# Demiryollarında Otonom İHA Kullanan Görme Tabanlı Akıllı Hata Tespit Sisteminin Geliştirilmesi

**Program Kodu:1001**

**Proje No:120E097**


Proje Yürütücüsü:

**Prof. Dr. İLHAN AYDIN**

Araştırmacı(lar)

Prof. Dr. ERHAN AKIN

Prof. Dr. MEHMET KARAKÖSE

Bursiyer(ler)

SELÇUK SİNAN KIRAT

KADİR ŞAHBAZ

EMRE GÜÇLÜ

HSSEN ALDARWICH

MEHMET SEVİ

HASAN ÇELİK



EYLÜL 2022

ANKARA

# ÖNSÖZ

Bu projede demiryollarında oluşan kusurların tespiti için otonom IHA tabanlı bir sistem geliştirilmesi hedeflenmiştir. Bu amaçla otonom IHA ile alınan görüntülerden otonom uçuş için görüntü işleme ve derin öğrenme tabanlı yol takip algoritmaları geliştirilmiş ve elde edilen görüntülerden ray ile diğer bileşenlerinde oluşan kusurlar tespit edilmiştir. Ayrıca kusurlu veri sayısını arttırmak amacıyla derin öğrenme tabanlı yöntemlerden olan üretici çekişmeli ağlardan faydalanılmıştır. Projede özellikle otonom takip algoritmaları geliştirilirken öncelikle küçük boyutlu programlanabilir İnsansız Hava Araçları (IHA) kullanılmış daha sonra simülasyon ortamında ray ve çevresinin oluşturularak otonom uçuşların burada gerçekleştirildikten sonra gerçek sahada testleri yapılmıştır. Otonom uçuş için IHA'nın ön kamerasını kullanarak ufuk noktası tabanlı yöntemlerin yanı sıra tek kamera ile hem otonom uçuş hem de veri toplama işlemi yapılmıştır. Elde edilen veriler geliştirilen bir ara yüz ile değerlendirilerek ray ve bileşenleri ile ilgili kusurlar belirlenmektedir. Proje kapsamında ray yüzey kusurları ve bağlantı elemanı kusurlarının yanı sıra traverslerdeki kaymalar da tespit edilmiştir. Hem kusur tespitinde hem de otonom uçuş için geliştirilen algoritmalarda düşük ağırlıklı ve gerçek zamanlı çalışmaya uygun algoritmalar seçilmiştir. Bunun temel nedeni otonom uçuş esnasında verilerin hızlı bir şekilde işlenerek otonom uçuş için kontrolcünün gerekli aksiyonu verebilmesindendir. Algoritmaların tamamı Python ortamında OpenCV, Tkinter ve Tensorflow kütüphaneleri kullanılarak geliştirilmiştir. Otonom uçuş ve kusur tespiti için önerilen yöntemler ve otonom ray izleme için etkili ve verimli sonuçlar elde edilmesini sağlayarak projenin amacına ulaşmasını sağlamıştır.

Projenin bu aşamaya gelmesinde yayın, bursiyer ve malzeme desteği açısından TÜBİTAK'ın katkısı çok büyüktür. Ayrıca veri hazırlama ve etiketleme amacıyla gerek ARDEB'in sunduğu Lisans Bursiyeri gerekse BIDEB'in sunduğu star bursiyerleri desteği oldukça fayda sağlamıştır. Bu nedenle 120E097 numaralı Araştırma Projesini destekleyen TÜBİTAK'a sonsuz teşekkürlerimizi sunarız.

## İÇİNDEKİLER

## TABLOLAR LİSTESİ

## ŞEKİLLER LİSTESİ

## ÖZET

Demiryolu taşımacılığı güven, konfor ve hız açısından önemli bir ulaşım aracıdır. Hızlı trenlerin gelişmesi ile birlikte ülkemizde de son yıllarda demiryolu ulaşımında önemli yatırımlar yapılarak yeni hızlı tren hatları açılmıştır. Demiryolu hattının kalitesi trenin güvenli bir şekilde çalışması için önemlidir. Bu yüzden demiryolunda oluşan kusurların erken bir aşamada belirlenmesi gerekir. Demiryolu alt aksamı genellikle travers, ray, bağlantı elemanı ve balast olmak üzere dört kısımdan oluşur. Bu malzemelerde oluşan arızalar travers ve raylarda oluşan çatlaklar, bağlantı elemanlarda oluşan kırıklar, balast boşalması ve travers kaymaları olarak verilebilir. Demiryolu hatlarında oluşan problemleri tespit etmek amacıyla genellikle ölçüm trenleri veya ray hattı için özel tasarlanmış kara araçları kullanılmaktadır. Fakat bu araçlar hem ray hattını meşgul etmekte hem de özellikle ray üzerinde bulunan yabancı cisimler, rayın bir bütün olarak görülememesinden dolayı travers kaymaları ve ray altının boşalması gibi problemleri tespit edemezler. Ayrıca bu tür problemler ölçüm araçlarının yoldan çıkmasına neden olabileceğinden ciddi mali kayıplara neden olmaktadır.

Bu projede demiryolu hattı üzerinde otonom olarak uçuş yapabilecek bir IHA için algoritmaların tasarlanması, geliştirilen otonom sistem ile veri toplama ve elde edilen verilerden ray bileşenleri ile ilgili kusur tespit algoritmaları geliştirilmiştir. Otonom uçuş için kenar çıkarımı, Gabor filtresi, derin Hough dönüşümü ve MobilNetV2 tabanlı düşük ağırlıklı çizgi segment algoritması gibi yöntemler önerilmiştir. Kenar çıkarımı tabanlı teknikler ortam ışıklarından etkilenmesine rağmen Gabor filtresi kullanıldığında bulunan gereksiz çizgiler elenebilmektedir. Derin Hough dönüşümünde ray çizgileri kesintisiz olarak bulunduğu için ufuk noktası daha kolay belirlenebilmektedir. MobileNetV2 tabanlı düşük ağırlıklı çizgi segment algoritması gerçek zamanlı çalışma için oldukça uygun bir algoritma olup özellikte tek kamera ile takip ve veri toplama için uygun bir çözüm olmuştur. Ayrıca otonom uçuş algoritmaları benzetim ortamında geliştirildikten sonra gerçek ortamda test edilmiştir. Ray yüzey kusurlarının belirlenmesi için iki düşük ağırlıklı ağ olan MobileNetV2 ve SqueezeNet'in özellikleri birleştirilmiştir. Ray bağlantı elemanlarında oluşan kusurların tespiti için düşük ağırlıklı bir ağ modelinin yanı sıra ön işleme adımında bağlantı elemanının tespiti için çizgi yerel ikili örüntü yönteminden faydalanılmıştır. Ayrıca hatalı verileri çoğaltmak amacıyla çekişmeli üretici ağlar kullanılmış ve bölütleme amacıyla popüler derin öğrenme yöntemleri olan UNet ve Mask-RCNN ağları kullanılmıştır. Bağlantı elemanlarının tespiti ve ray yüzey kusurlarının tespit edilerek sınıflandırılması için YOLO tabanlı nesne tespit algoritmaları karşılaştırılmıştır. Geliştirilen algoritmalar ile demiryolu bakım ve kontrolü için aşağıdaki katkılar sağlanmıştır.

- Ray üzerinde IHA tabanlı yüksek hızlı otonom uçuş algoritmalarının geliştirilmesi,
- Düşük maliyet ile ray bakımı için alternatif bir yaklaşımın sunulması,
- Elde edilen görüntüler için kullanıcı dostu bir arayüz ile kusur tiplerinin belirlenmesi,
- Kusur tespiti için geliştirilen düşük ağırlıklı ağların gerçek zamanlı çalışabilme özelliği.



**Anahtar kelimeler:** Demiryolları alt yapı analizi, ray arızaları, otonom IHA, bilgisayarlı görme, derin öğrenme, GPU geliştirme kartı.

## ABSTRACT

Railway transportation is an important means of transportation in terms of reliability, comfort and speed. With the development of high-speed trains, new high-speed train lines have been opened in our country by making significant investments in railway transportation in recent years. The quality of the railway line is important for the safe operation of the train. Therefore, defects occurring in the railway must be identified at an early stage. Railway undercarriage generally consists of four parts: sleeper, rail, fastener and ballast. Faults in these materials can be given as cracks in sleepers and rails, fractures in fasteners, ballast discharge and sleeper slips. Measuring trains or specially designed land vehicles for the rail line are generally used to detect the problems that occur on the railway lines. However, these vehicles both occupy the rail line and cannot detect problems such as foreign objects on the rail, traverse slips and under-rail discharge due to the fact that the rail cannot be seen as a whole. In addition, such problems cause serious financial losses as they can cause measurement tools to go astray.

In this project, algorithms were designed for a UAV that can fly autonomously on the railway line, data collection with the developed autonomous system and defect detection algorithms related to rail components from the obtained data were developed. Edge extraction, Gabor filter, deep Hough transform and MobilNetV2 based low weight line segment algorithm are proposed for autonomous flight. Although edge subtraction-based techniques are affected by ambient lights, unnecessary lines can be eliminated when using a gabor filter. In the Deep Hough transform, the vanishing point can be determined more easily because the rail lines are found uninterrupted. The MobileNetV2-based low-weight line segment algorithm is a very suitable algorithm for real-time work and has been a suitable solution for tracking and data collection with a single camera. In addition, autonomous flight algorithms were developed in the simulation environment and then tested in the real environment. The features of two low-weight networks, MobileNetV2 and SqueezeNet, are combined to detect rail surface defects. In addition to a low-weight mesh model for the detection of defects in the rail fasteners, the line local binary pattern method was used for the detection of the fastener in the preprocessing step. In addition, contentious generating networks were used to reproduce faulty data, and popular deep learning methods UNet and Mask-RCNN networks were used for segmentation. Yolo-based object detection algorithms are compared to detect fasteners and detect and classify rail surface defects. The following contributions have been made to the developed algorithms for railway maintenance and control.

- Development of high-speed autonomous flight algorithms based on UAV on rail,
- Offering an alternative approach to rail maintenance with low cost,
- Determination of defect types with a user-friendly interface for the images obtained,
- Real-time capability of low-weight networks developed for flaw detection.

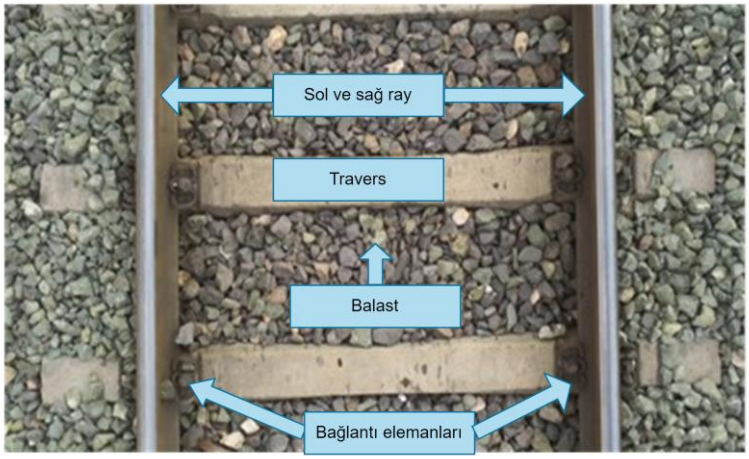


**Keywords:** Railways infrastructure analysis, rail component failures, autonomous UAV, computer vision, deep learning, GPU development board, embedded systems.

# 1. GİRİŞ

Raylı sistemler hem yük hem de yolcu taşımada sıklıkla kullanılan güvenli bir ulaşım aracıdır. Ülkemizde de son yıllarda raylı ulaşım sistemlerinde önemli ilerlemeler olmuştur. Ülkemizde yeni hızlı tren hatlarının kurulması ve var olan konvansiyonel hatların iyileştirilmesi amacıyla son yıllarda bir atılım olmuştur. Raylı sistemler çevre dostu, konforlu olmaları ve güvenirliklerinin yanı sıra özellikle hızlı trenler aracılığıyla hız ve zaman avantajı da sağlamaktadır. Ülkemizde raylı sistemlerin işletmesi Türkiye Cumhuriyeti Devlet Demir Yolları (TCDD) tarafından yapılmakta olup özellikle demiryolu hatlarının analizi ve bakımı için bir alt kurum olan Demiryolları Araştırma ve Teknoloji Merkezi (DATEM) sorumludur.

Demiryolu hatlarında ray ve bileşenlerinden oluşan alt aksamın belirli periyotlarda kontrolü gereklidir. Ülkemizde hızlı tren hatlarında bu kontrol genellikle bir ölçüm treni ile yapılmaktadır. Diğer konvansiyonel hatlarda ise kontrol işlemi demiryolu hat çalışanları tarafından gözle muayene ile yapılmaktadır. Özellikle hızlı tren hatlarında elektrikli trenler kullanıldığından bu hatlarda yüksek hızlarda güvenlik oldukça önemli bir kavramdır. Trenin hızı arttıkça hat güvenliği de önem kazanmaktadır. Dolayısıyla kestirimci bakım, güvenirlik ve kontrol demiryolu sektöründe çalışma maliyetlerini düşürmek, güvenlik ve güvenirlik açısından oldukça önemlidir. Demiryolu hattını oluşturan temel bileşenler Şekil 1.1'de verilmiştir.




