dropout katmanının CNN üzerindeki etkisi gösterilmiştir.



Şekil 34: Solda 2 gizli katmandan oluşan bir sinir ağının yapısı ve sağda Dropout katmanının uygulanmasından sonra oluşan ağ (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014)

Sınıflandırma Katmanı

Çok katmanlı ANN'lerde tahmin skorunun olasılık dağılımı elde edilen katmandır. Tek etiketli çoklu sınıflandırma problemlerinde Softmax genellikle tercih edilen popüler bir aktivasyon fonksiyonudur. Denklem (22)'de Softmax fonksiyonu verilmiştir:

$$\hat{y}[i] = \frac{e^{y[i]}}{\sum_{c=1}^C e^{y[c]}} \quad (22)$$

Burada $y[i]$ nöronlardan çıktı olarak elde edilen vektörün i . elemanını, C ise çıktı sınıf sayısını temsil etmektedir. Tüm elemanlar 0 ile 1 arasında değerlenir ve tüm elemanların toplamı 1 değerini verir yani bir anlamda her sınıfın olasılıkları elde edilmiş olur (Raitoharju, 2022).

Kayıp Fonksiyonu

CNN'ler bir çok amaçla kullanılmaktadır ve mutlaka ilgili amaca uygun bir kayıp fonksiyonu seçilmelidir. Regresyon problemlerinde MSE tercih edilirken ikili veya çoklu sınıflandırma problemlerinde Binary Cross Entropy kullanılır. Denklem (23)'de Binary Cross Entropy eşitliği verilmiştir.

$$L(\hat{y}_s, t_s) = - \sum_{i=1}^N (t_s[i] \log(\hat{y}_s[i]) + (1 - t_s[i]) \log(1 - \hat{y}_s[i])) \quad (23)$$

Burada t_s ikili one-hot encoding vektörünü, N ise sınıf sayısını temsil etmektedir (Raitoharju, 2022).

4.3.2. Konvolüsyonel Sinir Ağları'nda Yaygın Karşılaşılan Zorluklar

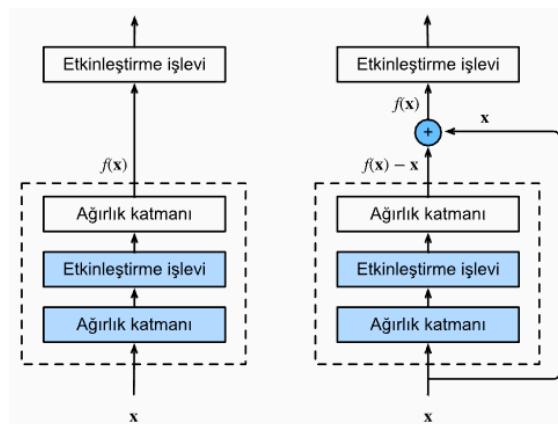
CNN'lerde yaşanan zorluklar başlıca overfitting, milyonlarca parametreyle eğitilmesi nedeniyle eğitim süresinin uzunluğu, eğitim süresinin uzunluğu sonucunda güncelleme kilitlemesi (update locking), küçük gradyanların üstel olarak azalmasıyla 0'a yaklaşarak gradyanların kaybolması veya büyük gradyanların üstel olarak artmasıyla kararsız hale gelerek gradyanların patlaması, ve içsel Kovaryans Kayması olarak özetlenebilir. Bu tür sıkıntılardan giderilerek modelin başarısının arttırılması amacıyla sıkça başvurulan aşağıdaki teknikler kullanılır ve bu teknikler birden fazla soruna çözüm olabilmektedir (Raitoharju, 2022):

- Veri Artırımı (Data Augmentation) kullanılarak çeşitlilik yaratılması,
- Dropout teknigi kullanılarak ağda rastgelelik sağlanması,
- Aktivasyon fonksiyonu olarak ReLU kullanılması,
- Transfer Learning yönteminden yararlanarak daha önceden benzer ve büyük veri setleri ile eğitilmiş CNN katmanlarının kullanımı,
- Ağırlık Düzenlileştirilmesiyle modelin karmaşıklığının azaltılması,
- Normalizasyon yöntemiyle alt katmanlarda gerçekleşen parametre güncellenmesinin sonraki katmanlarda yaşanan girdi dağılımının değişiminin önlenmesi.

4.3.3. ResNet, DenseNet ve CSPDenseNet

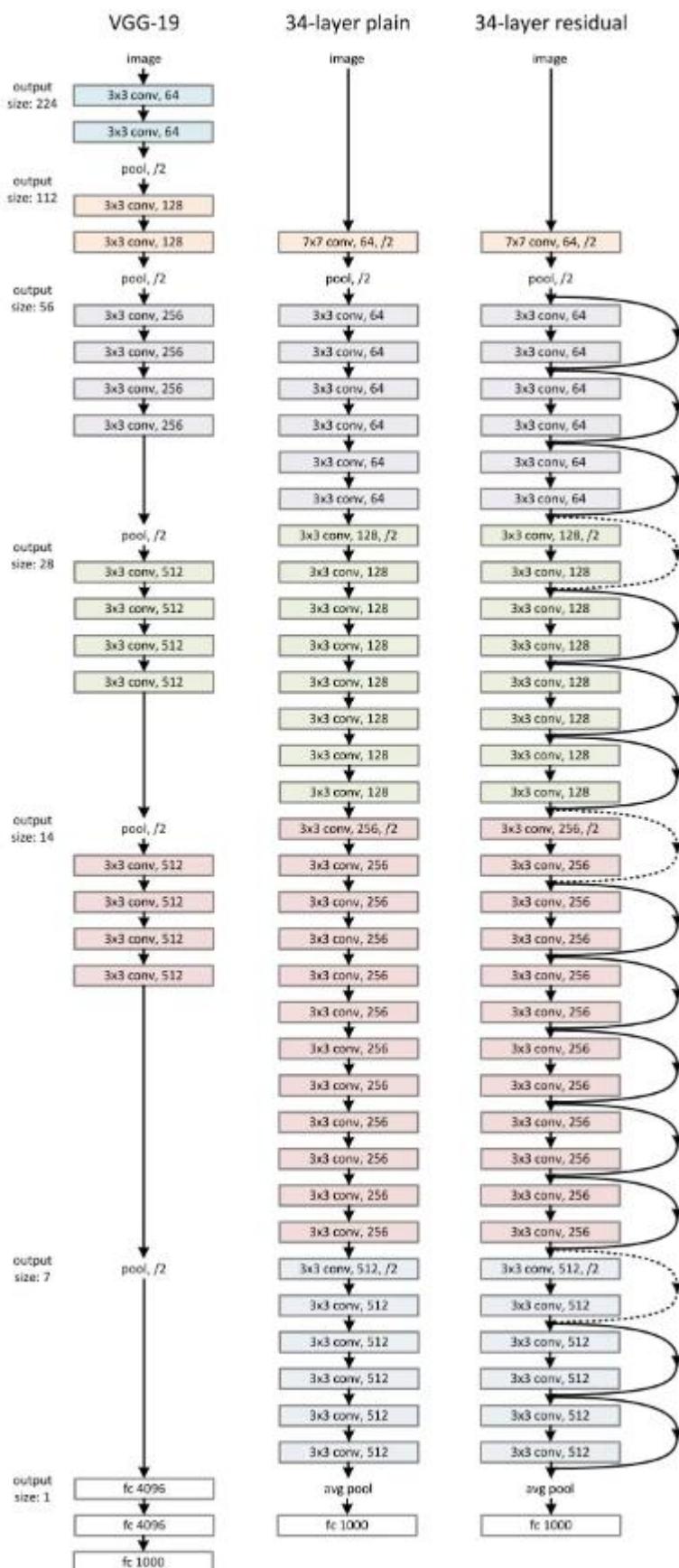
ResNet: Artık(Residual) bloklardan oluşan derin sinir ağına ResNet denir. Şekil(*)’de normal CNN bloğu ve Residual bloğu temsili olarak gösterilmiştir. Propagation sırasında yaşanan kaybolan gradyan problemini giderme motivasyonuyla önerilmiştir. Ayrıca kullanılan parametre sayısını da düşürmüştür (He, Zhang, Ren, & Sun, 2015). Bloğun girdisi, bloğun başından yeni bir kısayol bağlantısı aracılığıyla bloğun sonuna eklenir. Ağırlık katmanından çıkan veri ile x verisi aktivasyon fonksiyonunda toplanır. Bir x girdisi için; x girdisinin ağırlıklarla geçirdiği işlemler sonucuna $f(x)$, residual bloğun çıktısına ise $g(x)$ dersek denklem (*)’deki hesaplamalar geçerli olur. Sonuç olarak, g fonksiyonu hem doğrusal hem doğrusal olmayan şekilde bölünmüş olur (Dive Into Deep Learning, Tarih yok). Şekil 35’tে normal CNN bloğu ve sağda Residual blok karşılaştırılmıştır.

$$g(x) = \text{Aktivasyon}(f(x) + x)$$



Şekil 35: Solda normal CNN blok ve sağda Residual blok (Dive Into Deep Learning, Tarih yok).

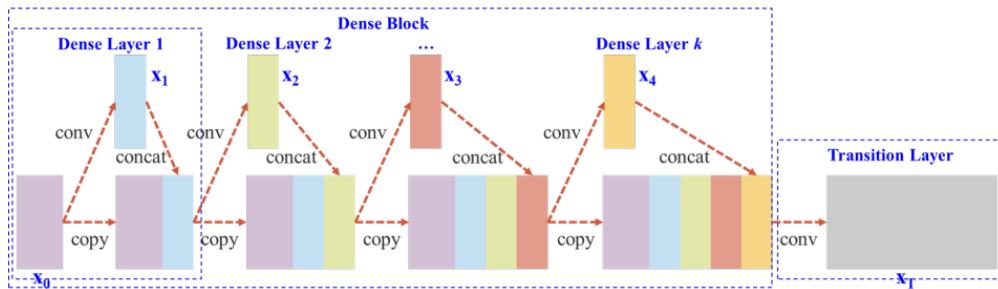
Şekil 36’da verilen VGG-19 modeli ile VGG-19 modelinin residual bloklar ile tekrar oluşturulmuş hali karşılaştırıldığında daha düşük hata oranı elde edildiği gözlemlenmiştir. Residual blokların başarısının anlaşılmasıından sonra zaman içerisinde ResNet50, ResNet101-152 gibi popüler ağlar ortaya çıkmıştır (He, Zhang, Ren, & Sun, 2015).



Şekil 36: VGG-19 ve ResNet (He, Zhang, Ren, & Sun, 2015)

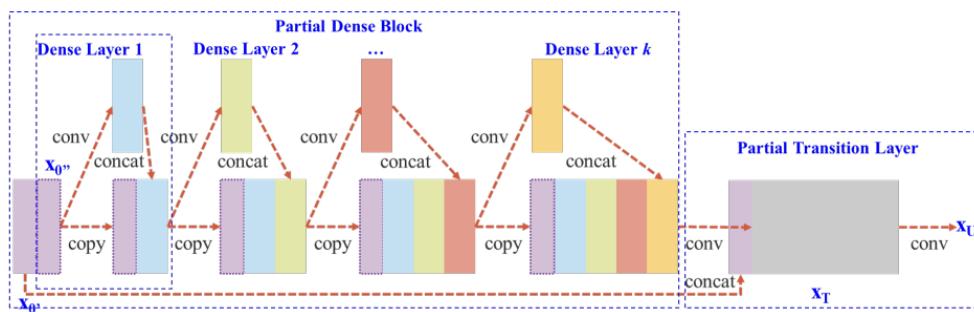
DenseNet: DenseNet, CNN katmanlarını içeren Yoğun (Dense) bloklardan oluşan sinir ağına denir. Dense blokların residual bloklardan farkı ise, toplama yerine birleştirme (concatenation) yöntemi kullanılmasıdır. Ağırlıklar arası bağımlılık çok yoğunlaştiği için bu isim verilmiştir. Şekil 37'de DenseNet ağında farklı Dense katmanlarından oluşan Dense Bloğunun sonundaki yoğunluk belirtilmiştir (Wang, ve diğerleri, 2019). Denklem (24)'de girdi x değerinin dense bloğunda çeşitli $f_n(x)$ fonksiyonlarından geçtikten sonra aldığı değerin hesaplanmasıne yer verilmiştir (Dive Into Deep Learning, Tarih yok).

$$x \rightarrow [x, f_1(x), f_2([x, f_1(x)]), f_3([x, f_1(x), f_2([x, f_1(x)])]), \dots] \quad (24)$$



Şekil 37: DenseNet örneği (Wang, ve diğerleri, 2019)

Cross Stage Partial DenseNet (CSPDenseNet): Wang ve arkadaşlarının önerdikleri CSP blok yapısıyla CNN ağlarda gerçekleşen matematiksel hesaplamların yaklaşık %20 oranında azalmasını sağladıklarını kanıtlamışlardır. Böylece bu blok yapısını kullanan sinir ağlarında öğrenme yeteneklerinin arttığını, darboğazların önüne geçildiğini ve bellek ihtiyacını düşürdüklerini duyurmuşlardır (Wang, ve diğerleri, 2019). Şekil 38'de CSPDenseNet örneği verilmiştir. CSPNet'te temek katmanın özellik haritası ikiye bölünür; bir bölümü dense bloktan geçenken diğer bölüm ise dense blok üzerinden iletilen diğer özellik haritası ile birleştirilir.



Şekil 38: CSPDenseNet örneği (Wang, ve diğerleri, 2019)

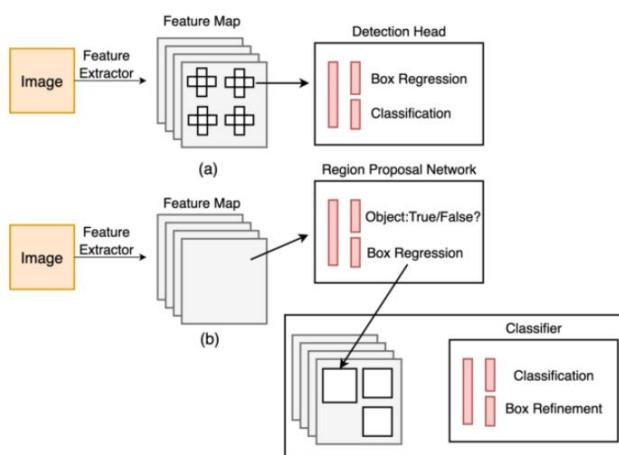
4.4. CNN ile Nesne Tespiti

Nesne tespiti, görüntü işleme disiplinin bir alt dalıdır. Nesne tespiti problemleri, “resim veya görüntüde bir obje varlığı var mı ?, var ise bu obje nerede ?” sorularına cevap arar. Sınıflandırma (classification) ve yerini belirleme (localization) işlemlerinin birleşik gerçekleştirilemesine tespit (detection) işlemi denir. Şekil 39’da bu işlemler arası fark gösterilmiştir. Gerçek dünyadan alınan görüntüler üzerinde nesnelerin sınıfı, konumları, uzaklıkları, hızı, adeti, ve rengi gibi nesnelere ait diğer özelliklerin tespit edilmesini amaçlamaktadır.



Şekil 39: Classification, Localization ve Detection farkı (Singh, 2020)

Nesne tanıma; İnsan trafiği, araç trafiği, yaya tespiti arama kurtarma, ev güvenliği, ülke sınır güvenliği, Rusya ile Ukrayna arasındaki savaşta gücünü kanıtlayarak değerine değer katan İnsansız Hava Uçakları (IHA), ve diğer kara, hava ve deniz savunma sanayi gibi sayısız geniş bir kullanım alanına sahiptir. Obje tespiti görevlerini başarıyla gerçekleştirebilmek için literatürdeki çalışmalar incelendiğinde CNN tabanlı modellerin kullanıldığı anlaşılmıştır. Nesne tespiti yöntemleri, aşamalarına göre iki aşamalı ve tek aşamalı olarak 2 gruba ayrılmaktadır.