Şekil 1. 1. Ray hattı temel bileşenleri

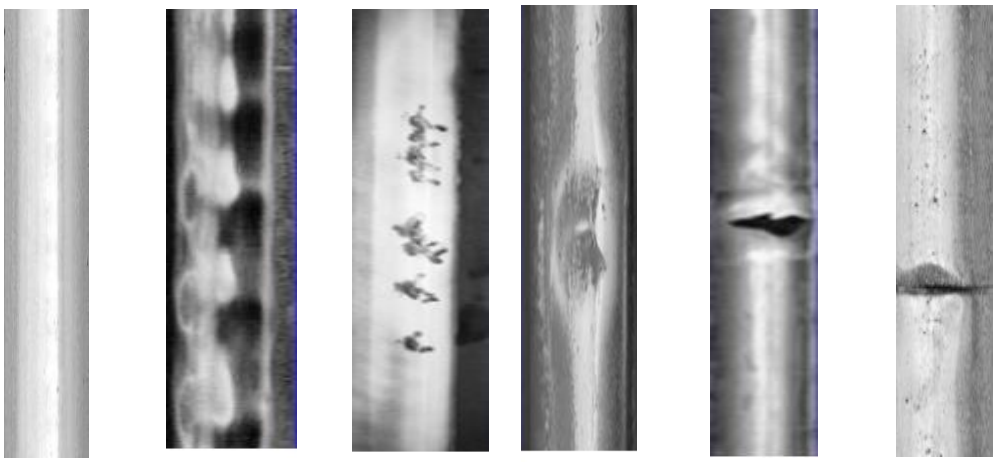
Yüksek hızlı demiryolu hatlarında hat durumunun sürekli iyi bir performansta tutulması periyodik bakımlar ile sağlanabilir. Güvenlik ve verimlilik diğer ulaşım ağlarında olduğu gibi demiryolu ağlarında da sıklıkla sorgulanabilmektedir. Demiryolu hatları için oluşturulan izleme sistemleri güvenli tren hatlarının sağlanması ve konforlu seyahat için kritik öneme sahiptir. Demiryollarında oluşan kazaların %60'ı raydan çıkma, %90'ı ise ray kırıklarından dolayı

olmaktadır [1]. Problemlerin çoğu ray alt aksamında meydana gelen kusurlardan dolayı oluşmaktadır. Bu yüzden özellikle hızlı tren hatlarında tren güvenliğini sağlamak için demiryolu bileşenlerinin izlenmesi esastır. Şekil 1.1'de gösterilen ray bileşenlerinden traversler daha önceleri ahşaptan yapılırken son yıllarda beton olarak imal edilmektedir. Bu traverslerin ömürleri genellikle 50 yıl olarak belirlenmesine rağmen alkali-silikon reaksiyonundan dolayı daha erken bozulabilmektedir. Ayrıca yağışlar sonrasında su ile nüfuz ve tren geçişleri esnasında oluşan basınçtan dolayı küçük çatlaklar oluşabilmektedir [2]. Bu çatlaklar zamanla daha ciddi hasarlara sebebiyet verip rayda da bozulmalara neden olabilir. Bağlantı elemanları her iki rayda ray ile traversi sabitleyen bileşenlerdir. Bağlantı elemanlarının çıkması veya kusurlu olması nedeniyle ray ve travers arasındaki sabitleme kaybolacağından tren raydan çıkabilir. Hat hızına bağlı olarak haftada bir veya iki kez yüksek hızlı tren raylarının kontrolü gerekir [3]. Bu manuel denetimler şu anda demiryolu personeli tarafından ya raylarda yürüyerek ya da çok düşük hızlarda yüksek raylı bir araçla gerçekleştirilmektedir. Fakat bu tür denetimler kişinin görmesi ve dikkatine bağlı olduğundan ve görsel bir kayıt oluşturmadığından yeni raylı sistemlerde tercih edilmezler. Ayrıca demiryolları genellikle 15 gün veya ayda bir özel hat geometrisi ölçüm araçlarıyla otomatik hata denetimleri gerçekleşir. Bu otomatik denetimler esnasında ray hatlarındaki hat geometrisinin genişlemesi ölçülür. Fakat bağlantı elemanı problemleri ancak erken bir aşamada tespit edilirse hat geometrisinde oluşabilecek problemlerin önüne geçilebilir.

Demiryolu alt aksamında oluşan kusurlar için modern demiryolu işletmeleri genellikle bir ölçüm aracı kullanmaktadır. Bu araçların altına yüksek çözünürlüklü ve saniyede yüksek görüntü çerçevesi yakalama özelliğine sahip kameralar yerleştirilmektedir. Ölçüm aracının altında her koşulda görüntülerin net olarak alınabilmesi için bir ışıklandırma sistemi de kurulmaktadır. Gelen görüntüleri analiz ederek kusurların tespit edildiği bir endüstriyel bilgisayar ise ölçüm aracı içerisine yerleştirilmektedir. Demiryolu hattında ölçümler belirli periyotlarda yapılmaktadır. Ölçüm aracı ile yapılan kontrollerde demiryolu hattı meşgul olmaktadır. Bu işlem genellikle gece ve tren geçişlerinin olmadığı zamanlarda yapılır. Fakat hem pahalı bir teknik olması hem de tren hattını kontrol esnasında meşgul ettiği için çok kullanışlı bir yöntem değildir. Ayrıca travers kaymaları, ray geometrisi ve ray üzerinde olabilecek engellerin tespiti mevcut ölçüm araçları ile gerçekleştirilemez. Demiryolu hattındaki ray ve bileşenleri hem konvansiyonel hem de hızlı tren hatları için önemli olup bu bileşenlerde oluşacak bir aksama tren hattının durması ve trenlerin yoldan çıkmasına neden olur. Bu aksamalar demiryolu işletmeleri için hem para hem de zaman kaybına neden olur. Bu yüzden bu tür bileşenlerin otomatik izlenmesi önemli bir gereksinim haline gelmiştir. Otonom bir IHA ile yapılan kontroller ile ölçüm aracı ile kontrol edilemeyen kusurlar belirlenebilir. Ayrıca böyle bir sistem tren hattını meşgul etmeyeceği için kontrollerin istenilen zamanda yapılmasına imkân verecektir.

## 1.1. Demiryollarında Alt Yapısında Oluşan Kusurlar

Demiryolu rayları hattın temel parçaları olup rayların sağlığı demiryolu işletmelerinin güvenliğini sağlamak için kritik öneme sahiptir. Ray ile ilgili kusurlar genellikle vagonlar, tren geçişlerinden dolayı aratan stresler ve zorlu çalışma ortamlarından dolayı oluşmaktadır. Ray yüzeylerinde farklı kusur tipleri görülmekte olup bu kusurların ray yüzeyi boyunca yayılması yüksek riskli arızalara ve ciddi kazalara neden olur. Ray yüzey kusurları ondülasyon, kabuklanma, bodruklar, taşlama izleri gibi farklı sınıflara ayrılabilir [4]. Klasik ray yüzeyi kusurları olarak ray ondülasyonları dalga şeklinde deformasyon ve düzensizlikleri ifade eder. Tren tekerleği ile ray temas gerilmesi ile oluşan kusur türü olarak kabuklanma kusurları oluştur. Bu kusurlar genellikle koyu lekeler olarak görülmektedir. Şekil 1.2'de oluşan ray yüzey kusurları verilmiştir.



(a) Sağlam (b) Ondülasyon (a) Ezilme (d) Çökme (e) Kabuklanma (f) Boşluk

Şekil 1.2. Ray yüzey kusur tipleri

Şekil 1.2'de gösterilen ray kusurları ray yüzeyinde oluşan kusurları göstermektedir. Bu kusurlar erken bir aşamada belirlenmezse daha ciddi kusurların oluşması kaçınılmazdır. Ray bileşenlerinde oluşan diğer bir kusur tipi ise bağlantı elemanı kusurlarıdır. Demiryolu bağlantı elemanı rayları traverslere sabitlemek için kullanılan önemli bir bileşendir. Bu bileşenlerde uzun süreli ray titreşimi ve sıcaklık değişimleri nedeniyle bağlantı elemanında hasarlar oluşabilir veya bağlantı elemanı tamamen çıkabilir [5]. Demiryolunun güvenliğini sağlamak için bağlantı elemanlarının periyodik olarak kontrolü gereklidir. Bağlantı elemanlarında oluşan kusurlar Şekil 1.3'te verilmiştir.

**(a)** Sağlam **(b)** Hasarlı **(c)** Eksik

Şekil 1.3. Bağlantı elemanı kusurları

Şekil 1.3'te gösterilen bağlantı elemanı problemlerini bilgisayarlı görme tabanlı tespit etmek amacıyla üç aşamalı bir yaklaşım kullanılmaktadır. İlk olarak bağlantı elemanını içeren görüntülerin elde edilmesi, elde edilen görüntülerden bağlantı elemanının çıkarılması ve daha sonra görüntü işleme alt sistemi ile kusur tespitinin yapılmasıdır. Ayrıca sadece rayı yukarıdan görüp tespit edilebilecek kusurlar da mevcuttur. Doğal afetler, toprak kayması ve sel gibi durumlarda ray altındaki balastın boşalması bu kusurlardan biridir. Şekil 1.4'te bu kusur tipi verilmiştir.

Şekil 1.4. Balast eksikliği

Şekil 1.4'te balast boşalması olan bir rayın yandan bir IHA ile çekilmiş görüntüsü görülmektedir. Bu tür kusurları ölçüm treni ile tespit etmek zordur. Çünkü ciddi hasar görmüş raylar ölçüm treninin raydan çıkmasına neden olur. Ray bileşenlerinde oluşan bir diğer kusur ise traversler

yüzeylerinde oluşan çatlaklar ve travers kaymalarıdır [6]. Travers yüzeylerinde bulunan çatlaklar erken bir aşamada belirlenmez ise traversin tamamen çıkması ve rayda ciddi kusurların oluşmasına neden olur. Şekil 1.5'te traverslerde oluşan bir çatlak durumu verilmiştir.

Şekil 1.5. Travers çatlakları

Traverslerde kontrol edilmesi gereken bir diğer kusur ise traverslerde kayma olup olmadığının belirlenmesidir. Travers kaymaları ray ölçüm araçları ile tespit edilemez. Bu kusurları tespit etmek için daha çok ray hattının belirli bir yükseklikten görüntülerinin alınması gerekir. Bu yüzden bir IHA ile alınan görüntülerden kaymalar tespit edilebilir. Şekil 1.6'da IHA'dan alınan bir görüntüde ray ve traverslerin tespiti gösterilmiştir.

**(a)** IHA ile alınan ray görüntüsü **(b)** Ray ve traverslerin tespiti

Şekil 1.6. IHA'dan alınan görüntülerden ray ve travers tespiti

Şekil 1.6'da ray ve travers bileşenleri alınan görüntü çerçevelerinde belirlendikten sonra her bir traversin orta noktası tespit edilir. Daha sonra ardışık traversler arasındaki mesafe elde edilir. Bu mesafeler bir zaman serisi olarak kaydedilip anomali durumlar travers kayması olarak belirlenebilir. Ayrıca diğer bir yaklaşım ise traverslerin bölütlenerek sağ ve sol ray dışında kalan bölümleri arasındaki periyodikliğin ölçülmesidir.

## 1.2. İnsansız Hava Araçları

İnsansız hava araçları uzaktan kumanda ile kontrol edilebilen veya otonom bir şekilde uçuş gerçekleştiren araçlardır. İnsansız hava araçları ilk olarak askeri amaçlar için tasarlanmış olmasına rağmen günümüzde hobi amaçlı, ticari ve askeri olmak üzere farklı sektörler için üretilmektedir. IHA'ları kategorilere ayırırken genellikle boyut, ağırlık, güç kaynağı ve otonomluk seviyesi gibi parametreler kullanılır [7]. IHA genellikle rotorlu, sabit kanatlı ve hibrit olmak üzere üç kategoriye ayrılabilir. Şekil 1.7'de IHA kategorileri için örnek IHA tasarımları verilmiştir.

**(a)** Rotorlu IHA **(b)** Sabit kantlı IHA **(c)** Hibrit IHA

Şekil 1.7. IHA kategorileri

Rotorlu IHA'lar genellikle bir veya daha fazla rotor ile dikey olarak kalkış yapabilme kabiliyetlerine göre gruplandırılmaktadır. Rotor sayıları genellikle bir, üç, dört, altı ve sekiz olarak seçilmektedir. IHA'ların manevra kabiliyeti, dayanıklılık ve taşıma kapasitesi rotor sayısı ve platform boyutundan etkilenmektedir. Çok rotorlu IHA'lar kullanım kolaylığı, kapalı ortamda çalışabilme ve erişilebilirlik özelliklerine sahip olmasına rağmen yük taşıma kapasiteleri düşüktür.

Sabit kanatlı IHA genel amaçlı uçaklara benzemekte olup Bernoulli prensibine göre üst ve alt taraf arasındaki hava basıncı farkından uçuş sağlanır. IHA'nın dikey kütlesinin dikey olarak kaldırılmasına gerek olmadığı için rotorlu IHA'lara göre enerji verimli uçuş sağlarlar. Bu IHA'lar uzun mesafeli uçuş, geniş alan kapsama ve yüksek hızda uçuş gibi avantajlara sahip olmasına rağmen pahalı olma, kullanım zorluğu ve uçuş ve iniş esnasında bir alan gerektirmesinden dolayı tercih edilmezler.

Hem sabit kanat hem de rotora sahip olan hibrit IHA'lar sabit kanadın doğrusal uçuş özelliklerini kullanır. Kanatlar uçuş esansında ana hava platformu olarak çalıştığından enerji verimliliği sağlar. Bazı modellerde aynı motorların hem yatay hem de dikey uçuş için kullanımını sağlayan 90 derece eğilebilen rotorlar bulunur.

### 1.2.1. IHA Uçuş Kontrolü

IHA'ları uzaktan uçurmak için farklı uçuş kontrol birimleri mevcuttur. Bu kontrol sistemlerinde sinyal erişimi, kolay kullanım ve taşınabilirlik gibi parametreler önemlidir. IHA'nın uzaktan kontörlü için kullanılabilecek bir yapı bir oyun koluna sahip olan uçuş kontrolcüsü kullanmaktır.

Bu kontrolcüler hem yazılım hem de donanım ara yüzleri ile bazı özellikler içermektedir. Bazı özellikler manuel olarak yönlendirilen bütün IHA'lar için özeldir. Uzaktan kontrolcü genellikle bir IHA'nın rotorlarının belirli hareketlerinin kontrolü için aynı özellikleri kullanır. Şekil 1.8'de iki tipik IHA için kontrolcü ara yüzleri verilmiştir.


**(a)** Parrot Anafi **(b)** DJI Phantom 4

Şekil 1.8. Farklı IHA'lar için uçuş kontrolcüsü

Şekil 1.8'de görüldüğü üzere kontrolcü ile bir cep telefonu bağlantısı veya ekran bağlantısı yapılarak IHA'nın aldığı görüntüler anlık olarak ekranda gösterilmektedir. Ayrıca cep telefonu veya tabletlere yüklenen uygulamalar ile kontrolün cep telefonu veya tablet ile yapılması da sağlanabilir. Fakat kontrolcü ile kullanılınca 4 km'ye kadar IHA'nın bağlantısı kopmadan kontrolü sağlanabilmektedir. Daha gelişmiş uçuşlar video girişi ve sinyal girişi gibi ek özellikler gerektiren yer kontrol istasyonları ile yapılabilir. Bu tür sistemlerin taşınabilirliği daha düşüktür. Ayrıca bazı yazılım destekleri ile IHA'lar için uçuş kontrol asistanı da sağlamaktadır. Ön programlanmış uçuş rotaları ve otonom nesne takibi gibi özellikler ara yüz programları ile sağlanabilmektedir.