Şekil 40: Tek ve çift aşamalı yöntemlerin işleyişi (Sharma, 2022)

4.4.1. İki Aşamalı (Two-Shot) Nesne Tespiti

İki aşamalı nesne tespit modellerinde, ilk aşamada nesnelerin potansiyel konumları için öneriler (proposal) belirlenir. İkinci aşamada ise Full Connected katmanlarda konumlar için sınıflandırma tahminleri yapılır. Bu tür yapılar her ne kadar yüksek başarı sağlsa da ağırlık miktarlarının devasa olması nedeniyle hesaplama yönünden çok masraflıdır, diğer ifadeyle oldukça yavaş çalışmaktadır. Temel olarak tanınan modeller Şekil(*)’de görüldüğü üzere R-CNN, Fast R-CNN, Faster R-CNN, RFCN ve Mask R-CNN’lerdir (Kundu, 2023).

4.4.2. Tek Aşamalı (One-Shot) Nesne Tespiti

Tek aşamalı nesne tespit modellerindeki yaklaşım ise hiçbir öneri yapmadan (proposal-free) tek aşamada tüm sınıfları ve tüm lokasyonları tek bir ağda aynı anda tahmin ederek doğrudan nesne tespiti yapmaktadır. Yani bir veri ağdan geçerken hem sınıflandırma işlemi hem de tespit işlemi aynı anda yapılır.

Kesinlik değerinden feragat edilse de kompüterSEL anlamda iki aşamalı yöntemlerin aksine çok hızlı çalışmaktadır. Bu yüzden gerçek zamanlı nesne tespitlerinde genel olarak Tek Aşamalı Nesne Tespit methodları kullanılmaktadır.

Şekil(*)’deki görselde belirtilen YOLO (You-Only-Look-Once) ve SSD(Single Shot Multibox Detector) algoritmaları bu yaklaşımı kullanan modeller olarak örnek verilebilir. YOLO, bu işlemi tek ağda tek seferde gerçekleştirdiği için bu ismi almıştır. Accuracy ve FPS anlamında optimum denge sağlar. SSD ise çok hızlı çıkarım yaparak düşük işlem maliyeti nedeniyle IHA, Blueberry PI gibi gömülü sistemlerde tercih edilir.

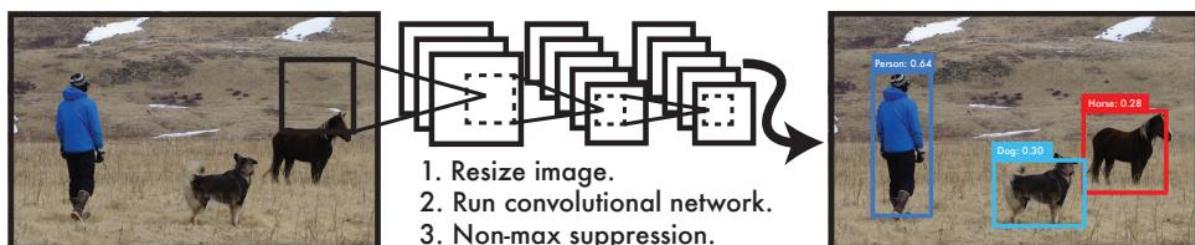
BÖLÜM 5.

MATERYAL-METHOD

Araştırmmanın bu bölümünde, gerçek zamanlı hareketli nesne tespiti için YOLO algoritmasının v5 ve v8 sürümleri kullanılarak atık verisetiyle eğitilmiştir. Daha sonra eğitilen modeller test verileriyle modellerin tespit doğruluğu analiz edilip kıyaslamaları yapılmıştır. En sonunda ise atıklar üzerinde nesne tespiti yapılabilmesi için YOLO modelini kullanan bir obje tespiti uygulaması ortaya çıkarılmıştır.

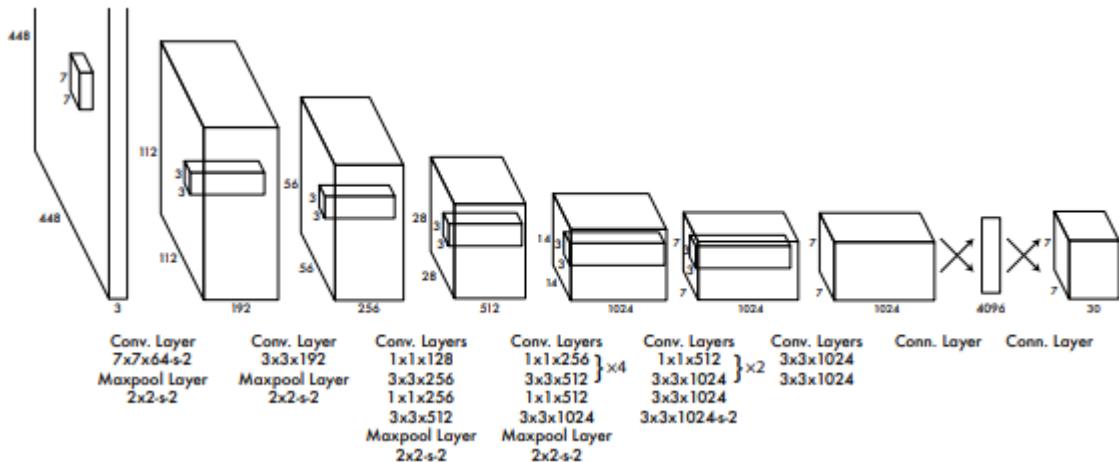
5.1. YOLO Algoritması

YOLO, görüntüyü “nerede obje var ve sınıfı ne?” mantığıyla tarar. Washington Üniversitesi, Allen Yapay Zeka Enstitüsü ve Facebook Yapay Zeka Araştırma kurumuna mensup Redmon ve arkadaşları tarafından tek aşamalı sinir ağı olarak nesne tespiti görevleri için uçtan uca optimize edilebilir şeklinde 2015'te tanıtılmıştır. R-CNN ve DPM gibi diğer gerçek zamanlı tespit modellerine göre hatalı yer tespiti yapmasına rağmen 2 kat daha fazla ortalama hassasiyet doğruluğu sağlamakla kalmayıp 25 milisaniye(ms) gecikme hızıyla göze çarpmıştır (Redmon, Divvala, Girshick, & Farhadi, 2016). Şekil 41'de modele genel bakış gösterilmiştir.



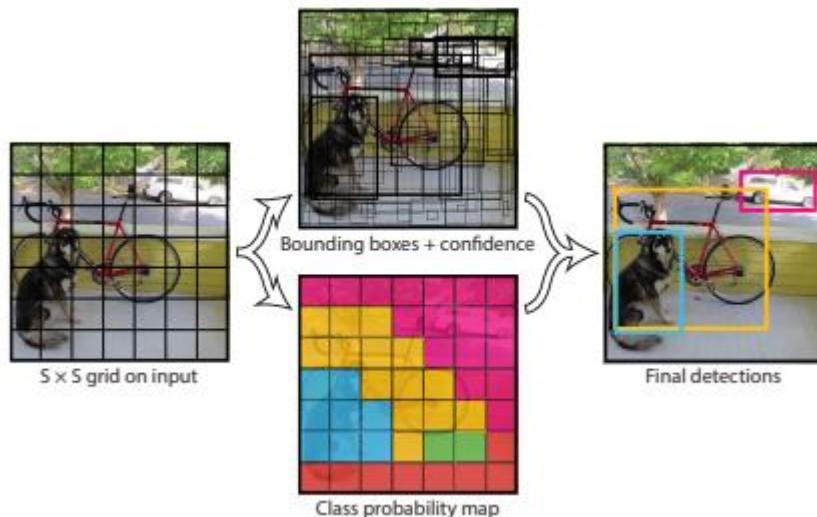
Şekil 41: YOLO modeline genel bakış (Redmon, Divvala, Girshick, & Farhadi, 2016)

Objenin tespiti için ImageNet veriseti üzerinde kullanılan CNN ağından oluşan YOLOv1 model mimarisini Şekil 42'de verilmiştir. Modelde 24 adet CNN katmanı ve 2 adet tam bağlı katmandan oluşmaktadır. Evrişim katmanlarında ve tam bağlı katmanlarında aktivasyon fonksiyonu olarak Leaky ReLU kullanılmaktadır.



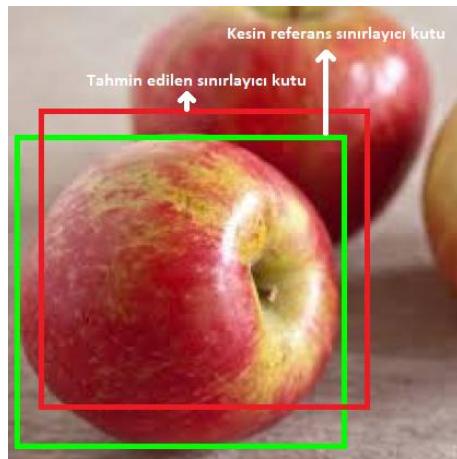
Şekil 42: YOLOv1'in mimaris (Redmon, Divvala, Girshick, & Farhadi, 2016)

YOLO'da resimler SxS ızgara hücrelerine bölünür. Her bir ızgara hücresi, resimdeki belirli bir bölgeye odaklanır ve nesne merkezi kendisindeyse nesne tespiti yapmakla sorumludur. Her hücre belirli B sayıda sınırlayıcı kutuyu tahmin eder. Her bir sınırlayıcı kutunun ise bir merkezi (x, y), genişliği w , yüksekliği h ve güven skoru (confidence score) vardır (Nepal & Eslamiyat, 2022). Şekil 43'de konuya anlatan görsel paylaşılmıştır.



Şekil 43: SxS grid, sınırlayıcı kutular, sınıf olasılık haritası ve tespitler (Redmon, Divvala, Girshick, & Farhadi, 2016)

Birkaç hücre aynı nesnenin merkezinin kendisinde bulunduğu söylenebilir, dolayısıyla fazladan sınırlayıcı kutu oluşabilir. Sınırlayıcı kutuların tahmini yapıldıktan sonra en doğruya yakın sınırlayıcı kutunun seçiliği için IoU(Intersection Over Union) değeri ölçülür. IoU, nesnenin bulunup bulunmadığına belli bir eşeğe göre karar vermeyi sağlar. nesnenin kesin referans kutusu (ground truth box) ve tahmin edilen sınırlayıcı kutusunun (predicted bounding box) birleşim alanlarındaki piksel sayısı ve kesişim alanlarındaki piksel sayısının bölünmesiyle bulunur. Çıkan sonuç, eşiği geçmesi durumunda nesnenin bulunduğu anlamına gelir (Hanbay & Üzen, 2017). Eşliğin geçilmemesi durumunda da modelin nesne olmadığını öngörmelidir. Şekil 44'te tahmin edilen ve gerçekte olan kesin referans sınırlayıcı kutusu örneği gösterilmiştir.



Şekil 44: Bir nesne için tahmin edilen ve kesin referans sınırlayıcı kutular

Güven skoru, modelin bir sınıf için ne kadar emin olduğunu gösteren skordur. Tahmin edilen nesnenin olasılığı $\Pr(\text{Objet})$ ve objenin gerçek referans ve tahmin edilen sınırlayıcı kutuların $IOU_{\text{tahmin}}^{\text{gerçek}}$ skoru ile Denklem (25)'deki matematiksel hesaplama sonucu güven skoru elde edilir.

$$\text{Güven Skoru} = \Pr(\text{Objet}) * IOU_{\text{tahmin}}^{\text{gerçek}} \quad (25)$$

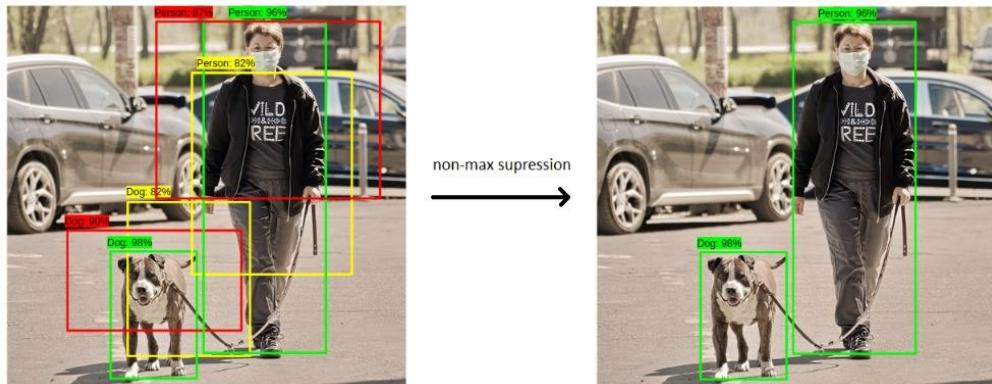
Sınıflar özelinde güven skorları sınıf koşullu olasılıklar ve obje güven skorunun çarpımı ile elde edilir. Denklem (26)'da bu hesaplamaya ilişkin sınıfa özel güven skoru formülize edilmiştir.

$$\Pr(\text{Sınıf}_i | \text{Objet}) * \Pr(\text{Objet}) * IOU_{\text{tahmin}}^{\text{gerçek}} = \Pr(\text{Sınıf}_i) * IOU_{\text{tahmin}}^{\text{gerçek}} \quad (26)$$

YOLOv1'in kayıp fonksiyonu Denklem (27)'de verilmiştir. Kayıp fonksiyonunda öngörülen nesnenin konumu, nesnenin grid içerisinde bulunurken ve bulunmazken ne oranda hatalı olduğu, sınırlandırmanın doğru yapılmıştır yapılmadığı ölçülmektedir.

$$\begin{aligned}
 \text{Kayıp Fonksiyonu} = & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \eta_{ij}^{obj} [(x_i - \hat{x}_l)^2 + (y_i - \hat{y}_l)^2] \\
 & + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \eta_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_l})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_l})^2 \right] \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \eta_{ij}^{obj} (C_i + \hat{C}_l)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \eta_{ij}^{noobj} (C_i + \hat{C}_l)^2 \\
 & + \lambda_{noobj} \sum_{j=0}^{S^2} \eta_{ij}^{noobj} \sum_{c \in \text{Siniflar}} (p_i(c) + \hat{p}_l(c))^2
 \end{aligned} \tag{27}$$

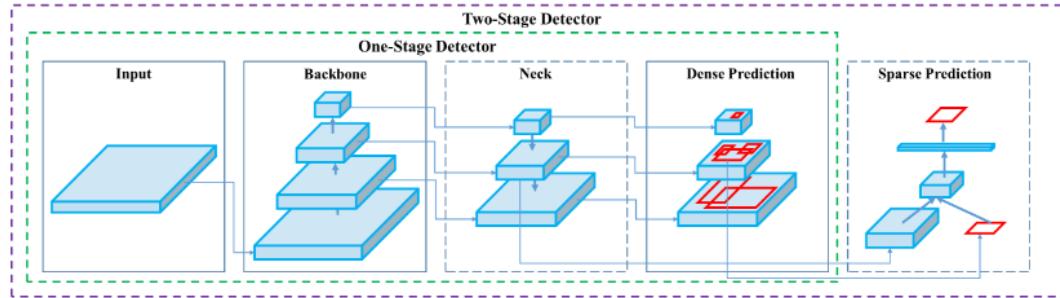
Şekil 45'te gösterilen maksimum olmayan bastırma (non-maximal suppression) teknigi ile düşük güven puanına sahip sınırlayıcı kutular susturulur. İlgili teknik yine tek aşamalı SSD modellerinde de kullanılmaktadır.