### 1.2.2. Faydalı Yük

IHA'lar farklı türde faydalı yükleri taşıma yeteneğine sahiptir. Bunlar optik sensörler olabileceği gibi mekanik araçlar da olabilir. Özel IHA uygulamaları için faydalı yük kullanımı IHA platformlarının yük kapasitesinden dolayı sınırlıdır. IHA'lar genellikle görüntü ve video kaydı için yüksek çözünürlüklü bir kamera ile donatılır. Bu kameralara ile endüstriyel kontrol işlemleri yapılabileceği gibi ticari uygulamalar için de görüntüleme yapılabilir. Video ve görüntü kalitesi kamera çözünürlüğü, odak, diyafram açıklığı ve deklanşör hızından etkilenir [8]. Ayrıca bazı IHA'lar özel uygulamalar için termal kamera ile donatılabilir. LiDAR sensörleri, 3D olarak görselleştirilmek üzere ortamların ve nesnelerin doğru nokta bulutu verilerini yakalayabilir. Şekil 1.9'da bazı kamera modülleri verilmiştir.

**(a)** Parrot 4K kamera

**(b)** DJI 4K kamera


**(c)** Canon gimbal kemara


**(d)** Flir termal kamera

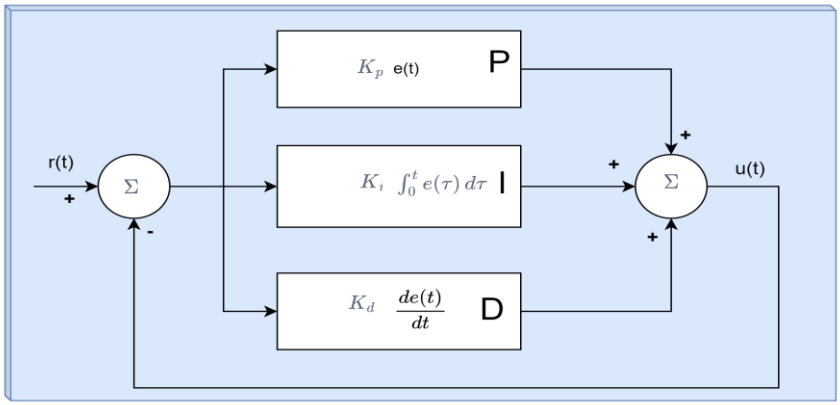
Şekil 1.9. IHA için Farklı kamera türleri

Şekil 1.9'da verilen kameralardan 4K olanlar daha yüksek çözünürlükte veri toplarken Canon kamera ise daha yüksek bir doğrulukta verileri elde eder. Flir'in termal kamerası ise nesne tespit, kurtarma operasyonları ve endüstriyel kontrol işlemlerinde hem termal hem de RGB görüntüleme sağlar.

### 1.2.3. IHA'ların Otonom Uçuşu

IHA'lar farklı seviyelerde otonom olarak çalıştırılabilirler. İnsan tarafından işletilme durumunda pilotluk ve görev ile ilgili kararlar operatör tarafından gerçekleştirilir. İnsan tarafından yetkilendirilmiş otonom IHA uygulamalarında IHA'nın hangi eylemleri yapması gerektiğini bir otomatik pilot aracılığıyla bildirir. İnsan denetimli otonom sistemde ise işletimsel kararlar vermek yerine işlevi güvence altına almak için sistem operatör tarafından denetlenir. Tam otonom IHA sistemlerinde ise IHA bütün kararları bir insan müdahalesi olmadan alır.

Tam otonom uçuş için IHA'nın harita üzerinden GPS bilgisi, görüntü bilgisi ve benzeri sensörlerden alınan bilgilere göre uçuş kontrolünü yapması gerekir. Uçuş kontrolü esnasında bir Oransal-Integral-Türevsel(Proportional Integral Derivative-PID) denetleyici kullanmak gerekir. Bir durumu seçilen bir duruma getirmek için bir kontrol geri besleme döngüsünde PID denetleyici kullanılabilir. PID bu işlemi gerçekleştirirken oransal, integral ve türevi kullanır. Her bir işlemin ayrı bir görevi vardır. Bazı sistemler sadece oransal kontrole veya oransal ile diğer bileşenlerden biri ile yani PI veya PD şeklinde tasarlanabilir [9]. Sadece oransal kontrol kullanıldığında salınımlara neden olur ve sistem hiçbir zaman yakınsamayabilir. Bu durumu çözmek için genellikle türev ile birlikte oransal birleştirilir. Eğer hızlı yakınsama gerekiyorsa integral denetleyici de kullanılabilir. Şekil 1.10'da basit bir PID denetleyicinin yapısı verilmiştir.




Şekil 1.10. PID denetleyici

PID denetleyici ile IHA kontrolü yapılırken x, y ve z ekseni için kontrol yapılarının tasarlanması gerekir. Dolayısıyla her bir eksende IHA'nın hareketi kontrol edilmiş olunur. PID ile denetlenen sistem referans değere göre hatanın sıfıra yakınsamasını sağlamaya çalışır. Projede kullanılan sistemde IHA hat üzerinde hareket ederken iki rayın orta noktası veya kesişim noktaları bulunur. IHA belirli bir yükseklikte uçtuğu için z eksenini kontrol etmeye gerek yoktur. Ayrıca y ekseni boyunca sabit bir hızda ilerleme sağlayacaktır. X eksenine göre IHA'nın konumu ile rayın orta noktası arasındaki mesafe sıfırlanırsa IHA ray ortasında hareket eder.

### 1.2.4. IHA'ların Programlanması

Hobi amaçlı kullanılan IHA'ların birçoğu programlanabilme özelliklerine sahip değildir. Bununla birlikte geliştirme amaçlı kullanılabilecek IHA'ların programlanabilme kabiliyetleri vardır. Proje kapsamında oluşturulan prototip IHA dışında algoritmaların test edilmesi ve veri toplama amacıyla iki farklı IHA kullanıldığından bu IHA'ların programlanma özelliklerinden verilmiştir.
DJI ailesinden Tello IHA'lar programlanabilme özelliğine sahiptir. Bu IHA 720p HD kameraya sahip olup havada kalma süresi 13-15 dakika arasında değişmektedir. IHA üzerinde kaza koruma özelliği bulunmakta olup elden kalkış özelliğine sahiptir. Bu IHA 87 gram ağılığında olup üzerinde Led ışık, telemetre, baometre gibi sensörler bulunmaktadır. Maksimum hızı saniyede 30 metre olup kamera ile 30 fps hızında görüntü alınabilmektedir. Şekil 1.11'de IHA'nın bir görünümü verilmiştir.

Şekil 1.11. DJI Tello IHA

Şekil 1.11'de gösterilen IHA Wİ-Fi ile uzaktan kontrol edilebilmektedir. IHA'nın programlanabilmesi için Pyhton SDK kurulması gerekmektedir. Şekil 1.12'de basit şekilde bir DJI Tello'yu programlama örneği verilmiştir.

```
import cv2
from djitellopy import Tello
tello = Tello()
tello.connect()
tello.streamon()
frame_read = tello.get_frame_read()

tello.takeoff()
cv2.imwrite("picture.png", frame_read.frame)

tello.land()
```

Şekil 1.12. DJI Tello'dan görüntü kaydetme

Şekil 1.12'de ilk olarak djitellopy kütüphanesinden Tello sınıfı eklenmektedir. Daha sonra bu sınıftan bir nesne oluşturulmakta ve IHA'ya bağlantı kurulmaktadır. IHA uçuş yaptıktan sonra bir frame alınarak kaydedilmekte ve tekrar iniş yapması sağlanmaktadır. Bu IHA ile çoklu sürü IHA'ların uçuşları yapılarak kontrolleri sağlanabilmektedir.

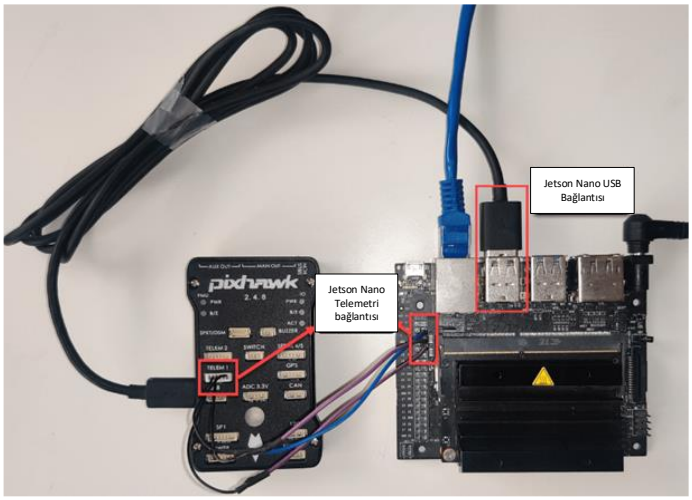
Programlanabilen bir diğer IHA ise Parrot Anafi modelidir. Parrot Anafi model IHA'ları programlamak için Olympe bir Python programlama arayüzü sunar. Bilgisayardan yürütülen uzak bir Python betiğinden IHA'yı bağlamak ve kontrol etmek için Olympe kullanılabilir. Olympe aynı zamanda simule edilmiş bir IHA'yı kontrol etmek için de kullanılır. Fakat fiziksel IHA'ya da bağlanarak kontrol sağlanabilir. Olympe Linux ortamında kurularak IHA'nın programlanması sağlanabilir. Şekil 1.13'te Parrot Anafi IHA için Olympe ile basit bir programlama kodu verilmiştir.

```
import olympe
import os
import time
from olympe.messages.ardrone3.Piloting import TakeOff, Landing

DRONE_IP = os.environ.get("DRONE_IP", "10.202.0.1")

def test_takeoff():
    drone = olympe.Drone(DRONE_IP)
    drone.connect()
    assert drone(TakeOff()).wait().success()
    time.sleep(10)
    assert drone(Landing()).wait().success()
    drone.disconnect()

if __name__ == "__main__":
    test_takeoff()
```

Şekil 1.13. Olympe ile basit uçuş kodu

Şekil 1.13'te basit şekilde bir Parrot Anafi IHA'nın kalkışı için bir Python metodu yazılmıştır. Burada DRONE_IP ile hangi IHA'ya bağlanacağı belirlenmektedir. Burada sanal veya gerçek IHA seçilebilmektedir. Gerçek IHA'nın fiziksel bir IP adresi olup eğer bu IP adresi girilirse kodlar gerçek IHA'ya gönderilmektedir. Bu kodlar ile IHA'ya bağlanılıp kalkışı yapılmakta 10 milisaniye sonra tekrar iniş yaptırılarak IHA bağlantısı kapatılmaktadır.

Proje kapsamında geliştirilen prototip IHA için Jetson Nano bilgisayarı kullanılmıştır. Bu bilgisayar üzerinde 4GB RAM ve ARM işlemci bulunmaktadır. Ayrıca derin öğrenme uygulamaları için 128-Core Maxwell GPU bulunmaktadır. Uçuş kontrol kartı olarak Pixhawk ile kullanılabilmektedir. Şekil 1.14'te Pixhawk ile Jetson Nano bağlantısı verilmiştir.

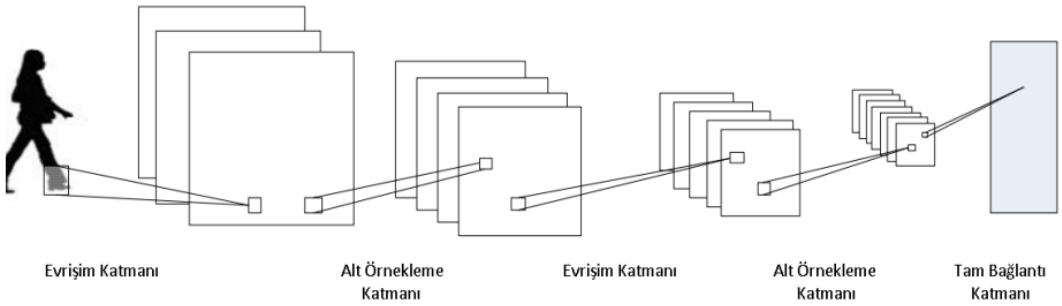



Şekil 1.14. Jetson Nano ve Pixhawk bağlantısı

Şekil 1.14'te Jetson Nano'nun J41 pini ile pixhawk'ın telemetri pini arasında bağlantı kurulmaktadır. Burada TX ve RX pinleri dışında topraklama ve 5V bağlantıları da yapılmaktadır. Jetson Nano üzerinde Ubuntu işletim sistemi kurulmaktadır. Ayrıca bütün görüntü işleme ve derin öğrenme kütüphaneleri kurularak uygulamalar gerçekleştirilmektedir. Fakat derin öğrenme algoritmaları normalde yavaş çalışmasına rağmen modeller optimize edilerek daha hızlı çalışmaları sağlanabilir. Bu amaçla TensorRT SDK'sı kullanılmaktadır. NVIDIA TensorRT, derin öğrenme uygulamaları için düşük gecikme süresi ve yüksek aktarım hızı sunan yüksek performanslı bir çıkarım iyileştiricisi ve çalışma zamanıdır [10]. TensorRT CPU platformlarından 36 kat daha hızlı performans göstermekte olup eğitilmiş sinir ağı modellerinin optimize edilmesi ve yüksek doğrulukla daha düşük hassasiyet için kalibre edilmesini sağlamaktadır. Yolo tabanlı bir nesne tespit algoritması Jetson Nano üzerinde saniyede 2 FPS ile çalışırken TensorRT ile optimize edilen bir nesne tespit algoritması Jetson Nano üzerinde 20 FPS'e kadar hızlanma sağlamaktadır.