Şekil 45: Non-maximal Suppression (Singh, 2020)

5.1.1. YOLO Genel Yapısı

Temel olarak hem tek aşamalı hem de iki aşamalı tespit algoritmaları, omurga (backbone), boyun (neck) ve kafa (head) bölümlerinden oluşur. İki aşamalı tespit algoritmalarında ek olarak Spase Prediction bölümü bulunur. Şekil 46'de iki aşamalı ve tek aşamalı tespitçilerin şeması verilmiştir.



Şekil 46: Dedektörlerin yapısı (Bochkovskiy, Wang, & Liao, 2020)

Backbone: Görsel verinin ilk girdiği bölüm olmakla beraber özellik haritalarının çıkarımının (feature formation) uygulandığı kısımdır. DenseNet tabanlı CSPResNext50, CSPDarknet53, EfficientNet-B3 ağları backbone'a örnek verilebilir.

Neck: Backbone'da elde edilen özellikler Neck kısmında toplanır (feature aggregation). Birçok neck türü bulunsa da YOLOV3'te FPN kullanılırken; YOLOv5 ve YOLOv8'de FPN'e bottom-up path augmentation eklenmiş hali olan Path aggregation Network (PANet) kullanılmaktadır. Özellik haritaları arasında aşağıdan yukarıya ve yukarıdan aşağıya bir akış mevcut olup tüm kapsamlı bilgi toplayarak edinmeyi sağlamaktadır (Çakmak, 2020). Şekil 47'de PANet yapısı gösterilmiştir.

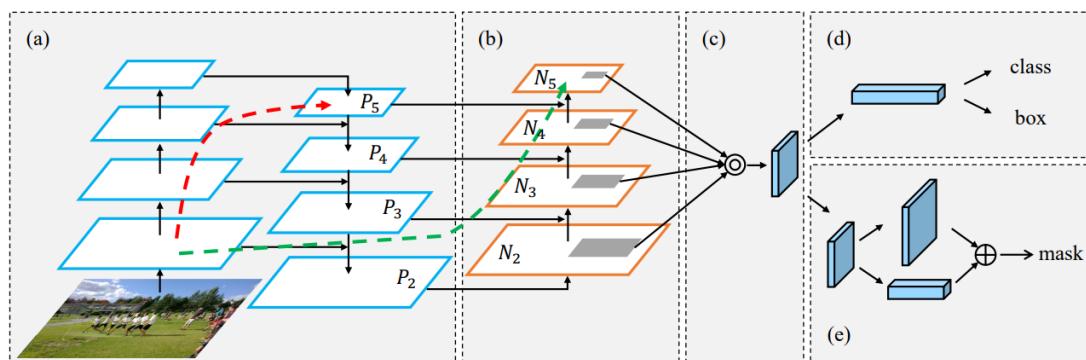


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

Şekil 47: PANet (Liu, Qi, Qin, Shi, & Jia, 2018)

Head: Head kısmında da bounding box'lar tespit edilip, classification işlemi yapılır. Küçük objeler yüksek çözünürlük ve fazla miktarda çapa kutusu (anchor box) gerektirirken büyük objeler için düşük çözünürlük ve daha az miktarda anchor box yeterli olmaktadır. 80x80x256, 40x40x512, 20x20x512 boyutlarını kabul eden 3 ayrı detection katmanı bulunmaktadır. Bu katmanlar sırasıyla küçük, orta, büyük boyuttaki objeleri tespit eder. Dolayısıyla daha büyük nesneler, en düşük çözünürlük ve en küçük anchor boxa sahip son tespiti katmanında tespit edilirken, daha küçük nesneler ise daha yüksek çözünürlük ve daha fazla anchor kutusuna sahip önceki detection katmanında tespit edilir (Jocher, 2023).

5.1.2. YOLO Tarihi

Backbone, neck ve head bölümlerinin farklı kullanımlarıyla orijinal YOLO günümüze kadar YOLOv2, YOLOv3, YOLOv4, YOLOv5 ve YOLOv8 sürümleriyle güncellenerek karşımıza çıkmıştır. Tablo 1'de çeşitli bölümlerden oluşan YOLOv3, YOLOv4, YOLOv5 mimarilerine yer verilmiştir.

	YOLOv3	YOLOv4	YOLOv5	YOLOv8
Neural Network Type	Fully convolution	Fully convolution	Fully convolution	Fully convolution
Backbone Feature Extractor	Darknet-53	CSPDarknet53	CSPDarknet53	CSPDarknet53
Loss Function	Binary cross entropy	Binary cross entropy	Binary cross entropy and Logits loss function	Binary cross entropy and Logits loss function
Neck	FPN	SPP and PANet	SPP and PANet	SPPF and PANet
Head	YOLO layer	YOLO layer	YOLO layer	YOLO layer

Tablo 1: YOLO'nun farklı sürümleri ve değişen yapısı (Nepal & Eslamiyat, 2022).

Sürüm değişiklikleri aşağıdaki gibi olmuştur:

YOLOv2: Batch normalization özelliği overfitting'in önüne geçilmesi için CNN katmanlarına eklendi.

YOLO9000: YOLOv2'de sadece 20 tane sınıf için tespit yapılabılırken, YOLO9000 sürümünde 9000 sınıf için tespit yeteneği getirilmiştir (Redmon, 2016).

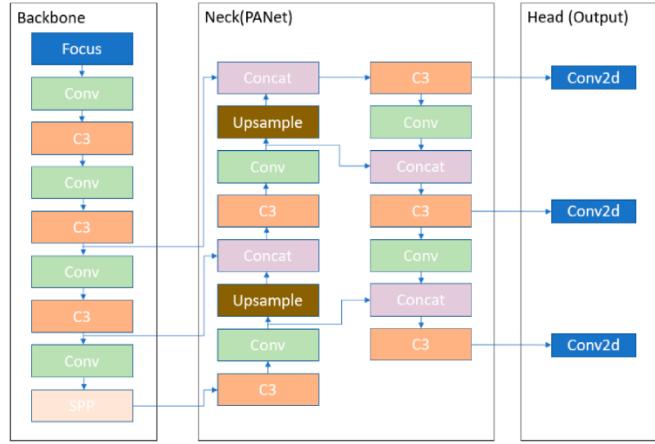
YOLOv3: Küçük objeleri tespit etmekte zorlanan DarkNet19 omurgası yerine DarkNet53 omurgası kullanıldı.

YOLOv4: DarkNet53 yerine CSPDarknet53 (Cross Stage Partial Network) omurgasına geçirilerek hız ve doğruluk artırıldı.

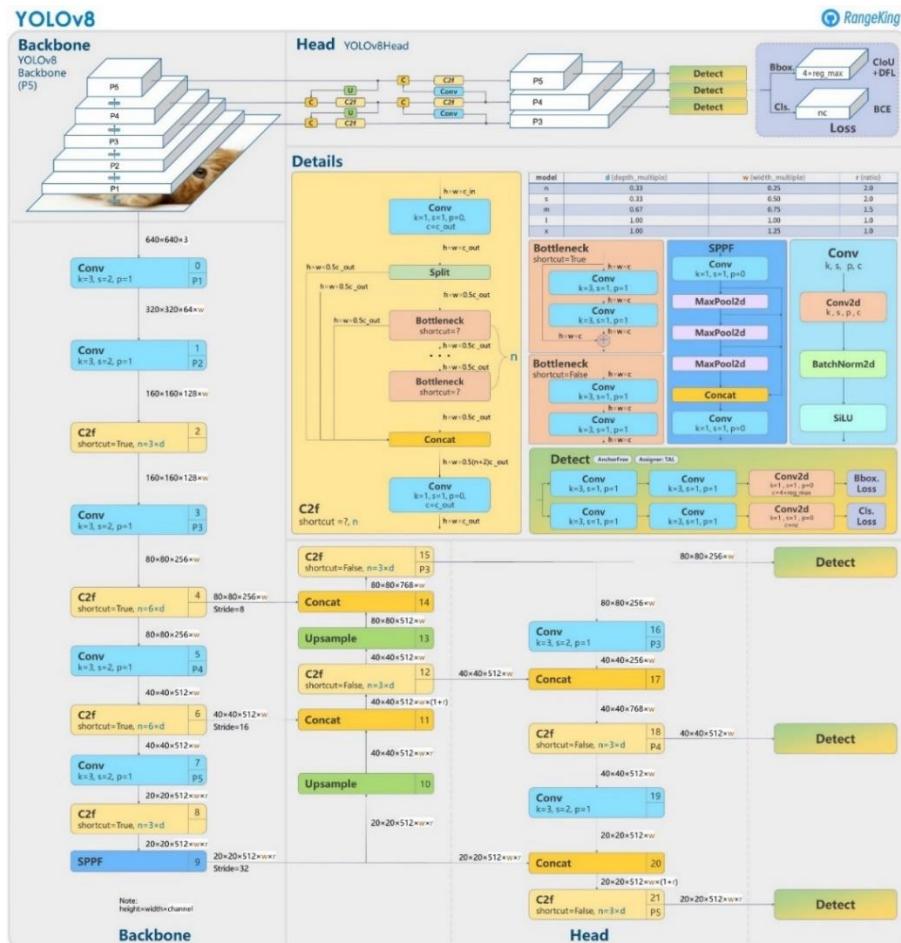
YOLOv5: DarkNet framework yerine PyTorch framework'üne geçildi. Model konfigürasyonu için CFG dosya formatından Yet Another Markup Language (YAML) dosya formatına geçildi.

YOLOv8: YOLOv5'ten farklı olarak Grid S ve RoadNet gibi ek özelliklere sahiptir, bu da ona belirli senaryolarda daha iyi performans sergileme imkanı tanımaktadır (Jocher, Are class and box losses calculated the same in YoloV8 and YoloV5?, 2023).

Bu çalışmanın gerçekleştirildiği tarihe kadar YOLO sürümlerinden en güncel mimari 2023 yılında çıkan YOLOv8 mimarisidir. Şekil 48'de YOLOv5 mimarisinin basitleştirilmiş hali gösterilmiştir. YOLOv8 mimarisinin hem özet hem de ayrıntılı olarak ise Şekil 49'de gösterilmiştir.



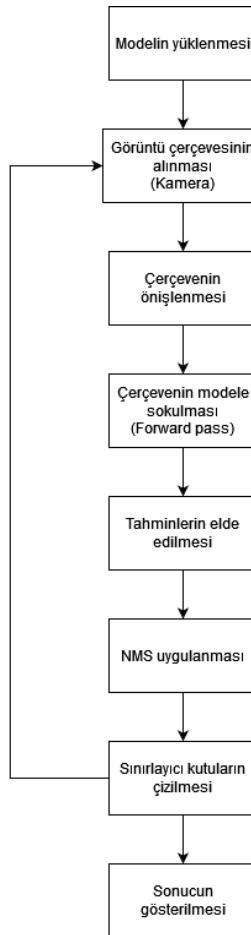
Şekil 48: YOLOv5 mimarisi için basitleştirilmiş bir gösterim (Nepal & Eslamiyat, 2022).



Şekil 49: YOLOv8 mimarisi için detaylı bir gösterim (King, 2023)

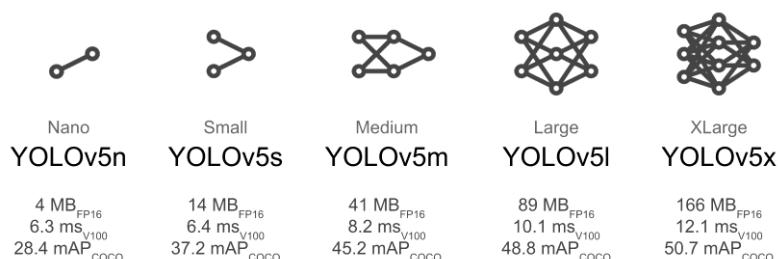
5.2. Uygulama

Hayata geçirilen uygulama sonucu görüntülerde bulunan varlıklar üzerinde dikdörtgen sınır çizgileri ve her nesnenin kategorisi yüzdelik bazda güven puanıyla belirtilmektedir. Çalışmada yapılan uygulama Şekil 50'deki akışı izleyecektir:



Şekil 50: Uygulamanın genel akış diyagramı

Uygulamanın hazırlanması esnasında ilk önce YOLOv5 ve YOLOv8 derin öğrenme mimarilerinin hem başarı hem de hız olarak optimum olması için Şekil 51'de verilen model spektrumundan medium(m) sınıfındaki sahip modeller eğitilecektir. Ardından eğitilen bu modeller, başarısına göre analiz edilecektir. En son başarılı model üzerinde uygulama geliştirilmesi hedeflenmiştir.



Şekil 51: YOLO model spektrumu (Jocher, Chaurasia, & Qiu, 2023)

5.2.1. Veri Seti

Çalışma için atık görsellerinin veriseti araştırması yapılmıştır. Son yıllarda bilgisayarlı gözü projelerinde oldukça popülerleşen Roboflow sitesinden açık kaynak olarak sunulan “YOLO Waste Detection Image Dataset” ismindeki bir veriseti bulunup içerisindeki elemeler yapılmıştır (Projectverba, 2022). Buna göre:

- Alüminyum kutu,
- Plastik şişe,
- Plastik poşet,
- Organik,
- Karton kutu,
- Cam şişe,
- Kimyasal içerik

sınıflarından herhangi birini içeren tüm görüntüler seçilmiştir. Ortaya 4,103 adetten oluşan görüntü kümlesi oluşturulmuştur. Bu görüntülerin etiketlemesi (labelling) çoktan yapıldığı için etiketlemenin nesnenin tam üzerinde olmadığı için küçük düzeltmeler haricinde ek bir düzenlemeye ihtiyaç duyulmamıştır. Nesne içeriklerinden birkaç örnek Şekil 52'de paylaşılmıştır.