## 1.3. Derin Öğrenme

Derin öğrenme, makine öğrenmesi yöntemlerinden Yapay Sinir Ağlarına (YSA) dayanır. YSA'daki gizli katman (ara katman) sayısı arttırıldığında ağ derinleşir. Derin öğrenme kavramındaki "derin" kelimesinin çıkış noktası YSA'nın gizli katman sayısının artırılması esasına dayanmaktadır. Derin öğrenme verinin öz niteliklerini çıkarma işini otomatik olarak yapar. Araştırmacının bu süreçten kurtulmuş olması derin öğrenmeyi çekici hale getiren önemli nedenlerden birisidir. Derin öğrenmede görüntü işleme için Evrişimsel Sinir Ağları (ESA) kullanılmaktadır. İlk ESA çalışması 90'lı yıllarda ABD'de posta kodlarının tanınabilmesi için geliştirilmiş LeNet-5 isimli ağdır. 2012 yılında ImageNet gibi çok fazla görüntü sınıfının ayırt edilebilmesi için yapılan yarışmada AlexNet isimli ağ büyük başarı gösterince ESA'lar ilgiyi üzerlerine çekmiştir [11]. ESA'lar derin öğrenme yöntemleri içerisinde en çok bilinen yapılardır. Örüntü tanıma çalışmalarında kullanılırken parametre sayısını azaltması ESA'ları önemli kılan özelliklerinden birisidir. ESA'lar özellikle görüntü sınıflandırma görevlerinde kullanılmaktadır. Şekil 1.15'te klasik bir ESA model yapısı görülmektedir.

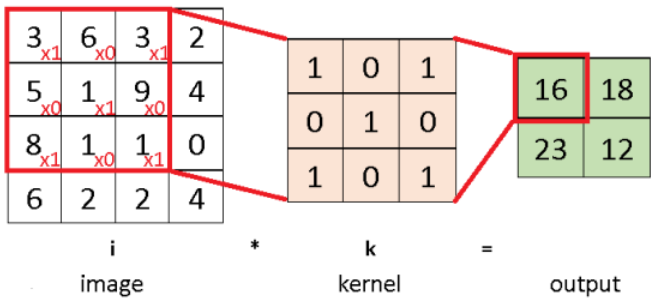



Şekil 1.15. Klasik Evrişimsel Sinir Ağı Modeli

Aşağıda temel bir sinir ağında olması gereken katmanlardan bahsedilmiştir. Bu katmanlar Sinir ağlarının temelini oluşturmaktadır.

**Giriş Katmanı:** Bu katman ESA mimarisinin ilk katmanını oluşturmaktadır. Giriş verisi modelin başarımı için önemlidir. Görüntü boyutu eğer yüksek seçilir ise test ve eğitim süresini uzatabilir, bu büyük veri bellek açlığı da oluşturabilir. Giriş verisinin boyutunu düşürmek ise süreyi azaltabileceği gibi modelin başarısını da olumsuz etkileyebilir. İdeal giriş verisinin boyutunun belirlenmesi model için faydalı olacaktır.

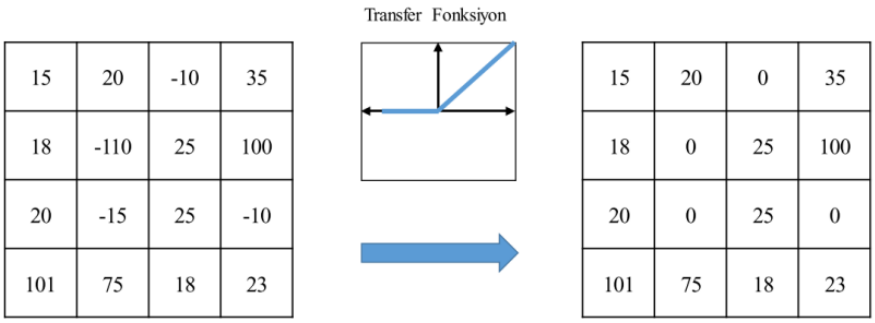
**Konvolüsyon Katmanı:** Evrişim katmanı evrişim işleminin yapıldığı yerdir. Bu işlem ile görüntünün öz nitelikleri çıkartılmaktadır. Temelde görüntüye bir filtre uygulanır, filtre istenmeyen bilgileri zayıflatır. İstenen bilgileri (öz nitelik) ise belirginleştirir. Bu sayede her evrişim katmanında görüntünün farklı öz nitelikleri çıkarılmış olur. Ağdaki ilk evrişim katmanında genel olarak görüntüdeki kenar bilgileri belirginleşir. Ağdaki katman sayısı artırıldıkça görüntüler üzerinde daha fazla ayrıntı çıkarılır. İleriki evrişim katmanlarında görüntü içindeki nesneler ayırt edilebilir hale gelir. Evrişim işlemi özel bir matris işlemidir. Kernel ya da filtre denilen ve genelde 3×3, 5×5 ya da 7×7 boyutlarındaki matris, görüntü matrisi üzerinde ilk pikselden başlanarak sırayla kaydırılır. Kaydırılma esnasında filtre içindeki sayısal değerler ile görüntü matrisi üzerinde denk geldiği piksel değerleri çarpılarak, çarpım sonuçları toplanır. $i$ görüntü matrisi ve $k$ kernel (filtre) olmak üzere, evrişim işlemi $(i * k)$ Şekil 1.16'daki gibidir.




Şekil 1.16. Evrişim İşlemi

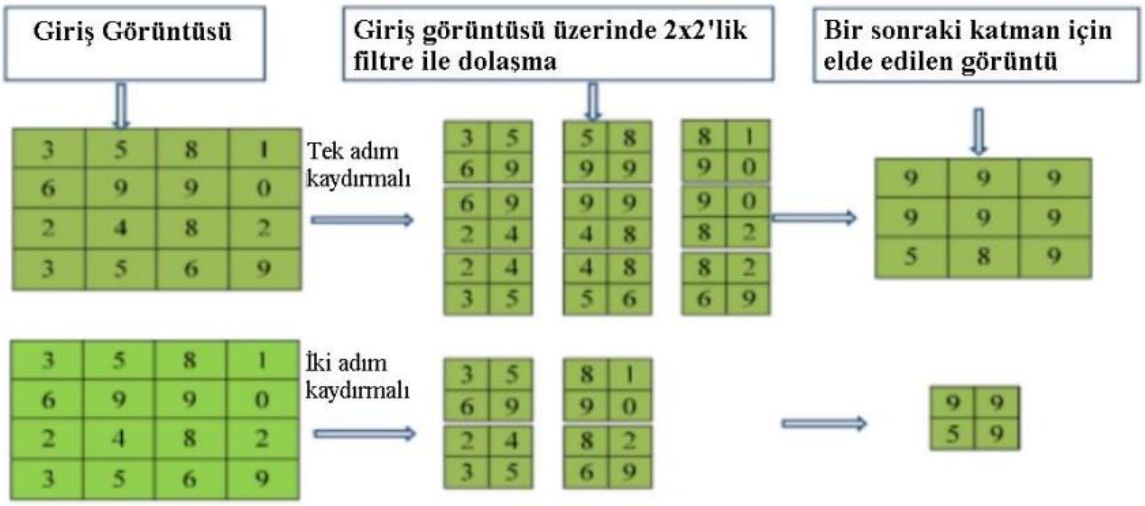
**ReLU Aktivasyon Fonksiyonu:** Evrişim işlemi neticesinde kerneldeki değerlere bağlı olarak çıktı matrisinde bazen negatif değerler bulunabilir. Bu değerleri elemine etmek için bir aktivasyon fonksiyonu (transfer fonksiyon) kullanılır. Evrişim katmanında aktivasyon

fonksiyonu olarak ReLU (Rectified Linear Unit) tercih edilir. ReLU, negatif değerleri 0 olarak, pozitif değerleri ise olduğu gibi bir sonraki katmana aktarır. Şekil 1.17'de evrişim işlemi neticesinde elde edilen örnek bir matrisin ReLU fonksiyonundan geçirildikten sonraki hali görülmektedir.




Şekil 1.17. ReLU Aktivasyon Fonksiyonu

**Havuzlama Katmanı:** Katmanın amacı evrişim katmanından gelen görüntüyü daha az parametre ile daha basit temsil edecek bir görüntüye düşürmektir. Evrişim katmanından gelen görüntünün boyutu küçülür. Kanal sayısında değişme olmaz. Şekil 1.18'de havuzlama işlemi görülmektedir.




Şekil 1.18. Havuzlama Katmanındaki İşlem

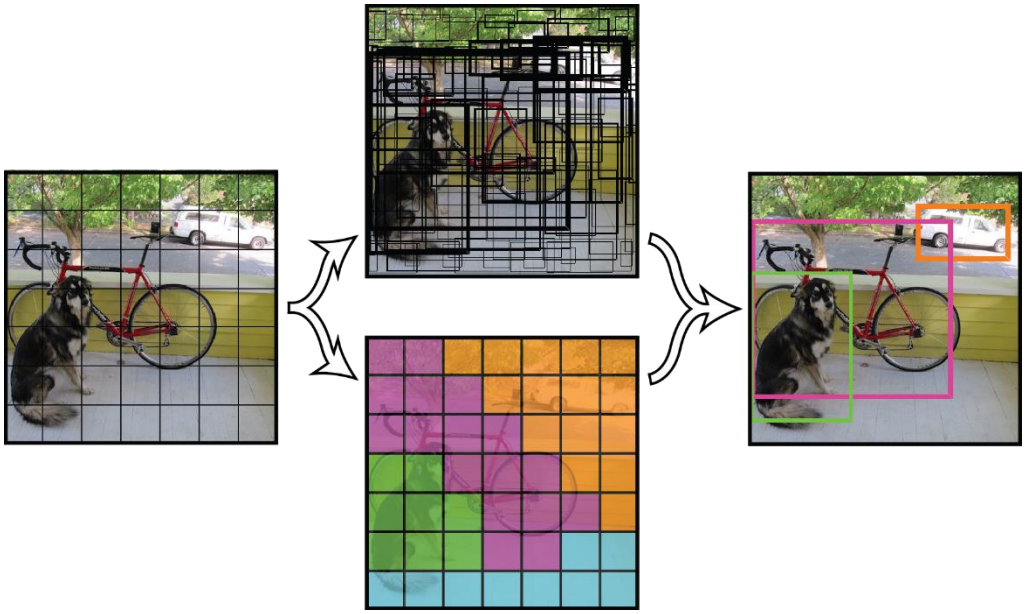
Şekil 1.18'de ESA'larda en sık kullanılan maksimum havuzlama işlemi görülmektedir. Örnekte görüntü matrisine 2×2'lik havuzlama filtresi uygulanmıştır. Maksimum havuzlama, her bölgedeki en büyük değeri seçecektir. Bu sayede görüntü matrisi üzerindeki her 2×2'lik bölge, en yüksek değerli piksel ile temsil edilecektir. Filtrenin adım parametresi de çıktıyı etkilemektedir. İki adım kaydırma yapılan örnekte görüntü matrisinin eni ve boyu yarıya düşmüştür. Bu sayede bir sonraki katmana çok daha az parametre gönderilmiş olacaktır.

Havuzlama çerçevesi 3×3 ya da daha büyük boyutta seçilirse, havuzlama katmanındaki çıktı daha çok küçülecektir.

**Tam Bağlantılı Katman:** Evrişim ve havuzlama katmanlarının art arda dizilmesiyle oluşan ağın sonunda bulunur. Tam bağlantı katmanına gelen görüntü matrisi iki boyutlu olduğu için öncelikle vektör haline getirilir. Elde edilen görüntü vektöründeki her bir değer YSA'lardaki nöronlarda olduğu gibi ağırlık değerleri ile çarpılarak çıkış hesaplanır. Adından da anlaşılacağı gibi bu katmanda giriş vektöründeki tüm değerler ağırlıklar ile çarpılır.

### 1.3.1. Nesne Algılama Modelleri

Nesne algılama sınıflandırma gibi bilgisayarlı görme konusundaki klasik problemlerden biridir. Ancak algılama; nesneleri tanımlayabilen, ancak nesnenin resimde nerede olduğunu tam olarak söylemeyen, birden fazla nesne içeren görüntüler için çalışmayan sınıflandırmadan daha karmaşık bir sorundur. Geleneksel nesne tespit algoritmalarında tespit edilen nesneyi sınıflandırmak için ekstra bir sınıflandırma işlemi yapılmaktadır. Algoritmalar öncelikle bir görüntüde nesnelerin bulunması muhtemel olan alanları belirlemektedir. Belirlenen alanlarda sınıflandırıcı olarak tasarlanan evrişimsel sinir ağları her bir bölge için ayrı ayrı çalıştırılarak nesne tespit edilmektedir. Yapılan sistemler iyi sonuçlar vermesine karşın görüntü iki ayrı işlemden geçirildiği için ortaya çıkan parametre sayıları ve ihtiyaç duyulan işlem gücü artmaktadır. Bu nedenle yapılan sistemleri gerçek zamanlı çalışan sistemlerde kullanmak mümkün değildir. Bu şekilde tasarlanan en yeni yaklaşımlardan biri olan R-CNN algoritmasında önce bir görüntüde bulunabilecek potansiyel sınırlayıcı kutular oluşturmakta ve ardından bu önerilen kutular üzerinde bir sınıflandırıcı çalıştırmak için bölge bazlı çalışan metotlar kullanılmaktadır [12]. Sınıflandırmadan sonra, üretilen sınırlayıcı kutuların güven değerlerini iyileştirmek, aynı nesnenin birden fazla kez algılanmasını ortadan kaldırmak ve görüntüdeki diğer kutulara göre kutuları yeniden sıralama gibi işlemler yapılmaktadır. Bu karmaşık işlemlerin her biri ayrı ayrı eğitildiğinden sistem oldukça yavaş ve optimize edilmesi zordur. YOLO, geleneksel nesne tespit yöntemlerinin aksine sınırlayıcı kutuların bulunması, sınıf olasılıklarının hesaplanması ve diğer tüm işlemleri tek bir regresyon problemi olarak ele almış ve nesne tespit alanına yeni bir görüş getirmiştir [13]. YOLO ile görüntü üzerinde hangi nesnelerin nerede olduklarını tespit etmek için görüntüye yalnızca bir kez bakılması yeterlidir. Tek bir evrişimsel ağ ile aynı anda birden fazla sınırlayıcı kutu tahmin edilmekte ve her bir sınıf için sınıf olasılıkları tahmin edilmektedir. Bu birleşik model, geleneksel nesne tespit etme yöntemlerine göre çeşitli faydalara sahiptir. Şekil 1.19'da YOLO tespit şeması görülmektedir.