Plastik şişe



Plastik poşet



Organik



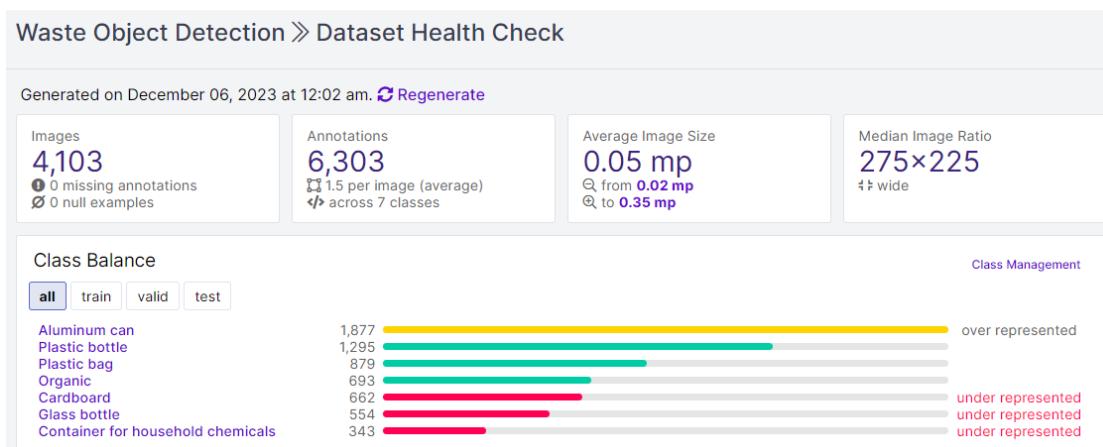
Alüminyum kutu



Karton kutu

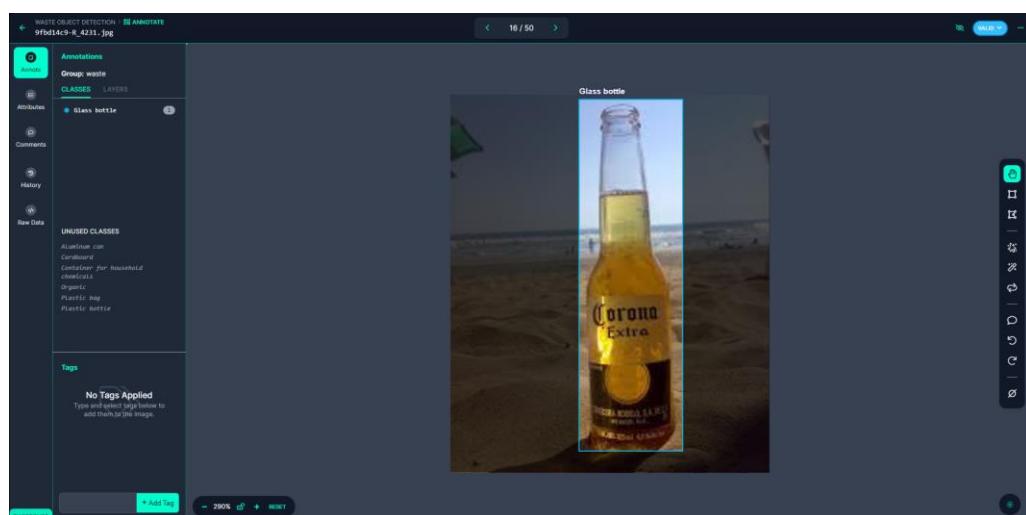
Şekil 52: Verisetindeki örnek görseller

Görüntü kümesinin Şekil 53'te gösterilen sağlık raporuna göre plastik şişe, plastik poşet, organik sınıfları diğer sınıflara göre alüminyum kutular aşırı temsil edilmişken (over represented); karton kutu, cam şişe, kimyasal içerik sınıfları yetersiz temsil edilmiş (under represented) durumdadır. Modelin başarısının artması için verisetinin sağlık durumunun iyileştirilmesi gerekmektedir. Fakat uygun çözümürlükte görüntü bulup ince detaylarla etiketleme işlemleri manuel işlemler olup çok zaman alacağından bu çalışmada bu tür bir iyileştirme yapılmayacaktır.



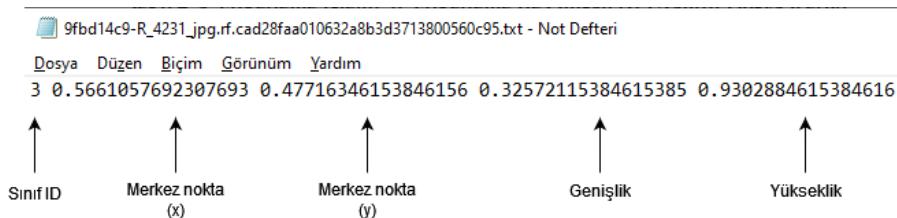
Şekil 53: Veriseti için hazırlanan görüntülerin sağlık durumu

Literatürde görsel verilerin etiketlenmesi için işletim sistemlerinde LabelImg programı yaygın olarak kullanılsa da bu çalışmada Roboflow sitesinin etiketleme platformundan ilerlemiştir. Şekil 54'te görüntünün etiketlenmesi gösterilmektedir. Veriler bu şekilde dikdörtgen şeklindeki kutuya alınarak etiketlenmektedir.



Şekil 54: Verilerin etiketlenmesi

Etiketlenen görseller için bir text dosyası oluşur. Oluşan text dosyalarında görseldeki nesnelerin index bilgisine karşılık nesnenin bulunduğu dikdörtgen şeklin merkezi konumu ve genişlik-yükseklik bilgileri tutulmaktadır. Şekil 55, Şekil 54'te etiketlenmiş olarak gösterilen görselin aynı isimdeki txt uzantılı dosyanın içeriğine yer verilmiştir.



Şekil 55: Etiketlenen görsele ait txt uzantılı dosya içeriği

Görüntü kümelerinden bir veriseti yaratmak Veri Önisleme ve Veri arttırma adımları için ise yine Roboflow sitesinden yararlanılmıştır.

Veri Önisleme (Data Preprocessing)

YOLO modelinde 360x360 ile 1024x1024 arasında çeşitli çözünürlüklerdeki görüntülerle train kümesi oluşturulabilir (Aslan & Yağımlı, İş Sağlığı ve Güvenliğinde Derin Öğrenme Tabanlı Risk Tespit ve Analizi, 2023). Tüm görseller 416x416 çözünürlüğüne yeniden boyutlandırılması sağlanmıştır. Auto-Orient ise görüntülerin yan çevirme, büyütme, küçültme gibi işlemlere maruz kaldığında nesneleri belirten etiketlerin kaymasını önlediği için aktif hale getirilmiştir. Şekil 56'de verisetine uygulanan preprocessing adımları gösterilmiştir.



Şekil 56: Veriseti preprocessing adımı

Veri Arttırma (Data Augmentation)

Veriseti miktarı küçük olduğundan verilerin eğitimi için basit veri artırma tekniği kullanılmıştır. Her ne kadar overfitting'i artırma şansı olsa da veri artırma tekniklerinin modeli başarıya sürüklendiği bilinmektedir. Veri artırma teknikleri olarak hem görüntü düzeyinde kırma (shear), dikey ve yatay çevirme (flip), pozlama (exposure); hem de sınırlayıcı kutu düzeyinde

90 derece döndürme (rotate) adımları kullanılmıştır. Şekil 57'de bu adımlara örnekler gösterilmiştir.



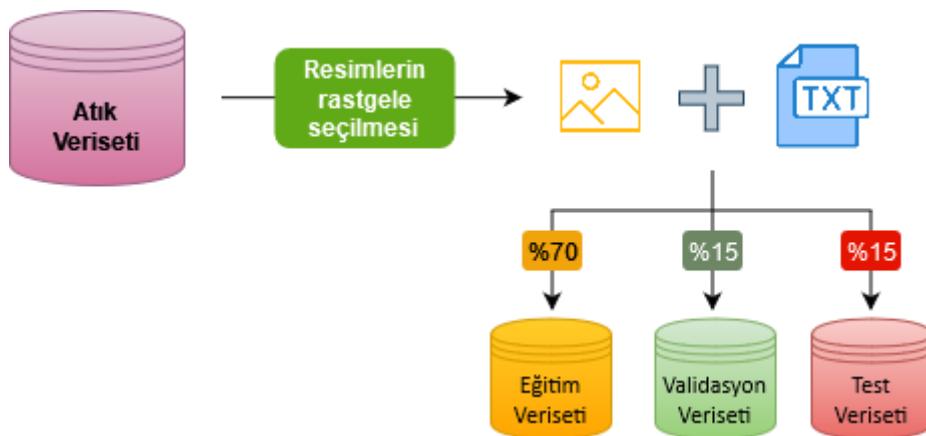
Şekil 57: Verisetinde kullanılan veri artırma teknikleri

Eğitim-Test-Validasyon Ayrımı

Verisetinde eğitim-test-validasyon ayımı yapılırken varyans etkeni göz önünde bulundurulmalıdır. Microsoft çalışanı Baheti'ye göre bu ayımı gözetirken 2 önemli endişe konusu bulunmaktadır (Baheti, 2021):

- Eğer eğitim verisi az olursa, model eğitimi yüksek varyans gösterecektir.
- Eğer test/validasyon verisi az olursa, model değerlendirmesi/model performansı yüksek varyans gösterecektir.

Bu çalışmada veriseti hazırlanırken standart sayılan %70, %15, %15 eğitim-test-validasyon ayımı uygulanmıştır. Verilerin ayrılması gösteren diyagram Şekil 58'deki gibidir.



Şekil 58: Veriseti train-validation-test ayımı

Veri Setinin Hazır Hale Getirilmesi

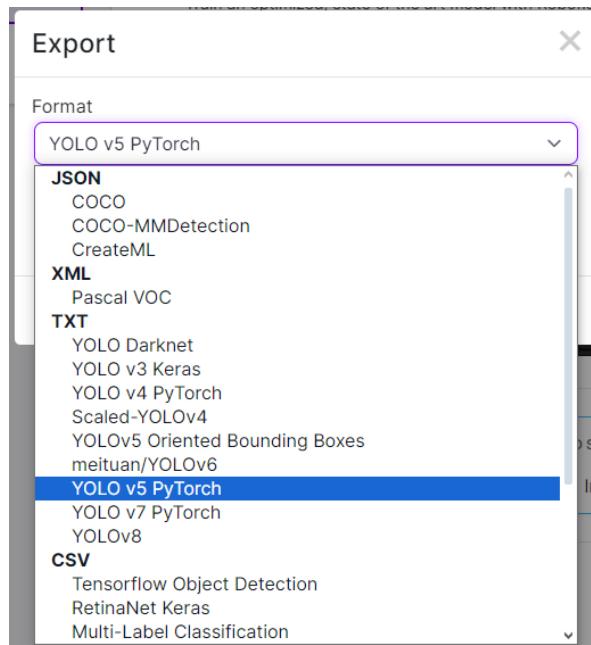
Veri önişleme, veri arttırma, ve veriseti train-valid-test ayrımı işlemlerinden sonra ise eğitim seti çoklama işlemi uygulanıp aşağıda yer alan Şekil 59'daki hesaplamalara göre verisetinde sonuç olarak 9.837 görüntü oluşmuştur.

2,867 training images × 3 variants
+ 618 validation images
+ 618 testing images

≤ 9,837 image output size

Şekil 59: Eğitim setinin çoğlanması

Derin öğrenme modellerinin kabul ettiği veriseti formatları XML, CSV, TXT gibi farklılık göstermektedir. Ancak Roboflow sitesi neredeyse her modele uygun formatlar sunduğu için esnek kullanımı mevcuttur. Bu çalışmada hazırlanan veriseti, Şekil 60'de belirtildiği gibi seçilerek kullanılacak model olan YOLOv5 ve YOLOv8 formatına uygun ZIP dosyasında export edilmiştir.



Şekil 60: Verisetinin modele uygun formatının seçilmesi

5.2.2. Başarı Metriği

Nesne tespiti modelleri ölçülürken, modelin çizdiği sınırlayıcı kutuların (bounding box) ne kadar doğru çizildiğine ve tespit edilen objenin doğruluğuna bakılır. Bu yüzden modelin başarısını hesaplamak için hata matrisi (Confusion Matrix), kesinlik (precision), hassasiyet (recall), ortalama hassasiyet doğruluğu (mAP – mean Average Precision), F1 score değerleri değerlendirilecektir.

		Gerçek Durum	
		Pozitif Durum	Negatif Durum
Tahmin	Pozitif Tahmin	TP	FP
	Negatif Tahmin	FN	TN

Tablo 2: Karmaşıklık matrisi

Tablo 2'ya göre;

TP (Gerçek Pozitif): Gerçekte pozitif durumların başarıyla pozitif olarak tahmin edilmesi,

TN (Gerçek Negatif): Gerçekte negatif durumların başarıyla negatif olarak tahmin edilmesi,

FP (Yanlış Pozitif): Gerçekte negatif durumların yanlışlıkla pozitif olarak tahmin edilmesi,

FN (Yanlış Negatif): Gerçekte pozitif durumların yanlışlıkla negatif olarak tahmin edilmesidir.

Recall, modelin gerçekten pozitif olan tüm örnekleri doğru bir şekilde tanıayıp tanımadığını ölçer. Denklem (28)'de recall formülü verilmiştir. 0 ile 1 arasında değer alır ve modelin recall çıktısı 1'e yakınsa model başarılı demektir.

$$\text{Recall } (r) = \frac{TP}{TP + FN} \quad (28)$$

Precision, modelin pozitif olarak etiketlediklerinin gerçekten pozitif olma olasılığını ölçer. Denklem (29)'de precision formülü verilmiştir. 0 ile 1 arasında değer alır ve modelin recall çıktısı 1'e yakınsa model başarılı demektir.

$$\text{Precision } (p) = \frac{TP}{TP + FP} \quad (29)$$

mAP, obje tespiti görevlerinde kullanılan metriktir. Farklı sınıflardaki nesnelerin model tarafından doğru şekilde tespit edilip edilemediğini ölçer. Ayrca 0 ile 1 arasında değer alır, bu değer ne kadar 1'e yakınsa model o kadar başarılıdır. mAP metriğinin hesaplanabilmesi için ilk olarak her sınıf için Average Precision (AP) metriğinin hesaplanması gerekmektedir. AP, Recall ve Precision eğrisinin altında kalan alana karşılık gelmektedir. Denklem (30)'de AP skorunun nasıl hesaplanacağı gösterilmiştir (Aslan & Yağımlı, İş Sağlığı ve Güvenliğinde Derin Öğrenme Tabanlı Risk Tespit ve Analizi, 2023):

$$Ortalama Hassasiyet (AP) = \int_0^1 p(r)dr \cong \sum_{k=1}^n p(k) \cdot \Delta r(k) \quad (30)$$

Denklem (31)'de mAP skorunun AP ile nasıl hesaplanacağı gösterilmiştir. (Aslan & Yağımlı, İş Sağlığı ve Güvenliğinde Derin Öğrenme Tabanlı Risk Tespit ve Analizi, 2023)

$$Genel Ortalama Hassasiyet (mAP) = \frac{\sum AP}{n} \quad (31)$$

F1 skoru, Precision ve Recall skorlarının harmonik ortalaması alınarak hesaplanmaktadır (Aslan & Yağımlı, İş Sağlığı ve Güvenliğinde Derin Öğrenme Tabanlı Risk Tespit ve Analizi, 2023). Metriğin matematiksel karşılığı Denklem(32)'de aktarılmıştır.

$$F1 = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} = 2 * \frac{(p * r)}{(p + r)} \quad (32)$$

5.2.3. Öğrenimin Transferi (Transfer Learning)

Derin Öğrenme modelleri eğitim için büyük veri kümelerine ihtiyaç duymaktadır. Ancak, her seferinde farklı görevler için uygun büyülükte veri bulmak kolay olmayabilir. Transfer Learning tekniğinde COCO gibi geniş veri kümeleri ile önceden eğitilmiş(pre-trained) stabil çalışan modellerin sahip olduğu ağırlıklar, başka modellere aktarılırak öğrenilen özellikler, desenler ve bilgiler korunur. Sınırlı veriye sahip olduğumuz durumlarda bile daha az veriye rağmen hızlı ve etkili öğrenim sürecine olanak sağlayan önemli bir stratejidir. Bu çalışmada ise verisetinin küçük olması nedeniyle “veri darboğazı” yaşıyarak derin öğrenme modeli gerçek dünya verilerinde iyi bir genelleme yapamayıp “over-fitting” eğiliminde olacaktır (Yanai & Kawano, 2015). Dolayısıyla, bu çalışmada Transfer Learning teknigi kullanılacaktır.