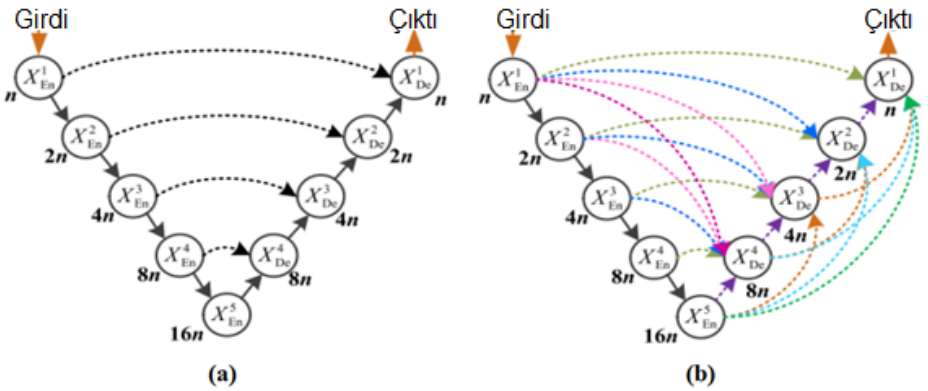


**Şekil 1.19.** YOLO Tespit Şeması

YOLO tespit modeli, eğitim setinde bulunan her bir görüntüyü Şekil 1.19'da gösterildiği gibi SxS kare hücrelere(grid) bölmektedir. Tespit edilmesi istenen nesnelerden herhangi birinin merkez konumu bölünen hücrelerden herhangi birinin içerisinde ise, merkezin bulunduğu hücre o nesneyi tespit etmekten sorumludur. Her bir hücre, B sınırlayıcı kutularını üretmekte ve bu sınırlayıcı kutular için güven skorunu tahmin etmektedir. Güven skoru, modelin ürettiği sınırlayıcı kutunun bir nesne içerdiğinden ne kadar emin olduğunu ve üretilen sınırlayıcı kutunun üretilme olasılığının ne kadar doğru olduğunu yansıtmaktadır.

### 1.3.2. Görüntü Bölütleme

Derin öğrenme temeline dayanan ESA'lar ile görüntüler sınıflandırılabilmektedir. Fakat bazı görevlerde her bir görüntünün sınıflandırılması yerine görüntü içerisindeki çeşitli bölgelerin işaretlenmesi gerekir. Bu durum görüntüyü oluşturan piksellerin sınıflandırılabilmesini gerektirmektedir. U-Net bu amaçla tasarlanmış bir ağdır ve ilk olarak medikal görüntülerde kullanılmıştır [14]. U-Net, kodlayıcı ve kod çözücü mimarisine sahip ağlardandır. Görüntüdeki her bir pikselin sınıf etiketini öğrenen U-Net, görüntüyü segmente edebilir. U-Net zamanla geliştirilerek farklı versiyonları çıkmıştır. U-Net ailesinin en son versiyonu U-Net 3+'dır [15]. Şekil 1.20'de U-Net ile U-Net3+ görülmektedir.



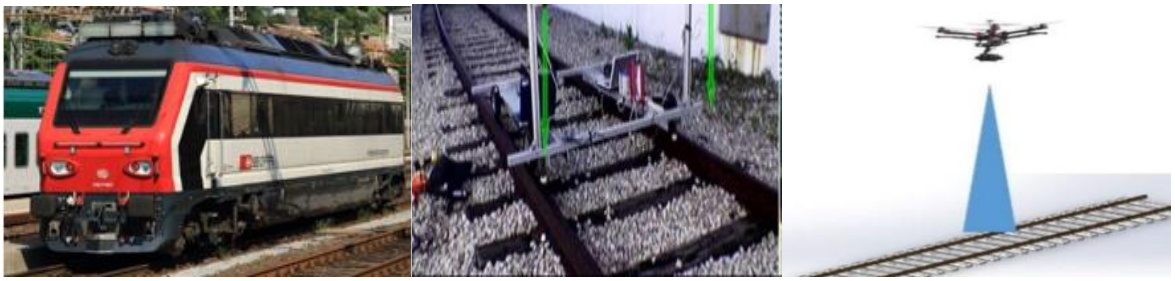

Şekil 1.20. U-Net Modelleri

Şekil 1.20'de görüldüğü gibi input tarafında X1-X5 arasındaki katmanlar kodlayıcı, output tarafında X4-X1 arasındaki katmanlar ise kod çözücü olarak görev yapmaktadırlar. Giriş tarafında X5 katmanına kadarki kısım daralma yolu, X5'ten çıkışa kadarki kısım genişleme yolu olarak adlandırılmaktadır. U-Net 3+'nın U-Net'ten temel farkı atlama bağlantılarından kaynaklanmaktadır. Görüntü daralma yolunda ilerlerken boyutu küçülmekte, temsil sayısı artmaktadır. Genişleme yolunda ilerlerken boyutu büyümekte, temsil sayısı azalmaktadır. Genişleme yolunda yukarı çıkarken boyutu büyüyen görüntü piksel değerleri açısından daralma yolunda aynı seviyedeki haline göre veri kaybına uğramıştır. Bu kaybı azaltmak için atlama bağlantıları ile imgenin daha yüksek çözünürlüklü temsilleri, genişleme yolundaki katmanlarda birleştirilir. Bu sayede ağ görüntü özniteliklerini öğrenirken ayrıntıları kaçırmamış olur. Özellikle U-Net 3+'da atlama bağlantılarının daha fazla olduğu görülmektedir.

# 2. LİTERATÜR ÖZETİ

Demiryolu ulaşımı gibi sürdürülebilir ulaşım yöntemlerine yönelik artan talebi karşılamak için demiryolu altyapısının güvenli, güvenilir ve verimli olması gerekmektedir. Demiryolu ulaşımının günümüzde karşılaştığı temel sorunlardan birisi yüksek trafik yoğunluğunun bakım ihtiyacını artırması ve aynı zamanda bakım faaliyetlerini gerçekleştirmek için zaman çizelgelerindeki boşlukların kullanılabilirliğini azaltmasıdır. Literatürde demiryollarının otonom izlenmesi ile ilgili çeşitli çalışmalar yapılmıştır. Bütün demiryolu işletmeleri uygun bir bakım planı uygulamak ister. Genel olarak bakım işlemleri geleneksel yöntemlerle yapılmaktadır. Demiryolu sistemlerinde çalışma esnasında oluşabilecek birçok kusur türü vardır. Güvenli yolculuğun sağlanması için bu bilinen kusur durumlarının izlenmesi ve erken teşhisin sağlanması gerekir [16]. Oluşacak arızaların birçoğu başka arızaları da tetiklemektedir. Örneğin, raylarda oluşacak ciddi bir arıza sadece raylara zarar vermez aynı zamanda demiryolu hattının demiryolu ağı üzerinde geniş hasara yol açabilir [17].

Demiryolu işletmelerinin bakım için en çok kullandığı yöntem geleneksel bakım yöntemleridir [18]. Bu yöntemler hem pahalı hem de inceleme esnasında sistemin servis dışı olmasına neden olmaktadır. Geleneksel yöntemlerde insan muayenesi ve temaslı ölçüm teknikleri, raylardaki arızaları tespit etmek için halen kullanılmaktadır. Bu da kusur tespitinde düşük hassasiyet ve düşük doğruluk oranına neden olmaktadır. Demiryollarının güvenliği için belli aralıklarla bu ölçümlerin yapılması gerekmektedir. Özel ray ölçüm cihazlarıyla yapılan bu muayene zamandan ve işten kayba neden olmaktadır. Şekil 2.1'de bakım yapmak için kullanılan farklı ray ölçüm araçları görülmektedir.



Şekil 2.1. Ultrasonik Ray Muayene Cihazı

Şekil 2.1'de verilen ölçüm treni ve ölçüm aracı kullanıldığında tren hattı meşgul olur. Bu yüzden bu cihazları kullanmak için demiryolunun boş olduğu bir zaman seçilmelidir. Günümüzde demiryolu incelemesi için ultrasonik dalgaların yanı sıra görsel kameralar, akustik emisyon ve termografi girdap akımı gibi yöntemler kullanılmaktadır. Bütün bu yöntemler zamandan ve iş gücünden kayba neden olmaktadır. Dezavantajlar düşünüldüğünde yeni teolojilere yelken açmak zorunlu hale gelmiştir. Geleneksel yöntemlerle yapılan demiryolu hattı bakımı maliyetli

ve ciddi zaman gerektiren bir işlemdir. Bakım işlemi otonom İHA'larla yapıldığında zaman ve maliyet düşürülürken aynı zamanda İHA'larla elde edilen görüntülere hibrit algoritmalar uygulanarak kusurlu demiryolu bileşenleri yüksek doğruluk oranıyla tespit edilebilmektedir. Bu analizlerden sonra demiryolu alanındaki diğer kuruluşlar gerekli geliştirme ve iyileştirmeleri yaparak kaliteli hizmet sunabileceklerdir.

Resendiz ve ark. [16] bir ölçüm aracı ile alınan ray görüntülerinden farklı kusur tiplerini tespit etmek amacıyla MUSIC algoritmasını önermişlerdir. Farklı ölçüm araçları kullanılarak demiryollarında farklı demiryolu alt yapısında oluşan kusurlar belirlenmiştir. Zhang ve ark.[18] ray yüzey kusurlarını belirlemek için koşullu rastgele alanlar ile evrişimli sinir ağ modelini birleştirmiştir. Önerilen yöntem ile tek bir ray yüzey kusuru belirlenmiştir. Ray yüzey kusurlarının belirlenmesi için eğrilik filtresi ve Gaussian karışım modeli tabanlı bir yaklaşım önerilmiştir [19]. Önerilen modelde ray çıkarımı yapıldıktan sonra eğrilik filtresi ile görüntü filtrelenmekte ve daha sonra Gauss karışım modeli ile modellenerek kusurlar belirlenmektedir. Kusurlar bölütleme ile elde edilmektedir. Ray yüzey kusurlarının tespiti için YoloV2 ve grabcut yöntemi tabanlı bir yaklaşım önerilmiştir [20]. İlk olarak kenar piksellerinin standart sapmasına göre ray çıkarımı yapılmıştır. Daha sonra diferansiyel kutu sayımı ve Grabcut yöntemi ile kusur bölütleme yapılmıştır. En son aşamada ise ray kusurlarının konumları YoloV2 ile ikili görüntü üzerinde konumlandırılmıştır. Gan ve ark. [21] ray yüzey kusurlarının doğru bir şekilde belirlenmesi için arka plana dayalı bir kusur denetçisi önermiştir. Önerilen yöntem ile kusurların doğru bir şekilde bölütlenmesi sağlanmıştır. Tu ve ark. [22] ray ve traverslerin doğru bir şekilde tespiti için gerçek zamanlı çalışan YOLACT tabanlı bir yaklaşım önermiştir. Raylar bölütleme sonucu tespit edildikten sonra kusur tipi hafif ağırlıklı bir evrişimsel sinir ağı ile belirlenmiştir. Bağlantı elemanı kusurları ise elde edilen ikili görüntü üzerinde şablon eşleştirme ile belirlenmiştir. Ray yüzey kusurlarının tespiti için bölütleme ve kusur tespit adımlarını içeren bir yaklaşım önerilmiştir [23]. Ray kusur verileri dört sınıfa ayrılmakta ve kusur tipini daha doğru bir şekilde belirlemek için derin öğrenme tabanlı bir bölütleme yöntemi önerilmiştir. Kusurlu veri sayısının az olduğu durumlarda kusurları doğru bir şekilde tespit etmek için iki aşamalı derin öğrenme yöntemi önerilmiştir [24]. Rayın yatay olarak her bir satırı evrişimsel sinir ağı ve LSTM tabanlı bir modele verilerek kusurular belirlenmiştir. Zhuang ve ark. [2] ray yüzeyindeki kusurları belirlemek için derin öğrenme yaklaşımına dayalı bir çatı önermiştir. Önerilen yapıda ray çıkarımı yapıldıktan sonra DenseNet tabanlı bir özellik çıkarımı ile ray kusurları belirlenmiştir. Ray yüzey kusurlarını tespit etmek amacıyla çoklu öğrenme tabanlı bir yöntem önerilmiştir [4]. Ray yüzeyindeki kusurların yanı sıra kir ve belirsiz kusurlar da tespit edilmiştir. Önerilen yöntemin performansı YoloV5 ile karşılaştırılmış ve daha iyi sonuçların elde edildiği deneysel olarak ispatlanmıştır. Bir IHA ile toplanan ray görüntülerinden yüzey kusurlarının tespiti için görüntü işleme tabanlı bir yaklaşım önerilmiştir [25]. Önerilen yaklaşım görüntü iyileştirme amacıyla yerel Weber tabanlı bir yaklaşım önermiştir. Önerilen yaklaşımın temel

avantajı ray yüzeyindeki gölgeleme ışık problemlerini çözmesi ve kusur bölütleme işleminin performansını arttırmasıdır. IHA alınan görüntülerde ray kusurlarının tespiti için derin öğrenme tabanlı bir bölütleme yaklaşımı önerilmiştir [26]. Bölütlenen rayın özellikle kenarlarında oluşan çatlaklar FCN-8 tabanlı bir bölütleme algoritması ile bölütlendikten sonra dikdörtgensel alandaki piksel sayma yöntemi ile kusurlu bölge belirlenmiştir. İnsansız hava aracı ile alına görüntülerde ray kusurlarının belirlenmesi için derin öğrenme tabanlı demiryolu sınır kılavuzu yöntemini önermiştir [27]. Önerilen yaklaşımın etkinliği farklı derin öğrenme tabanlı bölütleme yaklaşımları ile karşılaştırılmıştır. Sistem çalışma hızı ve bölütleme performansı açısından literatüre göre sunmaktadır.

Demiryolu bağlantı elemanlarında oluşan kusurlar genellikle iki şekilde olmaktadır. Bunlar genellikle bağlantı elemanının tamamen çıkması veya bağlantı elemanının bir kısmının kırılmasıdır. Ray bağlantı elemanlarının tespiti ve kusur sınıflandırması için derin öğrenme ve görüntü işleme tabanlı bir yaklaşım önerilmiştir [5]. Önerilen yaklaşım görüntü işleme tarafında yoğun SIFT ve uzaysal piramit ayrıştırma tekniklerini kullanmaktadır. Derin öğrenme tarafında ise Faster R-CNN yöntemi kullanılarak kusurlar belirlenmiştir. Ray bağlantı elemanlarının incelenmesi için düşük ağırlıklı derinlik tahmin ağı önerilmiştir [28]. Ray bağlantı elemanlarının üç boyutlu derinlik görüntüleri elde edilerek kusurlar bu ağ üzerinden tespit edilmiştir. Ray bağlantı elemanın konumu YoloV3 ile elde edilmiş ve bağlantı elemanı kusur tipi sınıflandırılmıştır. Bağlantı elemanının konumunu belirlemek amacıyla çok ölçekli derin tespit ağı önerilmiş ve alan sınıflandırma ağı ise kusur tipini sınıflandırmak için geliştirilmiştir [29]. Bağlantı elamanı dört bölgeye ayrılarak kusur bölgelerin analizine göre kusur olup olmadığı belirlenmektedir. Bağlantı elemanlarındaki kusurları daha doğru bir şekilde sınıflandırmak amacıyla değiştirilmiş Faster RCNN modeli önerilmiştir [30]. Kusur sınıflandırma amacıyla destek vektör veri tanımı modeli kullanılmıştır. Bağlantı elemanlarının gerçek zamanlı tespiti için YoloV3-Tiny modelinin değiştirilmiş bir modeli önerilmiştir [31]. Önerilen model gerçek zamanlı olarak bağlantı elemanlarını standart YoloV3 modeline göre daha doğru tespit etmektedir. Demiryollarında kusurların tespiti ve analiz için ray kontrol robotlarının nasıl kullanılabileceği üzerine bir çalışma yapılmış ve literatürde kullanılan analiz araçları incelenmiştir [32]. Kusurlu bağlantı elemanlarının az olduğu durumda kusurlu bağlantı elemanı üretmek için Liu ve ark.[33] dört ayrıcı üretici ağ ile yapay veri üretmiştir. Elde edilen veri setlerinde kusur tespiti için VGG16 modeli transfer öğrenme olarak uygulanmıştır. Kusurlu bağlantı elemanlarını üretmek amacıyla başka bir çalışmada bağlantı elemanının bölütlendikten sonra kusur oluşturulması ve veri setinin artırımı sağlanmıştır [34].

Ray traverslerinde oluşan çatlakların belirlenmesi için görüntü işleme, sezgisel yöntemler ve özellik birleştirmeye dayalı bir yaklaşım sunulmuştur [6]. Bu amaçla Haar dönüşümü, entropi, integral görüntüsü ve kenar algılama kullanılmıştır. Yöntem büyük bir veri kümesinde uygulanarak başarılı sonuçlar elde edilmiştir.

Tablo 2.1.Literatürde ray bileşenleri ile ilgili yapılan çalışmalar

| Referans | Kusur oluşan Bileşen | Kusur tipleri | Yaklaşım | Görüntü alma platformu |
|---|---|---|---|---|
| [2] | Ray yüzey kusurları | - Kabuklanma<br>- Ezilme<br>- Taşlama<br>- Oluk hatası | DenseNet tabanlı özellik çıkarımı ve sınıflandırma | Ölçüm treni |
| [4] | Ray yüzey kusurları | - Parçalanma<br>- Ezilme<br>- Çizik<br>-Çatlak | Nesne Tespiti tabanlı kusur tipi belirleme | Ölçüm treni |
| [17] | Ray yüzey kusurları | - Ezilme | Dalgacık analizi tabanlı sinyal özelliklerinin çıkarımı | Manyetik bobin ve hall sensör |
| [18] | Ray yüzey kusurları | - Ezilme<br>- Parçalanma<br>- Çatlak | Curvature filtresi ve Gauss karşım modeli tabanlı bölütleme | Ölçüm treni |
| [19] | Ray yüzey kusurları | -Ezilme | Şartsal rastgele alanlar ve derin öğrenme tabanlı bölütleme | Kıyaslama veri kümesi |
| [20] | Ray yüzey kusurları | -Ezilme | Görüntü işleme tabanlı ray çıkarımı ve derin öğrenme tabanlı bölütleme | Ölçüm treni |
| [21] | Ray yüzey kusurları | -Ezilme | Görüntü işleme tabanlı bölütleme | Kıyaslama veri kümesi |
| [22] | Ray yüzey kusurları | - Çizikler | Derin öğrenme tabanlı ray tespiti ve kusur sınıflandırma | Ölçüm treni |
| [23] | Ray yüzey kusurları | - Kırık<br>- Leke<br>- Çatlak | Derin öğrenme tabanlı kusur sınıflandırma ve bölütleme tabanlı kusur tespiti | Ölçüm aracı |
| [24] | Ray yüzey kusurları | - Ezilme | İki aşamalı CNN+LSTM Ağı | Kıyaslama veri kümesi |
| [25] | Ray yüzey kusurları | - Ezilme | Lokal Weber tabanlı görüntü iyileştirme ve görüntü bölütleme | IHA |
| [26] | Ray yüzey kusurları | - Kırık | FCN8 tabanlı derin öğrenme yöntemi ile görüntü bölütleme | IHA |
| [27] | Ray yüzey kusurları | - Ezilme | Derin öğrenme tabanlı geliştirilen RGBNET ile görüntü bölütleme ve görüntü işleme tabanlı kusur tespiti | IHA |
| [28] | Bağlantı elemanı | - Eksik somun<br>- Eksik bağlantı elemanı | MobilenEtV2 tabanlı derinlik tahmin ağı ve | Ölçüm aracı |

| | | | | |
|---|---|---|---|---|
| | | - Kısmen kırık bağlantı elemanı | destek vektör makine tabanlı kusur sınıflandırma | |
| [29] | Bağlantı elemanı | - Kısmen kırık bağlantı elemanı | Bağlantı elemanın tespiti ve dört bölgeye ayrılarak kusur olup olmadığının derin öğrenme ile tespiti | Ölçüm aracı |
| [30] | Bağlantı elemanı | - Kısmen kırık bağlantı elemanı | Optimize edilmiş Faster RCNN ile kusur tespiti | Ölçüm aracı |
| [31] | Bağlantı elemanı | - Kusur tespiti yok | Değiştirilmiş Yolov3-Tiny modeli ile bağlantı elemanı tespiti | Ölçüm aracı |
| [33] | Bağlantı elemanı | - Eksik somun<br>- Eksik bağlantı elemanı<br>- Kısmen kırık bağlantı elemanı | Önerilen üretici ağ ile kusurlu veri üretimi ve kusur sınıflandırma | Ölçüm aracı |
| [34] | Bağlantı elemanı | - Eksik somun<br>- Eksik bağlantı elemanı<br>- Kısmen kırık bağlantı elemanı | Bölütleme tabanlı bağlantı elemanı tespiti ve oluşturulan kusurlar ile veri üretimi | Ölçüm aracı |
| [6] | Travers | -Travers çatlakları | Görüntü işleme tabanlı travers tespiti ve kusur tespitleri | Ölçüm treni |

Tablo 2.1'de gösterildiği gibi farklı kusur tipleri için genellikle bir ölçüm treninden veya özel tasarlanmış el ile sürülebilen bir ölçüm aracından alınan görüntüler kusur tespiti için kullanılmıştır. IHA ile yapılan çalışmalarda genellikle sadece veri toplama işlemi gerçekleştirilmiştir. IHA ile otonom uçuş yapan çalışmaların sayısı oldukça azdır. İHA ile demiryolu analizi için literatürde quadcopters İHA'lar kullanılmıştır [3]. Çalışmada demiryolu denetimi için tamamen otonom bir İHA platformu geliştirilmiştir. Bir PID kontrolörü, bir Parrot Mambo quadrotor üzerindeki demiryoluna bakan kamerayı kullanarak, görüş tabanlı bir yolu etkin bir şekilde takip edecek şekilde İHA'nın durumunu kontrol etmek için tasarlanmıştır. Bilgisayarla görme kullanılarak, İHA konumundan yolun merkez konumuna kadar olan mesafe bulunmuştur. Bu mesafeyi en aza indirmek için sapma kontrolü kullanılması ilgi alanını kameranın görüş alanında tutarak, görüşe dayalı bir yol takibi yöntemini ortaya çıkarmıştır. İHA'lar, iş verimliliğini ve üretkenliği artırma potansiyelleri açısından son yıllarda ticari olarak birçok sektörde kullanılmıştır. Demiryolu endüstrisi son zamanlarda İHA'ları keşfetmiş ve test etmeye başlamıştır. İHA'ların özellikle demiryolu hattını meşgul etmemesi ve yüksek kalitede görüntü alabilmeleri bu alanda kullanımlarını arttırmıştır. Araştırma, geleneksel bakım faaliyetlerinin büyük ölçüde verimsiz, fiziksel olarak zorlu ve tehlikeli olan pistlerde manuel olarak yürüyerek ya da pist doluluk gerektiren test/ölçüm araçları kullanılarak

gerçekleştirildiğini göstermiştir. İHA kullanımının temel olarak; uzaktan denetim ve işletmeye olanak sağlaması, yol doluluk veya yol ihtiyacı olmadan demiryolu denetimini yapabilmesi tercih edilmesindeki başlıca nedenlerden birkaçıdır. İHA'lar aynı zamanda bakım faaliyetlerinin çalışma koşullarını, verimliliğini ve kalitesini artırmaktadır. Flammini vd. [35] demiryollarında alt yapılarda oluşan problemleri belirlemek için IHA tabanlı bir sistem önermişlerdir. Bu amaçla farklı IHA yapılarının temel özellikleri verilerek bu alanda kullanılabilmesi ile ilgili teorik bilgiler verilmiştir.

Son yıllarda bilgisayarlı görü ve derin öğrenme alanındaki gelişmeler özellikle otonom bir şekilde çalışan sistemlerin geliştirilmesi için önemli bir araç haline gelmiştir. Evrişimli sinir ağları ilk olarak görüntü sınıflandırmada kullanılmasına rağmen sonraki çalışmalarda görüntü bölütleme ve nesne tespiti gibi alanlarda başarılı sonuçlar vermiştir. İHA tabanlı çözümlerin temel avantajı tren trafiğinin engellenmemesi ve optimum kapasitede trenlerin çalışmalarına izin verilmesidir. Fakat yapılan çalışmalardaki temel eksiklik otonom bir şekilde hareket eden bir sistemin olmamasıdır. Bu alanda yapılan bir çalışmada ufuk noktası tespit edilerek otonom iki çalışmada otonom olarak ray üzerinde hareket sağlanmıştır. Fakat çalışmalar temel görüntü işleme yöntemlerinden olan kenar çıkarım yöntemi ile raylar belirlenmiş ve ray takibi yapılmıştır. Fakat literatürde yapılan çalışmalarda derin öğrenme tabanlı otonom uçuş ile ilgili çalışmalar mevcut değildir. Ayrıca önemli bir demiryolu bileşeni olan traverslerdeki kaymalara odaklanan çalışma yapmış olduğumuz literatür araştırmasına göre bulunmamaktadır. Bu proje kapsamında hem görüntü işleme hem de derin öğrenme tabanlı ray takip ve görüntü toplama algoritmaları geliştirilmiştir. Ayrıca ufuk noktası tabanlı geliştirilen yöntemlerde özellikle Gabor filtresi kullanılarak gereksiz tespit edilen bileşenlerin kaldırılması sağlanmıştır. Ayrıca projede kullanılan Anafi4k İHA'sı Gazebo ortamında simüle edilmiştir. Projenin yöntem kısmında kullanılan yöntemler önce simülasyon ortamında test edildikten sonra gerçek uygulamaları yapılmıştır. Proje kapsamında ray yüzey kusurları, bağlantı elemanlarında oluşan kusurlar ve ray kaymaları gibi kusurları belirlemek için çalışmalar yapılmıştır. Aynı zamanda derin öğrenme tabanlı görüntü çoğaltma teknikleri ile veri setlerinde eksik olan görüntüler çoğaltılmıştır.

# 3. GEREÇ VE YÖNTEM

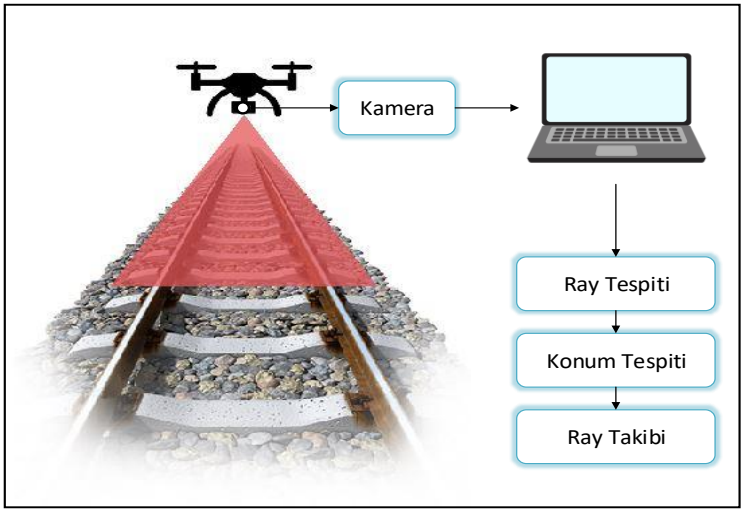
Proje kapsamında geliştirilen test edilmesi için simülasyon ve gerçek dünya ortamında düzeneklerin kurulması önemlidir. Projede, demiryolu hattına dair verilerin elde edilmesi amacıyla simülasyon ortamında geliştirilen otonom uçuş yöntemleri sayesinde sahadan veri toplayan bir sistem geliştirilmiştir. Demiryolu hattında oluşan arızaların belirlenmesi için demiryolu bileşenlerine ait görüntüler kullanılmıştır. Görüntü işleme uygulamalarının geliştirilmesi için demiryolu bileşenlerine ait sağlıklı ve kusurlu görüntülerinin toplanması gerekmektedir. Projede, İHA ile toplanan verilerin yanı sıra Demiryolları Araştırma ve Teknoloji Merkezi (DATEM) tarafından toplanan görüntüler de kullanılmıştır. Bu çalışmada, İHA üzerindeki yüksek çözünürlüklü kamera ile görüntüler elde edilmiştir. Şekil 3.1'de projede kullanılan İHA resimleri verilmiştir.

Şekil 3. 1. Sahada kullanılan Tello ve simülasyon ortamında kullanılan Anafi4K İHA'sı

Şekil 3.1'de kullanılan İHA'lar ile demiryoluna ait görüntüler kaydedilmektedir. Saha testlerindeki zaman ve maliyetten tasarruf etmek için veri toplama süreci öncelikle simülasyon ortamında gerçekleştirilmelidir. Simülasyon, bir sisteme ait neden sonuç ilişkilerinin bilgisayar ortamına daha detaylı bir şekilde aktararak sistemin davranışlarının bilgisayarda izlenmesini sağlar. Kısaca özetlemek gerekirse; gerçek veya teorik bir fiziksel sistemin modelini tasarlama, modeli yürütme ve yürütme çıktısını analiz etme işlemidir. Gazebo simülasyon ortamında geliştirilen otonom uçuş yöntemi sayesinde görüntüler elde edilir. Öncelikle elde edilen görüntülerden demiryolu hattı çıkarılır. Daha sonra kusur tespiti veya sınıflandırılması yapılmak istenen demiryolu bileşeni görüntüden çıkarılır. Kusurlu verinin azlığı nedeniyle veri artırım teknikleri ile yeni kusurlu veriler oluşturulur. Elde edilen demiryolu bileşenleri içerdikleri kusurlara göre yenilikçi derin öğrenme modelleri sayesinde sınıflandırılır. Ayrıca demiryolu bileşeni üzerindeki kusurlar derin öğrenme yöntemleri ile tespit edilir.