5.2.4. Modelin Eğitimi

Bu bölümde, 3.1. başlığında bahsedilen veriseti ZIP olarak export edildikten sonra konfigürasyon ayarlarını içeren data.yaml dosyasında verilerin dosya yolları düzenlenip Google Drive'a yüklenmiştir. Model, tespit edeceği nesnelerin miktarını ve isimlerini ilgili YAML uzantılı dosya sayesinde öğrenir. Şekil 61'de modelde kullanılan veriseti için hazırlanmış YAML dosyasının içeriği paylaşılmıştır. Verisetini YOLOv5 ve YOLOv8 mimarisinde eğitebilmek için bundan sonraki işlemlere ise Google Colab'tan devam edilmiştir.

```
*data.yaml - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 7
names: ['Aluminum can', 'Cardboard', 'Container for household chemicals', 'Glass bottle', 'Organic', 'Plastic bag', 'Plastic bottle']
```

Şekil 61: Çalışmada kullanılan YAML dosyasının içeriği

Özellikle makine öğrenimi ve veri analizi gibi yapay zeka alanlarında yüksek bilgisayar gücüne ihtiyaç duyulmaktadır. Colab, tarayıcı üzerinden erişilebilen ve Python programlama dilini destekleyen bir platformdur. İlgili alanlarda çalışmak isteyen kullanıcılar için sanal ortamda güçlü GPU, RAM ve CPU kaynaklarına ücretsiz ve ücretli erişim sağlayarak etkili bir çalışma ortamı sunar. İlk olarak Şekil 62'deki gibi colab sistemine drive ortamı ve veriseti yüklenmiştir.

```
✓  Setup
Clone repo, install dependencies and check PyTorch and GPU.

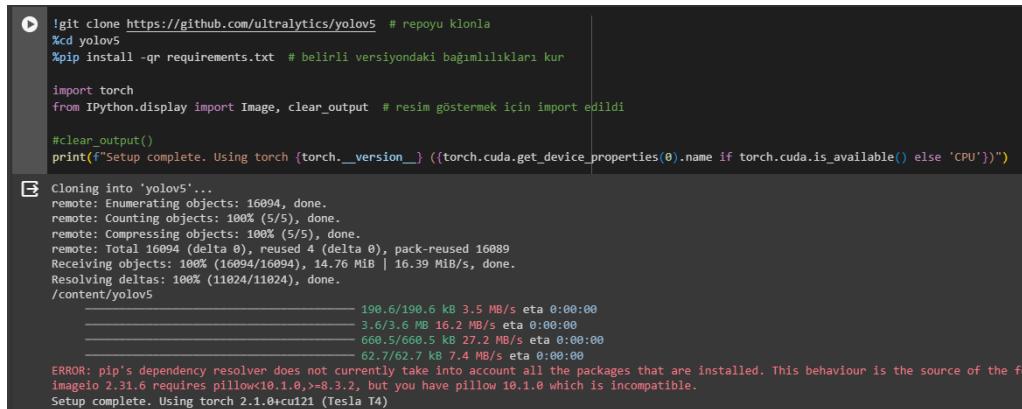
[2] from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive

[5] !unzip -q "/content/drive/MyDrive/DeepLearning/Waste Object Detection.v11.yolov5pytorch.zip" -d "/content/data"
```

Şekil 62: Çalışma Ortamına Drive'in bağlanması ve verisetinin açılması

Çalışmadaki modelin eğitiminde Pytorch kütüphanesi (torch) kullanılmıştır. YOLO modellerinin alınacağı Ultralytics Github Repository'si import edilmiştir. Colab çalışma ortamı ise GPU ile başlatılmıştır. Colab'de hesaplamalar için kullanılan cihazın CPU mu yoksa GPU'mu olduğunu Şekil 63'deki torch kütüphanesine ait fonksiyon sayesinde bilinmektedir. Ayrıca model, Yapay zeka araştırmalarında sıkılıkla tercih edilen yöntem olan grafik işlemci birimi üzerinde genel amaçlı hesaplama (General Purpose Computation on Graphic Processing Units) (GPGPU) yöntemi ile Compute Unified Device Architecture (CUDA) mimarisi kullanılarak eğitilmiştir. Böylece algoritmaların çalışma ağırlıkları en aza indirilmektedir. Bunun nedeni ise ilk çıktı

zamanlar 3D görselleştirme (rendering) görevi için kullanılan GPU'ların, CPU'lardan kat kat fazla transistörden oluştuğu için sahip oldukları paralel işleme yetenekleri sayesinde aritmetik işlemleri çok hızlı yapabilmesi ve hafıza band genişliği hızının fazla olduğunun anlaşılırak yüksek hesaplama gücü isteyen araştırmalar için araştırmacıların gözdesi olmalıdır (Şahin & Külür, 2016).



```

git clone https://github.com/ultralytics/yolov5 # repoyu klonla
cd yolov5
pip install -qr requirements.txt # belirli versiyondaki bağımlılıkları kur

import torch
from IPython.display import Image, clear_output # resim göstermek için import edildi

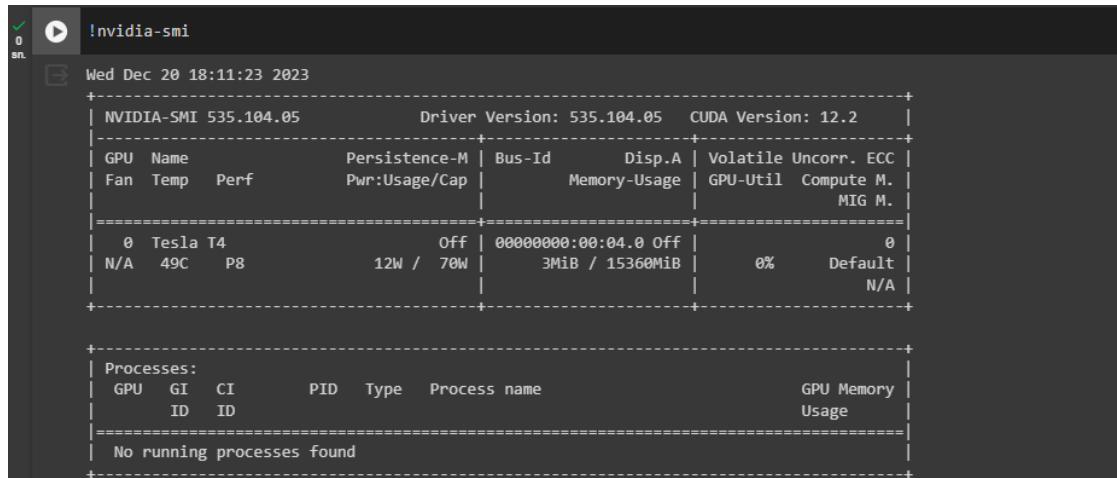
#clear_output()
print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")

Cloning into 'yolov5'...
remote: Enumerating objects: 16094, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 16094 (delta 0), reused 4 (delta 0), pack-reused 16089
Receiving objects: 100% (16094/16094), 14.76 MB | 16.39 MB/s, done.
Resolving deltas: 100% (11024/11024), done.
/content/yolov5
          190.6/190.6 kB 3.5 MB/s eta 0:00:00
          3.6/3.6 MB 16.2 MB/s eta 0:00:00
          660.5/660.5 kB 27.2 MB/s eta 0:00:00
          62.7/62.7 kB 7.4 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the fo
imageio 2.31.6 requires pillow<10.1.0,>=8.3.2, but you have pillow 10.1.0 which is incompatible.
Setup complete. Using torch 2.1.0+cu111 (Tesla T4)

```

Şekil 63: Çalışma Ortamına Github'ın klonlanması ve GPU'nun bağlanması

Şekil 64'te de gözüktüğü üzere model eğitimi için GPU olarak Colab'ın ücretsiz olarak kullanıcılaraya tahsis ettiği 16 GB DDR6 Ram'e sahip Nvidia Tesla T4 grafik birimi kullanılmıştır.



```

!nvidia-smi
Wed Dec 20 18:11:23 2023
+-----+-----+-----+
| NVIDIA-SMI 535.104.05      Driver Version: 535.104.05    CUDA Version: 12.2 |
+-----+-----+-----+
| GPU  Name        Persistence-M  Bus-Id      Disp.A  | Volatile Uncorr. ECC | | | | |
| Fan  Temp  Perf  Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
|          |          |             |           |           | MIG M.   |
+-----+-----+-----+
|  0  Tesla T4           Off  00000000:00:04.0 Off   |                  0 | | | | |
| N/A   49C   P8    12W /  78W |      3MiB / 15360MiB |     0%      Default |
|          |          |             |           |           | N/A      |
+-----+-----+-----+
+-----+-----+
| Processes:                               GPU Memory |
| GPU  GI  CI   PID  Type  Process name        Usage  |
| ID  ID
+-----+-----+
| No running processes found
+-----+

```

Şekil 64: Eğitimde kullanılan GPU özellikleri

Transfer learning teknigiyle önceden eğitilmiş yolov5m modeli, bu çalışmada kullanılan veri seti için data.yaml dosyasına göre eğitilmiştir. Eğitim, görüntüler 416x416 boyutunda olduğu için img değeri 416, batch 32, eğitim adım sayısı yani epok (epoch) 1000 ayarlarıyla başlatılmıştır.

- **Epok:** Modelin train verilerinin hepsini gördüğü eğitim sürecidir. Yüksek miktarda epok, aşırı öğrenmeye neden olabilmektedir.
- **Batch:** Modelin her bir iterasyonda aynı anda işleyeceği veri miktarını belirtir. Bu parametre için 16, 32 ve 64 gibi değerler tercih edilir. Fakat, batch ne kadar yüksek seçilirse bellek ihtiyacı o kadar fazla olacaktır.

Her epokta elde edilen en iyi ağırlıklar (best.pt) ve son ağırlıklar (last.pt) Google Drive'da Runs klasöründe kaydedilmiştir. Şekil 65'te modelin eğitimi için kullanılan script verilmiştir.

```
import datetime
now = datetime.datetime.now()

strProjectNamePath = "/content/drive/MyDrive/DeepLearning/Project_Yolov5/Runs"
strRunPath = "Run_" + now.strftime("%m-%d-%Y_%H-%M-%S")
strFullRunPath = str(strProjectNamePath + "/" + strRunPath)
strBestRunWeights = strFullRunPath + "/weights/best.pt"

!python train.py --img 416 --batch 32 --epochs 1000 --data data.yaml --weights yolov5m.pt --project $strProjectNamePath --name $strRunPath --cache
```

Şekil 65: YOLOv5m modelinin eğitimi

YOLOv5m modelinin hiperparametreleri, katman yapısı ve toplam ağırlıkları gibi genel bilgileri Şekil 66'da gösterilmiştir.

```
hyperparameters: lr=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0,
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 runs in Comet
TensorBoard: Start with 'tensorboard --logdir /content/drive/MyDrive/DeepLearning/Project_Yolov5/Runs', view at http://localhost:6006
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5m.pt to yolov5m.pt...
100% 755k/755k [00:00:00.00, 17.6MB/s]
100% 40.8M/40.8M [00:00:00.00, 216MB/s]

Overriding model.yaml nc=80 with nc=7

      from    n      params   module           arguments
  0       -1  1      5288  models.common.Conv  [3, 48, 6, 2, 2]
  1       -1  1     41664  models.common.Conv  [48, 96, 3, 2]
  2       -1  2     65288  models.common.C3   [96, 96, 2]
  3       -1  1     166272  models.common.Conv  [96, 192, 3, 2]
  4       -1  4     444672  models.common.C3   [192, 192, 4]
  5       -1  1     664320  models.common.Conv  [192, 384, 3, 2]
  6       -1  6     2512896 models.common.C3  [384, 384, 6]
  7       -1  1     2655744 models.common.Conv  [384, 768, 3, 2]
  8       -1  2     4134912 models.common.C3  [768, 768, 2]
  9       -1  1     1476864 models.common.SPPF  [768, 768, 5]
 10      -1  1     295688 models.common.Conv  [768, 384, 1, 1]
 11      -1  1          0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
 12     [-1, 6]  1      1182720 models.common.Concat [1]
 13      -1  2     1182720 models.common.C3   [768, 384, 2, False]
 14      -1  1      74112  models.common.Conv  [384, 192, 1, 1]
 15      -1  1          0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
 16     [-1, 4]  1      1182720 models.common.Concat [1]
 17      -1  2     296448 models.common.C3   [384, 192, 2, False]
 18      -1  1     332160 models.common.Conv  [192, 192, 3, 2]
 19     [-1, 14] 1      1182720 models.common.Concat [1]
 20      -1  2     1035264 models.common.C3   [384, 384, 2, False]
 21      -1  1     1327872 models.common.Conv  [384, 384, 3, 2]
 22     [-1, 10] 1          0 models.common.Concat [1]
 23      -1  2     4134912 models.common.C3   [768, 768, 2, False]
 24    [17, 28, 23] 1     48492  models.yolo.Detect [7, [[18, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [192, 384, 768]]

Model summary: 291 layers, 20895564 parameters, 20895564 gradients, 48.3 GFLOPs
```

Şekil 66: YOLOv5m modelinin genel yapısı

Modellerin katman miktarı ve ağırlık miktarları Tablo 3'te verilmiştir. Yolo5m modeli, yaklaşık olarak toplam 21 milyon ağırlık içерirken YOLOv8m modeli toplam 26 milyon ağırlık içermektedir.

Model	Katman Miktarı	Ağırlık Miktarı
YOLOv5m	291	~21 Milyon
YOLOv8m	295	~26 Milyon

Tablo 3: Modellerin katman ve ağırlık miktarları

Hiperparametre detayları ise Tablo 4'te verilmiştir. Her iki model için de öğrenme oranı (lr) 0.01, momentum 0.937, weight decay 0.0005 ve optimizer ise SGD seçilmiş olup değişiklikle gidilmemiştir.

Model	Optimizer	Learning Rate	Momentum	Weight Decay
YOLOv5m	SGD	0.01	0.937	0.0005
YOLOv8m	SGD	0.01	0.937	0.0005

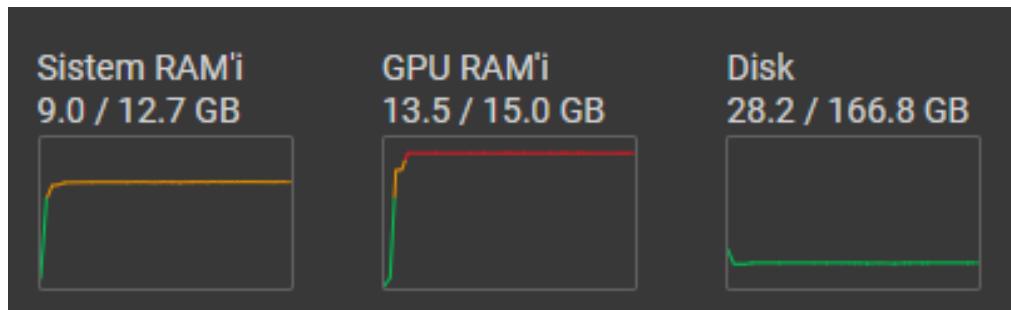
Tablo 4: Modellerin optimizer ve hiperparametreleri

Her bir epok her iki model için de yaklaşık 2 dakikada tamamlanmış olup, her epokta validasyon kümelerinden elde edilen sınırlayıcı kutu, obje ve sınıf kayıp değerleri ile mAP50 ve mAP50-95 ortalama hassasiyet başarısı bastırılmıştır. Şekil 67'de YOLOv5 modelinin eğitim anına ait bir görüntü verilmiştir.

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
3/999	6.25G	0.04414	0.02254	0.02232	85	416: 100% 269/269 [01:46<00:00, 2.54it/s]
	Class all	Images	Instances	P	R	mAP50 mAP50-95: 100% 10/10 [00:06<00:00, 1.46it/s]
		618	814	0.673	0.598	0.624 0.375
4/999	6.25G	0.04216	0.02226	0.02136	85	416: 100% 269/269 [01:44<00:00, 2.57it/s]
	Class all	Images	Instances	P	R	mAP50 mAP50-95: 100% 10/10 [00:06<00:00, 1.45it/s]
		618	814	0.554	0.528	0.567 0.349
5/999	6.25G	0.04067	0.02217	0.0192	66	416: 100% 269/269 [01:44<00:00, 2.56it/s]
	Class all	Images	Instances	P	R	mAP50 mAP50-95: 100% 10/10 [00:06<00:00, 1.56it/s]
		618	814	0.734	0.687	0.721 0.449
6/999	6.25G	0.03973	0.02175	0.01806	79	416: 100% 269/269 [01:45<00:00, 2.55it/s]
	Class all	Images	Instances	P	R	mAP50 mAP50-95: 100% 10/10 [00:06<00:00, 1.45it/s]
		618	814	0.775	0.61	0.681 0.422
7/999	6.25G	0.03848	0.02142	0.01777	79	416: 100% 269/269 [01:45<00:00, 2.55it/s]
	Class all	Images	Instances	P	R	mAP50 mAP50-95: 100% 10/10 [00:06<00:00, 1.59it/s]
		618	814	0.757	0.744	0.776 0.483
8/999	6.25G	0.03753	0.02094	0.01619	92	416: 100% 269/269 [01:44<00:00, 2.58it/s]
	Class all	Images	Instances	P	R	mAP50 mAP50-95: 100% 10/10 [00:06<00:00, 1.52it/s]
		618	814	0.641	0.704	0.66 0.381
9/999	6.25G	0.03739	0.02108	0.01616	112	416: 72% 193/269 [01:14<00:33, 2.26it/s]

Şekil 67: YOLOv5m eğitimi anından bir görüntü

Eğitim sırasında GPU kullanımı ise neredeyse maksimum seviyede seyretmiştir. Şekil 68'de kullanılan kaynakların durumu paylaşılmıştır.



Şekil 68: Eğitim sırasında kullanılan kaynakların durumu

Doğruluk değerleri belli bir aralıkta sürekli salınım göstermeye başlayıp üstüne validasyon kayıplarının da azalmak yerine giderek artması nedeniyle 125 epoktan sonra eğitime devam edilmemiştir. Eğitim sonunda, test kümelerindeki görüntüler üzerinde doğruluk kontrolü yapılmıştır. Şekil 69'da ise modelin bu çıktıları paylaşılmıştır.



Şekil 69: Modelin değerlendirdiği validasyon görüntülerleri ve güven skorları

5.3. Araştırma Sonuçları

5.3.1. Model Kayıpları

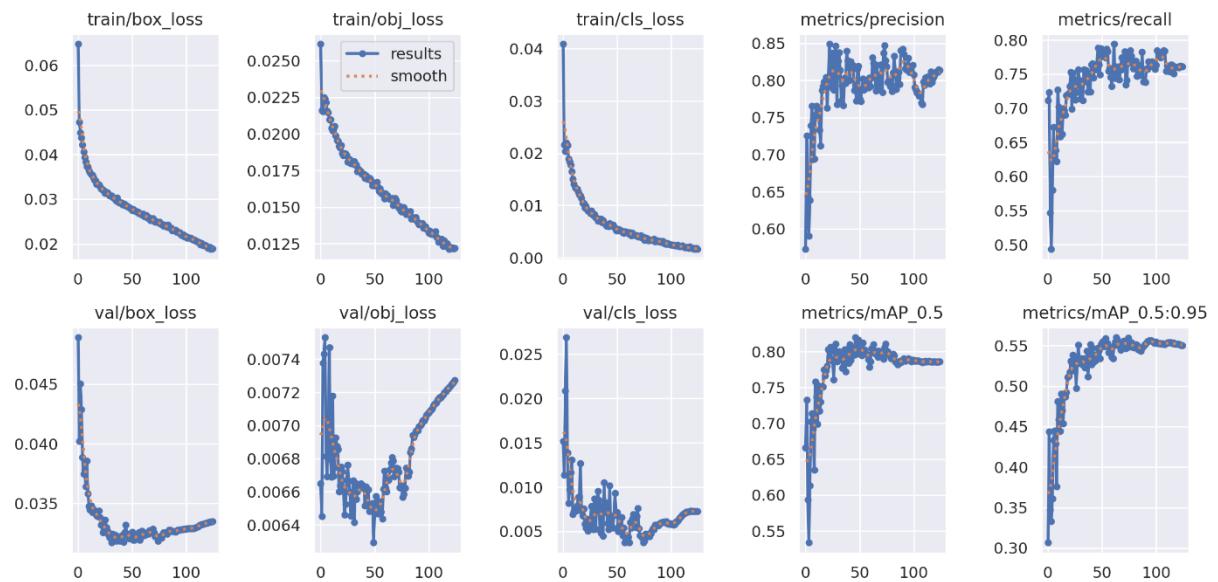
Modelin obje tespiti görevi için eğitim süresince her epok sonunda loss, precision, recall ve mAP değerlerini hesaplar. Şekil 70 ve Şekil 71'de YOLOv5 ve YOLOv8 modelleri için 125 epokluk eğitim sonucundaki performans çıktıları paylaşılmıştır. Performans çıktılarına göre ise kayıp değerleri aşağıdaki gibi yorumlanmıştır. Ek olarak, epoklarda eğer train kayıpları azalırken validasyon kayıpları artıyorsa modelin over-fitting yapmaya başladığı bilinir. Yorumlama bu bilgi ışığında yapılmıştır.

Sınırlayıcı Kutu Kaybı (box_Loss): Etiketleme sırasında manuel biçimde çizilen sınırlayıcı kutular ile modelin kendi oluşturduğu sınırlayıcı kutularının karşılaştırıldığında doğan hatayı temsil eder ve kutu kaybı MSE ile ölçülür. Eğitim süreci boyunca bu hatalar düzelttilir, ve böylece model doğru sınırlama kutuları oluşturmayı öğrenir. Her iki modelin de performans çıktıları incelendiğinde train ve doğrulama değerlerinin yaklaşık olduğu, fakat yaklaşık 50. epok sonrasında train kayıpları azalırken validasyon kayıplarının arttığı gözlemlenmiştir.

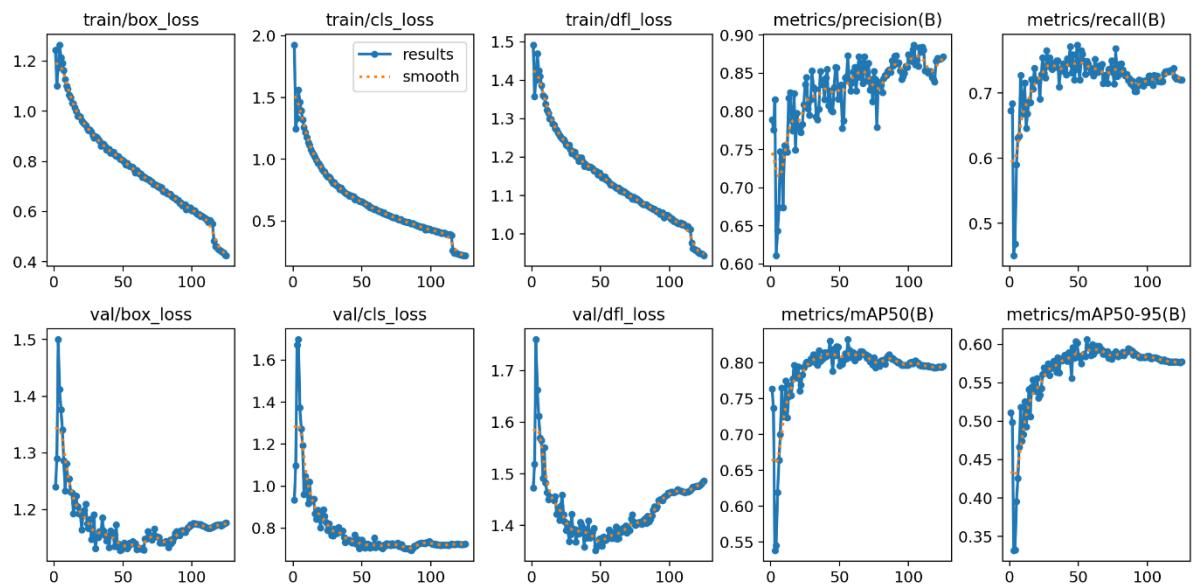
Sınıflandırma Kaybı (cls_loss): Modelin tahmin ettiği sınıf ile gerçek etiket arasındaki uyumsuzluğu ölçer. Ölçüm, her bir öngörülen sınırlayıcı kutunun güven skorları için Binary Cross-Entropy fonksiyonuna dayalı olarak hesaplanır (Jocher, Are class and box losses calculated the same in YoloV8 and YoloV5?, 2023). Düşük bir sınıflandırma kaybı, modelin doğru tahminler yaptığını ve kaybın düşük olduğunu gösterirken, yüksek bir sınıflandırma kaybı, modelin tahminlerinin yanlış olduğunu ve kaybın yüksek olduğunu gösterir. Bu çalışmada eğitilen YOLOv5 modelinin sınıflandırma kaybı train dasası için sürekli düşerken doğrulama kümesi için yaklaşık 75. Epoktan sonra arttığı gözlemlenmiştir. YOLOv8 modelinde ise doğrulama kümesi için aynı oranda artış yaşanmamıştır.

Nesne Kaybı (obj_loss): YOLOv5 modelinde bu kayıp bileşeni kullanılırken; nesne varlığına bağlı olasılıklar, YOLOv8 modelinde daha iyi entegre bir şekilde işlendiği için kullanılmamaktadır (Yocher, 2023). YOLOv5 modelinin doğrulama verisinde 50.epoktan sonra ciddi artış yaşandığı not edilmiştir.

Dağılım Odak Kaybı (df1_loss): Bu kayıp fonksiyonu, sınıf dengesizliğinden doğan sorunları ele alır. Kümedeki nadir örneklerin etkisini, kolay örneklerin ağırlığını azaltarak vurgular (Yocher, 2023). YOLOv8 modelinde kullanılan kayıp türüdür. Yaklaşık olarak 60. epokta doğrulama kümesindeki verilerde bu hata değeri artışa geçmiştir.



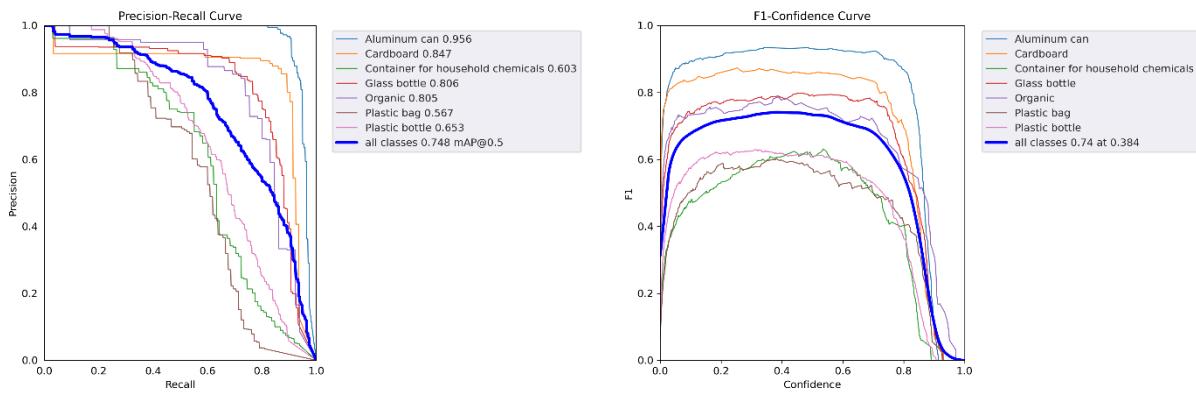
Şekil 70: YOLOv5 Modelinin 125 Epokluk Eğitim Performans Çıktıları



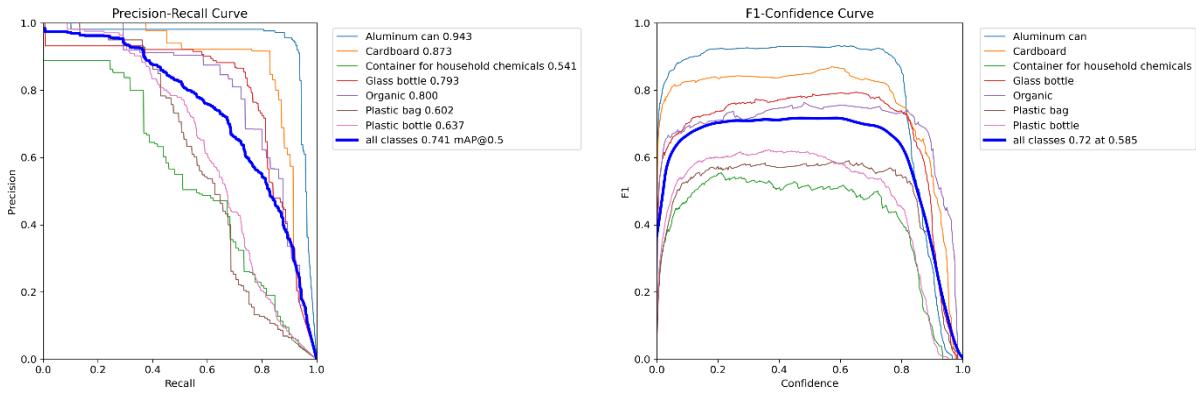
Şekil 71: YOLOv8 Modelinin 125 Epokluk Eğitim Performans Çıktıları

5.3.2. mAP ve F1 Skorları

Modellerin taze test verileri için Precision-Recall eğrisi ve F1-Confidence eğrisi Şekil 72 ve Şekil 73'e göre irdelenmiştir. Precision-Recall eğrisi için konuşacak olursak; organik, plastik poşet, ve plastik şişe nesneleri için her iki modelde de ortalamanın aşağısında mAP skoru elde edilmiş. F1-Confidence eğrisi grafiği en iyi performansın hangi güven skoru eşliğinde elde edildiğini belirtir. Düşük güven eşik değeri aşırı kabul, yüksek güven eşiği ise aşırı ret demektir. YOLOv5m modelinin güven eşik değeri 0.384 iken %74, YOLOv8m modelinde ise 0.585 güvenlik eşik değerinde %72 F1 skoru elde edilmiştir.



Şekil 72: YOLOv5 modelinin Test kümlesi için Precision-Recall Eğrisi ve F1-Confidence Eğrisi Çıktıları



Şekil 73: YOLOv8 modelinin Test kümlesi için Precision-Recall Eğrisi ve F1-Confidence Eğrisi Çıktıları

5.3.3. Sınıf Tespiti

YOLO modellerinde mAP skoru için hem mAP50 hem de mAP50-95 performansına bakılmaktadır. mAP50 0.5 güven eşiği; mAP50-95 ise 0.5 ile 0.95 arasında kalan güven (confidence) eşiği ile elde edilen performansı belirtmektedir. Tablo 5 ve Tablo 6'da sırasıyla YOLOv5m ve YOLOv8m modellerinin test çıktıları paylaşılmıştır. Tüm sınıflar için YOLOv5m'in mAP50 skoru 0.748 iken YOLOv8m'in mAP skoru 0.741'dir. Fakat mAP50-95 için YOLOv8m'in 0.524 skoru ile 0.51 skor alan YOLOv5m'i geçtiği gözlenmiştir. Her iki metrik için model arası farklar çok küçük olsa da YOLOv8m'in yüksek güven eşiğinde daha iyi sonuç verdiği ortaya çıkmıştır.