## 3.1. Otonom Ray Takibi

Otonom İHA'lar insan müdahalesi olmadan hareket edebilen araçlardır. Otonom İHA'lar çevreyi algılamak için çeşitli sensörlere sahiptir. Sensörlerden alınan verilere göre yol çıkarımı ve çevre algılaması yapabilirler. Bu sayede, herhangi bir insan müdahalesi olmadan çeşitli görevleri yerine getirebilirler. Bu çalışmada kullanılacak İHA, demiryolu boyunca otonom hareket edecektir. Bu şekilde insan yerine bir yazılım tarafından kontrol edilerek demiryolundan görüntü alınması sağlanacaktır. Önerilen algoritma üç bölümden oluşmaktadır. İlk aşamada alınan görüntülerden ray tespiti yapılır. İkinci aşamada ray hatları kullanılarak konum tespiti yapılır. Son aşamada ise, bulunan ufuk noktası kullanılarak demiryolunun otonom takibi sağlanır. Sistemimizin blok diyagramı Şekil 3.2'de verilmiştir.

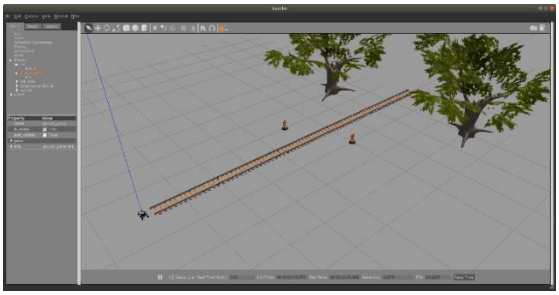



Şekil 3. 2. Otonom ray takibi için blok diyagramı

Otonom İHA'nın kontrolünde ray çizgilerini algılama önemli bir kısımdır. İHA demiryolunda ilerlerken ray çizgilerinin gerçek zamanlı olarak algılanması önemlidir. Ayrıca, ray çizgileri gerçek zamanlı tespit edilirken ray tespit algoritması hızlı ve güvenilir bir şekilde çalışmalıdır. Ray tespiti bilgisayarlı görmeye dayanan bir yöntemdir. Ray çizgileri, İHA'nın ön tarafında bulunan kamera sensörü ile tespit edilir. Kamera tarafından yakalanan karelerin genellikle yüksek bir kaliteye ihtiyacı vardır. Çünkü alınan görüntülerin canlı olması, yanlış bilgi alma olasılığını en aza indirecektir. Bu nedenle ray tespiti için en az 720p çözünürlüklü kamera içeren drone kullanılmalıdır.

### 3.1.1. Simülasyon Ortamında İHA ile Otonom Uçuş ve Ray Takibi

Gazebo, 3 boyutlu bir robot simülatörüdür [36]. Gazebo, çeşitli sensör modellerini, fizik motorlarını kullanmayı ve 3 boyutlu bir dünya ortamını simüle etmeyi sağlamaktadır [37]. Gazebo'nun önemli özelliklerinden birisi de İHA, ağaç, ev gibi popüler modelleri simülasyon ortamı için sağlamasıdır. Şekil 3.3'te Gazebo simülasyon ortamı görülmektedir.



Şekil 3. 3. Gazebo ortamı

Parrot Anafi4k Gazebo ortamıyla uyumlu çalışan bir İHA'dır. Bir adet ön kamerası mevcuttur. Demiryolu ray görüntüleri bu kamera ile kayıt altına alınmıştır. Yüksek menzilli kamera eğim açısı -90 ile +90 derece arası İHA üzerindeki nesneyi incelemeye izin vermektedir. Gazebo'da bir demiryolu ray modeli oluşturmak için 3D Warehouse'dan Collada formatında (.dae) 3 boyutlu bir model dosyası indirilmiştir [38]. Ardından, Gazebo ile uyumlu SDF dosyası oluşturulmuştur. Gerçek dünyada perspektiften bakıldığında demiryolu rayları ufukta buluşuyor gibi görünmektedir. Bu paralel hatların kesişme noktalarından izlenen demiryolunun ufuk noktası tanımlanabilir. Rayların görüntüden nasıl çıkarıldığına ilişkin prosedürün bir açıklaması aşağıda verilmiştir. Bir görüntüdeki çizgileri tespit edebilmek için, keskinleştirme, kenar algılama, yumuşatma gibi düşük seviyeli ön işleme tekniklerine ihtiyaç duyulmaktadır. Rayların tespiti için öncelikle Gabor filtresi kullanılmıştır. Gabor filtresi yardımıyla Anafi4k İHA'sından alınan canlı görüntüler üzerinde belli bir yöne uzanan ayrıtlar tespit edilmiştir.

Elde edilen görüntüdeki kenarları tespit edebilmek için Canny kenar çıkarım yöntemi kullanılmıştır. Canny algoritması John Canny tarafından geliştirilen popüler bir kenar bulma algoritmasıdır [39]. Canny algoritması 4 ana kısımdan oluşmaktadır: Girdi resim Gauss filtresi ile yumuşatılır, her piksel için kenar olma eğilimi hesaplanır, En büyük olmayan komşunun kenar olma eğilimi sıfırlanır ve en son aşamada çift eşik seviyesi uygulanır. Denklem (3.1) standart sapma değeri ile önceden tanımlanmış bir Gauss filtresini kullanarak görüntülerin kenarlarının algılanmasını sağlamaktadır [40].

$$Kenar\ Olma\ Eğilimi(G) = \sqrt{G_x + G_y} \tag{3.1}$$

Görüntüde bulunan raylar Hough dönüşümü kullanılarak çıkarılmıştır. Hough dönüşümü, bir görüntüdeki düz çizgileri ve parametrik eğrileri tespit etmek için kullanılır [41]. Denklem (3.2)'de Hough dönüşümü gösterilmiştir.

$$\rho = xcos\theta + ysin\theta \tag{3.2}$$

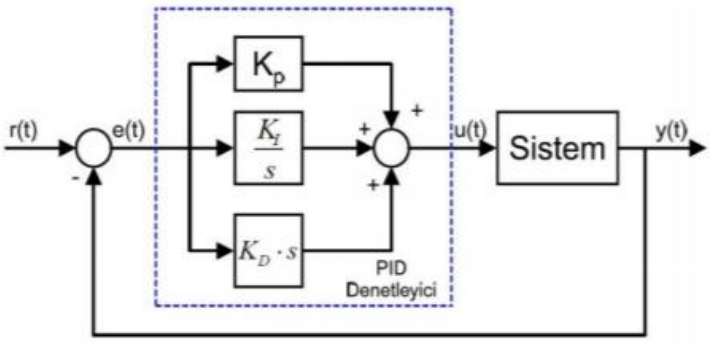
Denklem (3.2)'de kullanılan $\theta$ parametresi, doğru parçasının açı değerini temsil etmektedir [42]. Bu gösterimle, x-y düzlemindeki her nokta, ρ-θ parametre uzayında bir sinüzoide eşlenir. Parametre uzayında x-y düzleminde iki nokta eşdoğrusal ise, bunların sinüzoidleri Denklem (3.2)'ye göre doğruyu tanımlayan belirli bir noktada (ρ, θ) kesişecektir [43]. Algoritmanın genel fikri, birçok sinüzoidin kesiştiği uzay parametrelerindeki noktaları (ρ, θ) bulmaktır. Bu noktalar orijinal görüntüdeki çizgilere karşılık gelecektir [43]. Elde edilen Hough dönüşümünün başlangıç ve bitiş noktaları iki ayrı dizide tutularak en küçük kareler yöntemi ile ufuk noktası tahmini yapılmıştır. Intersect() fonksiyonu, parametre olarak iki ayrı diziyi alarak ufuk noktasının koordinatlarını döndürmektedir.

Tespit edilen ufuk noktası takipEt() fonksiyonuna parametre olarak verilip takip işlemi PID (Oransal, İntegral, Türev) denetleyici ile sağlanmaktadır. PID, endüstride yaygın olarak kullanılan kapalı döngü geri bildirim sistemi yapısıdır. Bir PID denetleyicisi sürekli olarak hatayı hesaplar ve bu hatayı oransal, integral ve türev terimleriyle düzeltmeye çalışır. Matematiksel olarak PID denetleyicisi Denklem (3.3) ve (3.4) şeklinde gösterilmektedir.

$$u(t) = K_p e(t) + K_i \int_0^t e(t) + K_d \frac{de(t)}{dt} \tag{3.3}$$

$$e(t) = r(t) - y(t) \tag{3.4}$$

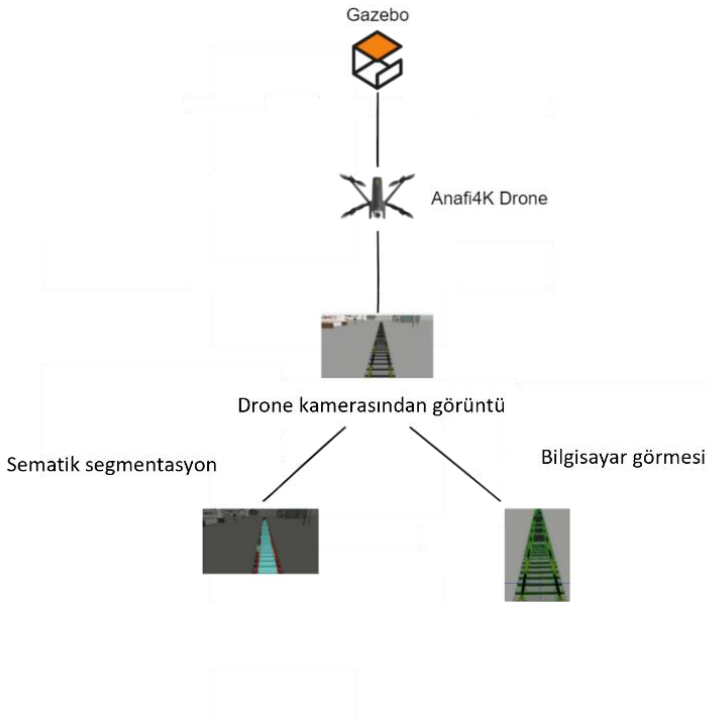
Denklem (3.3) ve (3.4)'te r(t) giriş, u(t) kontrol, e(t) hata ve y(t) sistemin çıktı sinyalidir [44]. $K_p$, $K_i$, $K_d$ katsayıları ise sırası ile oransal, integral, türev katsayılarıdır [44]. Şekil 3.4'te PID denetleyicisine ait şema görülmektedir. PID denetleticiyi sayesinde Anafi4k İHA yaw ekseninde anlık olarak hatayı hesaplayarak ufuk noktasını tahmin etmektedir.




Şekil 3. 4. PID denetleyicisi şeması

Bu çalışma, yapay görme tabanlı sistemlerin hızlı gelişimi, esnekliği ve kullanım kolaylığı sayesinde otomatik ve otonom ray algılama ve demiryolu altyapısı izleme ve denetimine olanak tanır. Çalışmanın temel amacı, gerçek dünya deneylerinde kullanıma hazır bir sistem tasarlamaktır. Gazebo simülasyon ortamında hazırlanan bu ortam, semantik segmentasyon

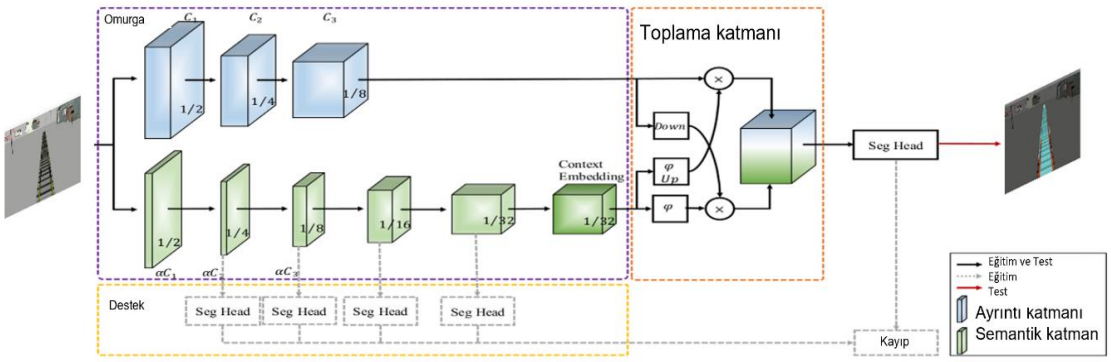
yöntemi ile demiryolu hattını gerçek zamanlı olarak tespit etmektedir. Ardından görüntü işleme teknikleri ile basit ve otonom olarak uçan drone, demiryolu hattında uçmaktadır. Şekil 3.5'te çalışmanın genel yapısı verilmiştir.




Şekil 3. 5. Çalışmanın genel yapısı

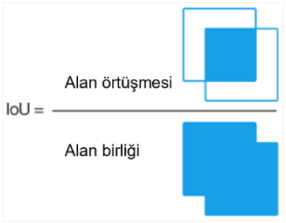
Literatürde yapılan çalışmalarda drone deneyleri genellikle gerçek dünya ortamında denenmiştir. Bu çalışmada kullanılan Anafi4k drone Gazebo ortamında simüle edilmiştir.
Gazebo hem iç hem de dış mekanlar için geliştirilmiş açık kaynaklı, ücretsiz, 3 boyutlu bir robot deney ortamıdır. Çardak, üç boyutlu bir ortamda çeşitli modelleri simüle etme yeteneğine sahiptir. Çalışmada Gazebo ortamında simülasyon ortamı için Anafi4K drone, ray, ağaç, posta kutusu, trafik işareti, bina gibi modeller eklenmiştir. Gazebo, deneysel ortamda kullanılabilecek fiziksel Anafi4K drone ile aynı olan bir Parrot Anafi4k modeli sağlar. Anafi 4K modelinde bir adet ön kamera bulunmaktadır. Demiryolu hattı, hat takip mantığı ile bu kamera ile takip edilmektedir. Şekil 3.5, Gazebo ortamında kullanılan Parrot Anafi4K drone modelini göstermektedir.
Ray takibi ile drone modelinin demiryolu hattını otonom olarak takip etmesi amaçlanıyor. Öncelikle Python ortamında drone kamerasından alınan gerçek zamanlı görüntü kırpılır. Buradaki amaç drone'un sadece ray kısmına odaklanmasını sağlamaktır. Daha sonra gerçek zamanlı görüntü gri tonlamaya dönüştürülür ve Gauss bulanıklığı uygulanır. Görüntüdeki bulanıklığı gidermek için Gauss kullanıldı. Görüntü, elde edilen görüntüye bir renk eşiği uygulanarak ikili bir değere dönüştürülür.