Nesne	Örnekler	P	R	mAP50	mAP50-95
Tümü	1071	0.785	0.703	0.748	0.51
Aluminum can	196	0.962	0.904	0.956	0.642
Cardboard	93	0.877	0.845	0.847	0.642
Glass bottle	150	0.794	0.772	0.806	0.586
Organic	65	0.847	0.723	0.805	0.592
Plastic bag	105	0.681	0.529	0.567	0.35
Plastic bottle	364	0.717	0.542	0.653	0.391
Container for household chemicals	98	0.618	0.602	0.603	0.363

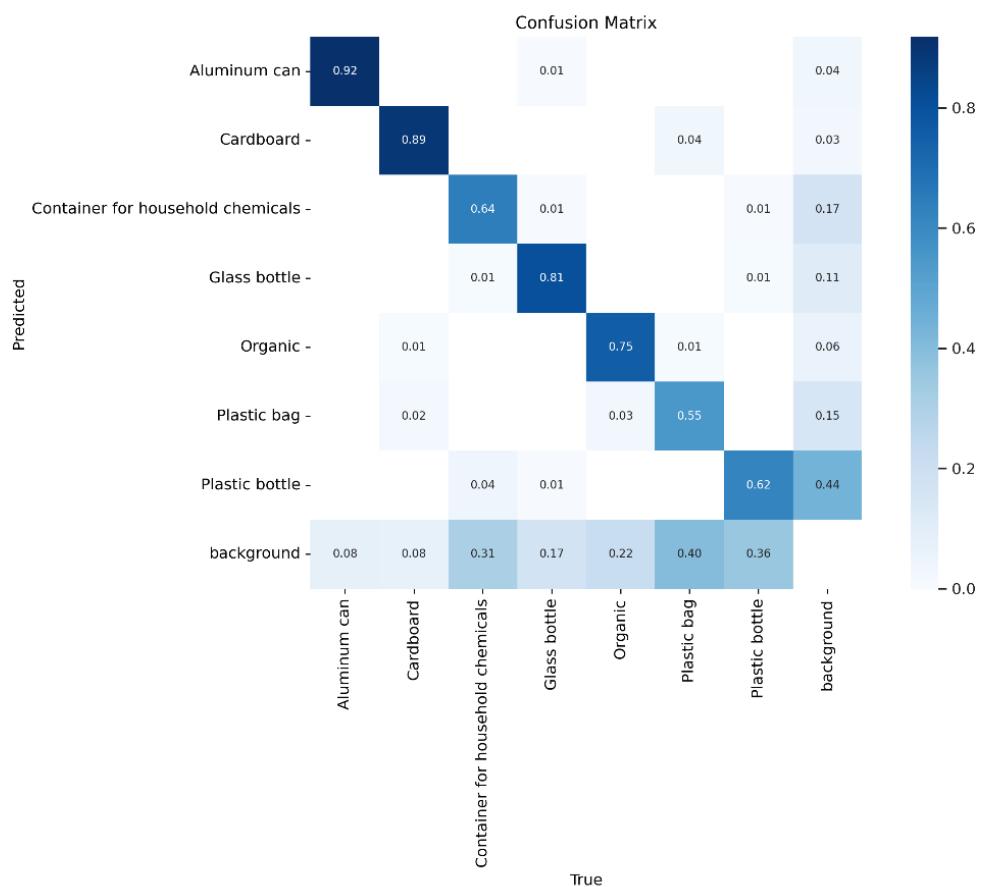
Tablo 5: Sınıfların YOLOv5m modeline göre test başarısı

Nesne	Örnekler	P	R	mAP50	mAP50-95
Tümü	1071	0.798	0.659	0.741	0.524
Aluminum can	196	0.933	0.924	0.943	0.662
Cardboard	93	0.909	0.828	0.873	0.686
Glass bottle	150	0.844	0.74	0.793	0.587
Organic	65	0.823	0.692	0.8	0.61
Plastic bag	105	0.693	0.505	0.602	0.386
Plastic bottle	364	0.78	0.478	0.637	0.396
Container for household chemicals	98	0.602	0.449	0.541	0.342

Tablo 6: Sınıfların YOLOv8m modeline göre test başarısı

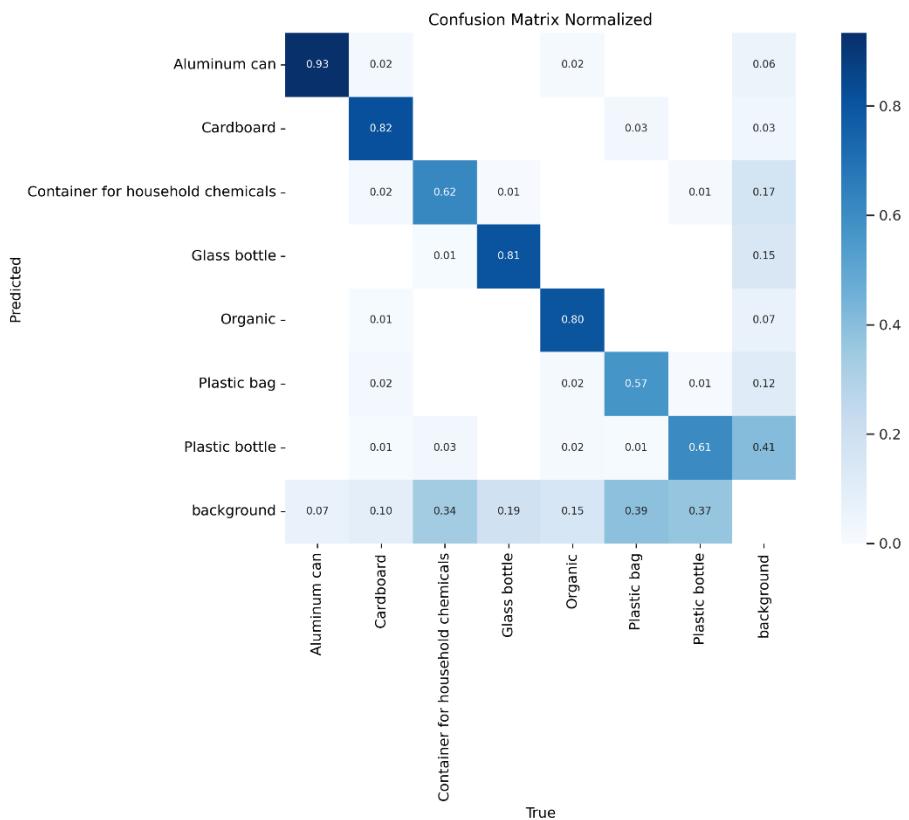
5.3.4. Karmaşıklık Matrisi

YOLOv5m modelinin eğitimi sonrasında test verilerine ait normalize edilmiş karmaşıklık matrisi Şekil 74'de verilmiştir. İlgili karmaşıklık matrisi incelendiğinde, aluminyum kutuların %92, karton kutuların %89, cam şişelerin %81 ve organik objelerin %75 tespit oranıyla doğru tahmin edilebildiği ortaya konmuştur. Fakat plastik poşetlerin %55, kimyasal içeriklerin %64, ve plastik şişelerin %62 gibi düşük tespit oranlarıyla tahmin edilebildiği gözlenmiştir.



Şekil 74: YOLOv5 modelinin normalized edilmiş karmaşıklık matrisi

YOLOv8m modelinin eğitimi sonrasında test verilerine ait normalize edilmiş karmaşıklık matrisi Şekil 75'de verilmiştir. İlgili karmaşıklık matrisi incelendiğinde, aluminyum kutuların %93, karton kutuların %82, cam şişelerin %81 ve organik objelerin %80 tespit oranıyla doğru tahmin edilebildiği ortaya konmuştur. Fakat plastik poşetlerin %57, kimyasal içeriklerin %62, ve plastik şişelerin %61 gibi düşük tespit oranlarıyla tahmin edilebildiği gözlenmiştir.



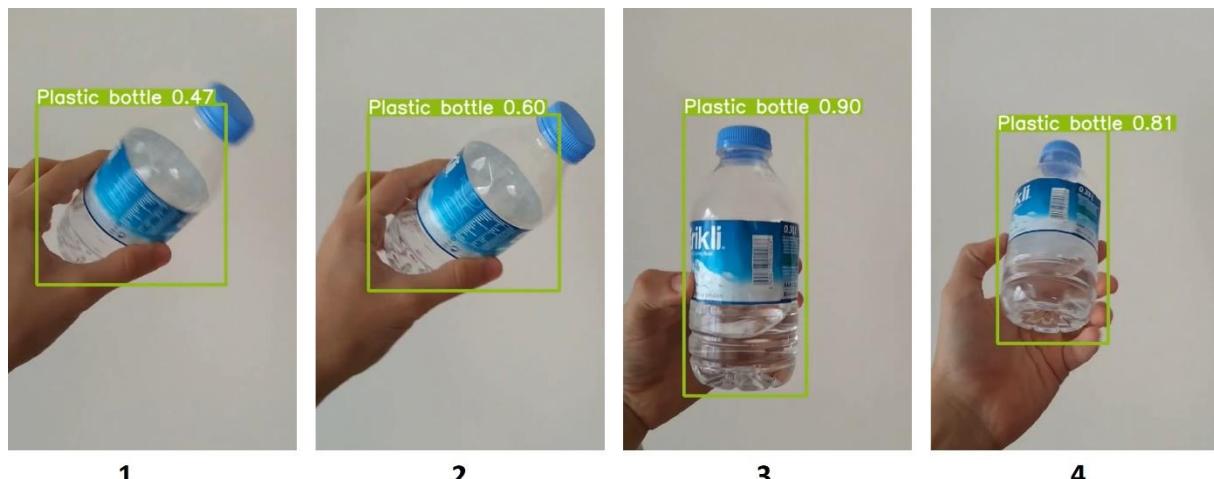
Şekil 75: YOLOv8 modelinin normalized edilmiş karmaşıklık matrisi

Eğer model bir örnekten herhangi bir çıkarım yapamazsa, o veriyi sadece bir arka plan (background) olarak ele alır. İki modelin oranlarındaki düşüklüğün ise nesneler arası benzerlikten ziyade modelin test verisinde yer alan nesneler için bir çıkarım yapamayıp onları arka plan olarak tahmin etmesinden kaynaklanmıştır.

BÖLÜM 6.

SONUÇ

Bu çalışmada gerçek zamanlı obje tespiti için konvolüstonel sinir ağları (CNN) temelli 2 farklı derin öğrenme modeli kullanılarak atık tespiti eğitimi yapılmıştır. Pytorch kütüphanesi destekli Ultralytics kütüphanesinden yararlanılarak YOLOv5 ve YOLOv8 algoritmaları kullanılmıştır. Bu modellerin eğitimi 7 farklı atık nesnesine ait toplam 4,103 görselden oluşan veri kümesi üzerinde %70-%15-%15 train-validate-test oranlarıyla gerçekleştirilmiştir. Modeller eğitime önceden eğitilmiş ağların ağırlıkları ile başlatılmıştır. En sonunda train ve test verileri üzerinde doğruluk (accuracy), hassasiyet (precision), ve ortalama hassasiyet (mAP) metrikleri doğrultusunda karşılaştırılmıştır. Karşılaştırma sonucuna göre YOLOv5m modelinin mAP50 skoru %74.8, YOLOv8m modelinin mAP50 skoru ise %74.1 sağladığı gözlenmiştir. YOLOv5m model kullanılarak yapılan uygulamada gerçek zamanlı görüntüde farklı açılarda plastik şişe nesnesi test edilmiştir. İlgili nesnenin sınırlayıcı kutu içerisinde olduğu ve güven skorunu da ekrana bastırıldığı gözlenmiştir. Teste ait farklı zamana ait görüntüler Şekil 76'da verilmiştir.



Şekil 76: Gerçek videoya ait farklı açılardan tespit edilen nesnenin güven skorları

6.1. Tartışma

Bu projede nesne tanıma modelini geliştirme çabaları sırasında, başarı yüzdesini artırmak için çalışırken, bazı noktalar göze çarpmıştır. Gelecek çalışmalarda bu noktaların göz önünde bulundurulması önemlidir.

1. Literatür incelendiğinde veri kümelerindeki görüntülerin elde edildiği kaynağın özelliği (kameranın megapikseli), kaynağın hareketliliği (IHA gibi) ve ortamın değişkenliği (karlı, yağmurlu, sualtı vs.) önem teşkil ettiği anlaşılmıştır.
2. Nesnelerin farklı açı perspektifleri modelin nesne doğruluğunu etkilemektedir. Herhangi bir görev için ihtiyaç olan her açıdan çeşitli perspektiflere sahip görsel örneklerin eğitilmesi gerekebilir.
3. Kullanılan veri kümelerinin genişletilmesi, modellerin genel biçimde daha başarılı olmasına katkıda bulunabilir.
4. Veri kümelerindeki görsellerin gerçek çözünürlükleri küçüktü, daha yüksek çözünürlklere sahip görsellerin kullanılması, modelin daha keskin ve doğru sonuçlar elde etmesine yardımcı olabilir, ancak bu durumda hesaplama maliyetinin artabileceğinden çıkarım süresinin uzayarak gecikme artışının yaşanabileceği unutulmamalıdır.
5. Önceden hazırlanmış veri kümelerindeki görsellerin etiketlemeleri bazı veriler üzerinde doğru gözükmüyor. Etiketlemelerin nesneleri tam içerecek şekilde yapılması başarayı artırabilir.
6. Veri kümelerindeki bazı nesneler az, bazı nesneler ise aşırı temsil edilmiştir. Her nesneden benzer miktarda örnek bulundurmanın modelin genel başarısını artıracağı düşünülmektedir.
7. Hyperparameter fine-tuning tekniğinden yaralanılmamıştır. Teknik modellere uygulansayıdı farklı sonuçlar elde edilebilirdi. Fakat verisetinin çok kaliteli olmaması nedeniyle modellerin yine de benzer performans sergileyeceği düşünülmektedir.
8. Araştırmalar esnasında bu çalışmada kullanılan modeller için bazı hız artıracı formatların önerildiği görülmüştür. Örneğin, CPU çıkarım hızının 3 kat artması için ONNX formatı, GPU çıkarım hızının 5 kat artması için TensorRT formatı tavsiye edilmektedir (Jocher, 2023).
9. YOLOv5 ve YOLOv8 algoritmaları için farklı ağırlık miktarına sahip n,s,m,l,x modelleri bulunmaktadır. Bu çalışmada başarı oranı ve çıkarım hızının dengeli olması açısından m modelleri ile ilerlenmiştir. X modeli ile gerçekten daha yüksek mAP skoru aldığı daha fazla araştırmalar sırasında gözlenmiştir.

KAYNAKÇA

- Abdelmalek, B., Ahmed, K., & Amine, T. M. (2019). A Survey on Lightweight CNN-Based Object Detection Algorithms for Platforms with Limited Computational Resources. *International Journal of Informatics and Applied Mathematics*, 28-44.
- Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 717-727.
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 1-74.
- Anding, K., Haar, L., Polte, G., Walz, J., & Notni, G. (2019). Comparison of the performance of innovative deep learning and classical methods of machine learning to solve industrial recognition tasks. *Joint TC1 - TC2 International Symposium on Photonics and Education in Measurement Science 2019*. Jena: Spie.
- Artificial Neural Network Tutorial*. (tarih yok). Javatpoint:
<https://www.javatpoint.com/artificial-neural-network> adresinden alındı
- Aslan, Z. (2018). ON THE USE OF DEEP LEARNING METHODS ON MEDICAL IMAGES. *The International Journal of Energy & Engineering Sciences*, 1-15.
- Barla, N. (2023, Nisan 25). *Self-Driving Cars With Convolutional Neural Networks (CNN)*. Neptune: <https://neptune.ai/blog/self-driving-cars-with-convolutional-neural-networks-cnn> adresinden alındı
- Brownlee, J. (2019, January 16). *A Gentle Introduction to Batch Normalization for Deep Neural Networks*. Machine Learning Mastery:
<https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/> adresinden alındı
- Carpenter, K. A., Cohen, D. S., Jarrell, J., & Huang, X. (2018). Deep learning and virtual drug screening. *Future Medical Chemistry*, 2557–2567.
- Cayla, B. (2021, Mart 7). *The Stochastic Gradient Descent (SGD) & Learning Rate*. Aishelf: <http://aishelf.org/sgd-learning-rate/> adresinden alındı
- Das, S., Dey, A., Pal, A., & Roy, N. (2015). Applications of Artificial Intelligence in Machine Learning: Review and Prospect. *International Journal of Computer Applications*, 31-41.
- Doğan, Ö. (2020, Kasım 26). *CNN (Convolutional Neural Networks) Nedir?* Teknoloji: <https://teknoloji.org/cnn-convolutional-neural-networks-nedir/> adresinden alındı
- Dongare, A., Kharde, R., & Kachare, A. D. (2012). Introduction to Artificial Neural Network. *International Journal of Engineering and Innovative Technology*, 189-194.
- Forjan, J. (2021, Mart 6). *Overfitting and Methods of Addressing it*. AnalystPrep:
<https://analystprep.com/study-notes/cfa-level-2/quantitative-method/overfitting-methods-addressing/> adresinden alındı
- French, R. M. (2000). The Turing Test: the first 50 years. *Trends in Cognitive Sciences*, 115-122.
- Gandharv, K. (2022, Haziran 29). *Top 5 applications of Convolution Neural Network*. Indiaai: <https://indiaai.gov.in/article/top-5-applications-of-convolution-neural-network> adresinden alındı

- Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 2627-2636.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Güzel, K. (2018, 7 20). *Geri Yayımlı Çok Katmanlı Yapay Sinir Ağları-2*. Medium: <https://kadirguzel.medium.com/geri-yayimli-cok-katmanli-yapay-sinir-aglari-2-6a47b4f3a6c> adresinden alındı
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.
- Høye, T. T., Ärje, J., Bjerge, K., Hansen, O. L., Iosifidis, A., Leese, F., Mann, H. M. R., Meissner, K., Melvad, C., Raitoharju, J. (2021). Deep learning and computer vision will transform entomology. *Proceedings of the National Academy of Sciences*, 1-10. doi:<https://doi.org/10.1073/pnas.2002545117>
- İnik, Ö., & Ülker, E. (2017). Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, 85-104.
- Karimi, G. (2021, Nisan 15). *Introduction to YOLO Algorithm for Object Detection*. Section: <https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/> adresinden alındı
- Kattenborn, T., Leitloff, J., Schiefer, F., & Hinz, S. (2021). Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 24-49. doi:<https://doi.org/10.1016/j.isprsjprs.2020.12.010>
- Kayalı, N. Z., & Omurca, S. İ. (2021). Konvolüsyonel Sinir Ağları (CNN) ile Çin Sayı Örütülerinin Sınıflandırılması. *Journal of Computer Science*, 184-191. doi:<https://doi.org/10.53070/bbd.989668>
- Keskin, M. V. (2022, Eylül 23). *Evrişimli Sinir Ağları*. Geleceği Yazanlar: <https://gelecegiyazanlar.turkcell.com.tr/konu/egitim/derin-ogrenme-cnn/evrisimli-sinir-aglari> adresinden alındı
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 84-90.
- Lecun, Y. (1989). Backpropagation applied to handwritten zip code. *Neural computation*, 541-551.
- Lee, D.-H., Kim, Y.-T., & Lee, S.-R. (2020). Shallow Landslide Susceptibility Models Based on Artificial Neural Networks Considering the Factor Selection Method and Various Non-Linear Activation Functions. *Remote Sensing*, 1-28.
- Marsalli, M. (tarih yok). *McCulloch-Pitts Neurons*. The Mindproject: https://mind.ilstu.edu/curriculum/mcp_neurons/index.html adresinden alındı
- Mostafa, S., & Wu, F.-X. (2021). Chapter 3 - Diagnosis of autism spectrum disorder with convolutional autoencoder and structural MRI images. *Neural Engineering Techniques for Autism Spectrum Disorder*, 23-38.
- Noriega, L. (2005). Multilayer Perceptron Tutorial. 1-12.
- Raitoharju, J. (2022). Chapter 3 - Convolutional neural networks. *Deep Learning for Robot Perception and Cognition*, 35-69.
- Rosebrock, A. (2021, Mayıs 14). *Convolution and cross-correlation in neural networks*. Pyimagesearch: <https://pyimagesearch.com/2021/05/14/convolution-and-cross-correlation-in-neural-networks/> adresinden alındı
- Rosenblatt, F. (2017). *The Perceptron A Perceiving and Recognizing Automation*. Buffalo: Cornell Aeronautical Laboratory, Inc.

- Ruder, S. (2017). An overview of gradient descent optimization algorithms. *Arxiv*, 1-14.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. 1-43. doi:<https://doi.org/10.48550/arXiv.1409.0575>
- Sharma, P. (2020, Temmuz 27). *7 Popular Image Classification Models in ImageNet Challenge (ILSVRC) Competition History*. Machine Learning Knowledge: <https://machinelearningknowledge.ai/popular-image-classification-models-in-imagenet-challenge-ilsvrc-competition-history/> adresinden alındı
- Singh, S. A., Meitei, T. G., & Majumder, S. (2020). Short PCG classification based on deep learning. *Deep Learning Techniques for Biomedical and Health Informatics*, 141-164.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. (Y. Bengio, Dü.) *Journal of Machine Learning Research*, 15, 1929-1958.
- Sun, S., Cao, Z., Zhu, H., & Zhao, J. (2019). A Survey of Optimization Methods from a Machine Learning Perspective. *Arxiv*, 1-30.
- Toda, Y., & Okura, F. (2019). How Convolutional Neural Networks Diagnose Plant Disease. *Plant Phenomics*, 1-14. doi:<https://doi.org/10.34133/2019/9237136>
- Wanga, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 61-67. doi:<https://doi.org/10.1016/j.patrec.2020.07.042>
- Taşhan, B. (2017) Road lane detection system with convolutional neural network (Tez No. 467573) [Yüksek lisans tezi, Bahçeşehir Üniversitesi]. YÖK Tez Merkezi. https://tez.yok.gov.tr/UlusalTezMerkezi/tezDetay.jsp?id=r_HEN_uwwz9qfeOkPFkiZw&no=1ECK3_7sRvgkMOExfHr7cw
- Akgül, T., Çalık, N., & Töreyin, B. U. (2020). Bulanık Sualtı Görüntülerinde Derin Öğrenme Tabanlı Balık Tespiti. IEEE, 1-4. doi:<https://spacing.itu.edu.tr/pdf/tansel.pdf>
- Ammour, N., Alhichri, H., Bazi, Y., Benjdira, B., Alajlan, N., & Zuair, M. (2017). Deep Learning Approach for Car Detection in UAV Imagery. *remote sensing*, 1-15. doi:<https://doi.org/10.3390/rs9040312>
- Aral, R. A., Keskin, Ş. R., Kaya, M., & Hacıomeroglu, M. (2018). Classification of TrashNet Dataset Based on Deep Learning Models. 2018 IEEE International Conference on Big Data (Big Data) (s. 2058-2062). Seattle, WA, USA: IEEE. doi:<https://www.doi.org/10.1109/BigData.2018.8622212>
- Aslan, T., & Yağımlı, M. (2023). İş Sağlığı ve Güvenliğinde Derin Öğrenme Tabanlı Risk Tespit ve Analizi. *The Journal of International Scientific Researches*, 224-236.
- Atık Yönetimi Vize Ders Notları. (Tarih yok). İstanbul Üniversitesi: [https://cdn.istanbul.edu.tr/FileHandler2.ashx?f=atik-yonetimi-ders-notlari-\(vize\).pdf](https://cdn.istanbul.edu.tr/FileHandler2.ashx?f=atik-yonetimi-ders-notlari-(vize).pdf) adresinden alındı
- Baheti, P. (2021, Eylül 13). Train Test Validation Split: How To & Best Practices. V7labs: <https://www.v7labs.com/blog/train-validation-test-set> adresinden alındı
- Barreto, S. (2023, Haziran 19). Differences Between Computer Vision and Image Processing. Baeldung: <https://www.baeldung.com/cs/computer-vision-image-processing-differences> adresinden alındı

- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *Computer Vision and Pattern Recognition*, 1-17.
doi:<https://doi.org/10.48550/arXiv.2004.10934>
- Carolis, B. D., Ladogana, F., & Macchiarulo, N. (2020). YOLO TrashNet: Garbage Detection in Video Streams. *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)* (s. 1-7). Bari, Italy: IEEE.
- Çakmak, D. (2020, Kasım 4). YOLOv4 ve YOLOv5. Medium:
<https://deryacakmak.medium.com/yolov4-ve-yolov5-9c4ad208579e> adresinden alındı
- Department of Electrical and Computer Engineering. (Tarih yok). Computer Vision and Image Processing. Northwestern University:
<https://www.mccormick.northwestern.edu/electrical-computer/academics/graduate/masters/specializations/computer-vision-and-image-processing.html#:~:text=Image%20processing%20algorithms%20are%20used,recognition%20and%20categorize%20image%20data> adresinden alındı
- Dive Into Deep Learning. (Tarih yok). Densely Connected Networks (DenseNet). Dive Into Deep Learning: https://tr.d2l.ai/chapter_convolutional-modern/densenet.html adresinden alındı
- Dive Into Deep Learning. (Tarih yok). Residual Networks (ResNet) and ResNeXt. Dive Into Deep Learning: https://tr.d2l.ai/chapter_convolutional-modern/resnet.html adresinden alındı
- Erin, K., Bingöl, B., & Boru, B. (2022). YOLO – Based Waste Detection. *Journal of Smart Systems Research (JOINSSR)*, 120-127. <https://dergipark.org.tr/en/download/article-file/2748494> adresinden alındı
- Hanbay, K., & Üzen, H. (2017). Nesne tespit ve takip metotları: Kapsamlı bir derleme. *Turkish Journal of Nature and Science*, 40-49.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Computer Vision and Pattern Recognition*, 1-12.
doi:<https://doi.org/10.48550/arXiv.1512.03385>
- Jocher, G. (2023, Mayıs 25). Are class and box losses calculated the same in YoloV8 and YoloV5? Github: <https://github.com/ultralytics/ultralytics/issues/2789> adresinden alındı
- Jocher, G. (2023, Nisan 24). Asking about YOLO head for detection Layers. Github: <https://github.com/ultralytics/yolov5/issues/11299> adresinden alındı
- Jocher, G. (2023, Aralık 15). Model Benchmarking with Ultralytics YOLO. Ultralytics: <https://docs.ultralytics.com/modes/benchmark/#why-is-benchmarking-crucial> adresinden alındı
- Jocher, G., Chaurasia, A., & Qiu, J. (2023, Ocak 10). Ultralytics YOLO. Github: <https://github.com/ultralytics/ultralytics> adresinden alındı
- King, R. (2023, Ocak 10). Brief summary of YOLOv8 model structure - Issue \#189. Github: <https://github.com/ultralytics/ultralytics/issues/189> adresinden alındı

- Kundu, R. (2023, Ocak 17). YOLO: Algorithm for Object Detection Explained. V7labs:
<https://www.v7labs.com/blog/yolo-object-detection> adresinden alındı
- Kurban, T. (Tarih yok). Dijital Görüntü İşlemede Temel Kavramlar. Erciyes Üniversitesi Akademik Veri Yönetim Sistemi:
<https://avesis.erciyes.edu.tr/resume/downloadfile/tubac?key=b2f6a348-b7a5-480e-9584-0387c7a56a0e> adresinden alındı
- Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. Computer Vision and Pattern Recognition, 1-11.
doi:<https://doi.org/10.48550/arXiv.1803.01534>
- Memiş, E. M. (2023, Nisan 9). Enkazda arama- kurtarma çalışmalarına yapay zeka desteği. (O. Sarı, & E. Yüzüğündü, Röportajı Yapanlar) <https://www.dha.com.tr/video/enkazda-arama-kurtarma-calismalarina-yapay-zeka-destegi-video-2233377> adresinden alındı
- Nepal, U., & Eslamiat, H. (2022). Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. Sensors, 1-15.
doi:<https://doi.org/10.3390/s22020464>
- Orak, İ. (Tarih yok). Çevre Eğitimi. Bingöl Üniversitesi:
https://www.bingol.edu.tr/documents/AA_Çevre%20eğitimi.pdf adresinden alındı
- Projectverba. (2022, Eylül 5). YOLO Waste Detection Image Dataset. Roboflow:
<https://universe.roboflow.com/projectverba/yolo-waste-detection/dataset/1> adresinden alındı
- Redmon, J. (2016). YOLO9000: Better, Faster, Stronger. Computer Vision and Pattern Recognition, 1-9. doi:<https://doi.org/10.48550/arXiv.1612.08242>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Computer Vision and Pattern Recognition, 1-10.
doi:<https://doi.org/10.48550/arXiv.1506.02640>
- Sharma, A. (2022, Nisan 11). Understanding a Real-Time Object Detection Network: You Only Look Once (YOLOv1). Pyimagesearch:
<https://pyimagesearch.com/2022/04/11/understanding-a-real-time-object-detection-network-you-only-look-once-yolov1/> adresinden alındı
- Singh, A. (2020, Ağustos 4). Selecting the Right Bounding Box Using Non-Max Suppression (with implementation). Analytics Vidhya:
<https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/> adresinden alındı
- Sonka, M., Hlavac, V., & Boyle, R. (1993). Image Processing, Analysis and Machine Vision. Springer Science Business Media. <https://link.springer.com/book/10.1007/978-1-4899-3216-7> adresinden alındı
- Şahin, H., & Küfür, M. S. (2016). GPGPU Yöntemi ile Görüntülerin Gerçek Zamanlı Ortorektifikasyonu. Harita Dergisi, 12-22.
<https://www.harita.gov.tr/uploads/files/articles/gpgpu-yontemi-ile-goruntulerin-gercek-zamanli-ortorektifikasyonu-1158.pdf> adresinden alındı

- Tsung-Yi Lin, P. D., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *Computer Vision and Pattern Recognition*, 1-10.
- Türk Bilim Araştırma Vakfı. (2020). Derin Öğrenme Yöntemleri Kullanarak Gerçek Zamanlı Araç Tespit. 1-14. <https://dergipark.org.tr/tr/download/article-file/1173793> adresinden alındı
- Türkiye İstatistik Kurumu. (2023). Adrese Dayalı Nüfus Kayıt Sistemi Sonuçları, 2022. <https://data.tuik.gov.tr/Bulton/Index?p=Adrese-Dayal%C4%B1-N%C3%BCfus-Kay%C4%B1t-Sistemi-Sonu%C3%A7lar%C4%B1-2022-49685> adresinden alındı
- Türkiye İstatistik Kurumu. (2023). Atık İstatistikleri, 2022. <https://data.tuik.gov.tr/Bulton/Index?p=Atik-Istatistikleri-2022-49570> adresinden alındı
- Wang, C.-Y., Liao, H.-Y. M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., & Hsieh, J.-W. (2019). CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *Computer Vision and Pattern Recognition*, 1-14. doi:<https://doi.org/10.48550/arXiv.1911.11929>
- Yanai, K., & Kawano, Y. (2015). Food image recognition using deep convolutional network with pre-training and fine-tuning. *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (s. 1-6). Turin, Italy: IEEE. doi:<https://doi.org/10.1109/ICMEW.2015.7169816>
- Yocher, G. (2023, Temmuz 30). Interpretation of the loss values. Github: <https://github.com/ultralytics/ultralytics/issues/4025> adresinden alındı