Bu çalışmada, BiSeNetV2 algoritması kullanılarak demiryolu görüntüleri için hızlı anlamsal bölütleme ray tespiti gerçekleştirilmiştir. Semantik bölütleme, bilgisayarlı görmede önemli bir alandır ve piksel düzeyinde bir görüntüyü anlamayı amaçlar. Semantik bölütlemenin temel amacı, görüntüdeki her pikseli belirli bir sınıfa etiketlemektir. Bu çalışmada, gerçek zamanlı anlamsal bölütleme için iki yönlü bölütleme ağı (BiSeNetV2) olarak adlandırılan iki yönlü bir mimari önerilmiştir. BiSeNetV2, yerelleştirme problemini anlamsal çıkarım probleminden ayırmanın ve ardından iki bilgiyi uygun şekilde birleştirmenin bir yolunu önerdi. Segmentasyon için önceden eğitilmiş bir model kullanıldı. Çalışma, BiSeNetV2'yi RailSem19 veri setinin yalnızca üç etiketli ("ray-yükseltilmiş", "ray-ray", "arka plan") değiştirilmiş bir versiyonu üzerinde eğitmiştir. BiSeNetV2, gerçek zamanlı anlamsal segmentasyon için oluşturulmuş iki yönlü bir mimaridir. İlk yol, Ayrıntı Dalı adı verilen geniş kanallar ve sığ katmanlarla uzamsal ayrıntıları yakalamak için tasarlanmıştır. Buna karşılık, dar kanallar ve derin katmanlarla kategorik anlambilimi çıkarmak için Semantik Dal adı verilen diğer yol tanıtıldı. Semantik Dal, anlamsal bağlamı yakalamak için geniş bir alıcı alana ihtiyaç duyarken, ayrıntılı bilgi Ayrıntı Dalı tarafından sağlanabilir. Bu nedenle, Semantik Dal, daha az kanal ve daha hızlı alt örnekleme stratejisi ile çok hafif hale getirilebilir. Her iki özellik temsili türü, daha güçlü ve daha kapsamlı bir özellik temsili oluşturmak için birleştirilir. BiSeNetV2'nin yapısı Şekil 3.6'da gösterilmektedir.




Şekil 3. 6. Detay dal ve semantik dalın yapısı

Detay Dal ve Semantik Dal, özellik temsilinde birbirini tamamlayıcıdır. Bu nedenle, bir Toplama Katmanı, iki tür özellik temsilini birleştirmek için tasarlanmıştır. Jaccard indeksi olarak da bilinen Intersection-Over-Union (IoU), anlamsal segmentasyonda en sık kullanılan performans metriklerinden biridir. IoU, Şekil 3.7'de gösterildiği gibi, tahmin edilen segmentasyon ile kesin bilgi arasındaki örtüşme alanıdır ve tahmini segmentasyon ile gerçek gerçeklik arasındaki birlik alanına bölünür. 0, örtüşme olmadığını ve 1 mükemmel örtüşen segmentasyonu gösterir.




Şekil 3. 7. IoU hesaplaması

### 3.1.2. HSV'ye Dayalı Ray Takip Algoritması

Yol çizgilerini algılama, yoldaki şeritleri tespit etmek ve her şeridin doğru konumunu ve şeklini bulma işlemidir. Yol çizgilerini algılama işlemi otonom sürüş sistemlerini mümkün kılan temel tekniklerden biri olarak kabul edilmektedir. Bu bölümde, demiryolu raylarının tespit edilebilmesi için bir görüntü işlemeye dayalı yöntem önerilmiştir. Önerilen yaklaşımda, raylar başarıyla tespit edilmiştir. Ray takibi iki ayrı drone ile yapılmıştır. Kullanılan dronlar Parrot Anafi 4K ve DJI Tello'dur. Ray tespiti için dronların üzerinde bulunan kameradan alınan görüntüler kullanılmıştır. DJI Tello kullanarak ray takibi yapılırken alınan görüntüye, görüntü ön işleme adımları uygulandıktan sonra Hough dönüşümü ile yol çizgileri bulunmuştur. Tespit edilen ray çizgileri ileriki bölümde yol takibi için kullanılmıştır. Parrot Anafi'de ise kamera 90 derece aşağı bakacak şekilde ayarlanarak iki ray çizgisi belirlenip rayların orta noktası kullanılarak ray takibi yapılmıştır.

HSV renk uzayı, görüntü segmentasyonu, nesne tanıma, özellik algılama ve görüntü analizinde yaygın bir biçimde kullanılmaktadır [45]. RGB formatındaki görüntüyü HSV formatına dönüştürmek için görüntünün R, G ve B bileşenlerinin çıkarılması gerekir. RGB'den HSV'ye dönüşüm için gerekli denklemler aşağıda verilmiştir.

$$\mathrm{H} = \arccos \frac{\frac{1}{2}(2\mathrm{R} - \mathrm{G} - \mathrm{B})}{\sqrt{(\mathrm{R} - \mathrm{G})^2 - (\mathrm{R} - \mathrm{B})(\mathrm{G} - \mathrm{B})}} \tag{3.5}$$

$$\mathrm{S} = \frac{\max(\mathrm{R}, \mathrm{G}, \mathrm{B}) - \min(\mathrm{R}, \mathrm{G}, \mathrm{B})}{\max(\mathrm{R}, \mathrm{G}, \mathrm{B})} \tag{3.6}$$

$$\mathrm{V} = \max(\mathrm{R}, \mathrm{G}, \mathrm{B}) \tag{3.7}$$

Çalışmanın bu bölümünde ilk olarak, RGB formatlı görüntü HSV formatına dönüştürülmüştür. Eşik seçimi, renge göre bölütleme işleminde önemli bir faktördür. Görüntü HSV renk uzayına dönüştürüldükten sonra, yalnızca ilgilenilen renge göre algılama yapılması gerekmektedir. Bu

nedenle yol çizgilerinin, üst ve alt renk sınırları belirlenip renge göre maskeleme işlemi yapılmıştır.

Renge göre maskeleme işleminin ardından ikili görüntü elde edilmiştir. İkili görüntünün kullanım açısından birçok avantajı vardır. Renkli bir görüntü, hedef nesne dışında, arka planda bulunan nesneleri de içerir. Ancak, renge göre maskeleme işleminden sonra elde edilen ikili görüntüde raylar 1'e (beyaz) rayların çevresi ise 0'a (siyah) dönüştürülmüştür. Böylece bu aşamadan sonra yalnızca ikili görüntüler üzerinde işlem yapılacak ve bu da hesaplama hızını artırarak ray çizgilerini belirlemenin olası bir yolu olacaktır.

Elde edilen görüntüde eşikleme işleminin ardından yol çizgilerinin daha belirgin olabilmesi için bazı morfolojik işlemler uygulanmıştır. Morfolojik işlemler, genellikle ikili görüntülere uygulanır. İkili görüntüler çok sayıda kusur içerebilir. Morfolojik işlemler, görüntünün biçimini ve yapısını hesaba katarak bu kusurları ortadan kaldırmak için kullanılır. Bu çalışmada, genişletme (dilation) ve açma (opening) işlemleri uygulanmıştır. Morfolojik işlemlerin ardından elde edilen görüntüden kenar çıkarımı yapılmıştır.

Kenarlar, bir görüntünün önemli bilgilerini gösterir. Kenar algılama sonuçları, görüntü analizini doğrudan etkileyecektir. Canny algoritması, John F. Canny tarafından 1986'da önerilen, aşamaları olan popüler bir kenar algılama algoritmasıdır. Genel kenar algılama algoritmasıyla karşılaştırıldığında, çoğu durumda Canny algoritması en iyi performansa sahiptir [46].

Algoritmada ilk aşama görüntü üzerindeki gürültünün azaltılması işlemidir. Bu işlem görüntüdeki gürültüyü 5x5 Gauss filtresi ile gidermektedir. Daha fazla Gauss filtresi ile daha az kenar algılanır. Algoritmanın ikinci aşaması ise görüntünün yoğunluk gradyanını bulma işlemidir. Gauss filtresi uygulanan görüntü daha sonra, birinci türevi bulmak için hem yatayda (Gx) hem de dikeyde (Gy) bir Sobel çekirdeği ile filtrelenir. Bu iki görüntüden, her piksel için kenar gradyanı ve yönü Denklem 3.8 ve 3.9'daki gibi bulunur.

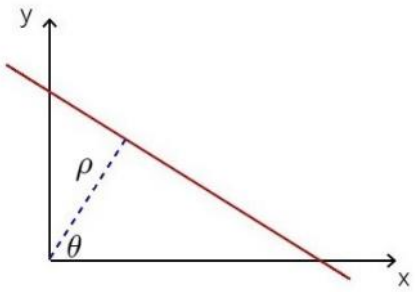
$$\text{Edge Gradient}(\mathrm{G}) = \sqrt{\mathrm{Gx}^2 + \mathrm{Gy}^2} \tag{3.8}$$

$$\text{Angle}(\theta) = \tan^{-1}\left(\frac{\mathrm{Gx}}{\mathrm{Gy}}\right) \tag{3.9}$$

Gradyan büyüklüğü ve yönü bulunduktan sonra, hesaplanan açı değeri bilgisi kullanılarak kenarı oluşturmayabilecek istenmeyen pikselleri kaldırmak için tam bir görüntü taraması yapılır. Ardından ikili eşikleme ile kenarlar daha belirgin hale getirilir.

Bu çalışma için uygulanan ve değerlendirilen tüm kenar dedektörlerinden en iyi kenar görüntülerini üreten kenar bulma algoritması, Canny kenar bulma algoritmasıdır. Canny oparatörü, daha düşük hata oranlarına, daha yüksek hassasiyete ve tek yanıta sahiptir. Bu nedenle, kenar çıkarım işlemi için "Canny" kenar çıkarma algoritması kullanılmıştır.

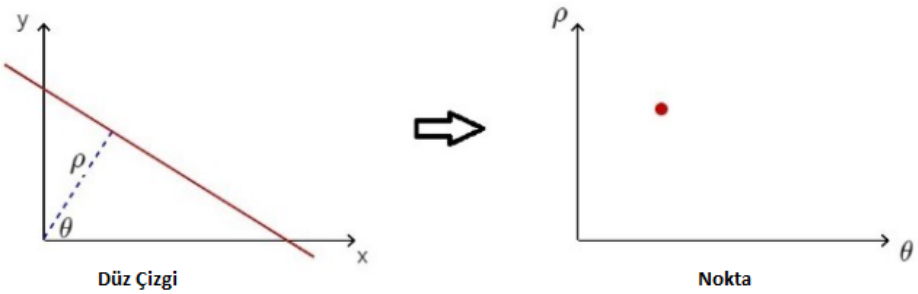
Hough dönüşümü, görüntü analizinde kullanılan bir özellik çıkarma yöntemidir. Hough dönüşümü, çizgiler, daireler, elipsler gibi herhangi bir düzenli eğrinin özelliklerini belirlemek için kullanılabilir. Görüntü uzayındaki çizginin denklemi y = mx + c biçimindedir, burada m eğimdir ve c doğrunun y kesişimidir. Ancak, bu denklem dikey çizgileri bulamaz. Bu nedenle kutupsal sistemde (ρ, θ) çifti kullanılarak doğru açık bir şekilde tanımlanmalıdır. Burada ρ, başlangıçtan çizgiye en kısa mesafedir. θ ise, x ekseni ile mesafe çizgisi arasındaki açıdır (Şekil 3.8). Böyle temsil etmenin faydalarından biri, dikey çizgileri ρ ve θ ile tanımlayabilmemizdir. Kartezyen sistemde sadece (a, b) parametrelerini kullanarak böyle bir tanımlama mümkün değildir. Belirli bir doğru için, belirli ρ ve θ değerlerini belirlenir. Ardından, bu çizgiye ait her bir x, y noktası için Denklem 3.10'daki eşitlik sağlanır.




Şekil 3. 8. Çizgiye en kısa mesafe ve oluşan açı

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{3.10}$$

Çizgi, belirlenen ρ, θ koordinatlarıyla Hough uzayında tek bir nokta olarak ifade edilir (Şekil 3.9). Bu şekilde görüntü uzayında ortak bir noktada kesişen başka çizgiler de varsa bu noktalar sinüs yapısı oluşturur. Hough uzayında bir sürü sinüs yapısı elde edilir. Bu yapıların kesişimi ile düz çizgiler tespit edilir.

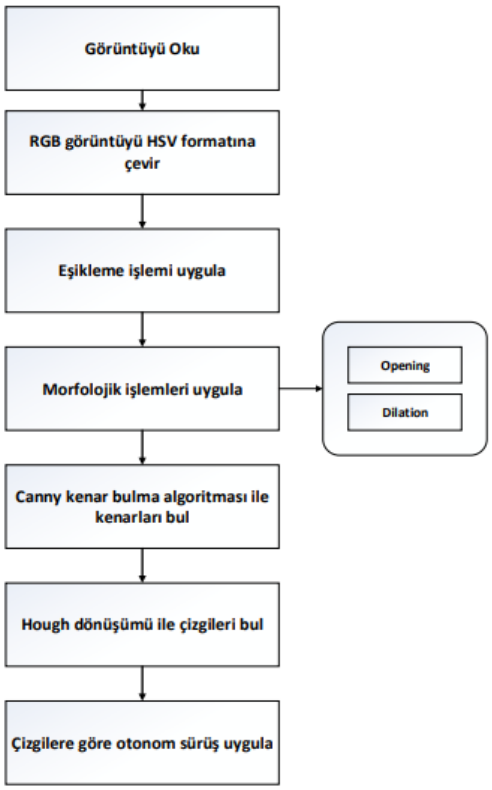



Şekil 3. 9. Tek nokta ile ifade edilmesi

### 3.1.2.1. DJI Tello ile Ray Takibi İçin Önerilen Yöntem

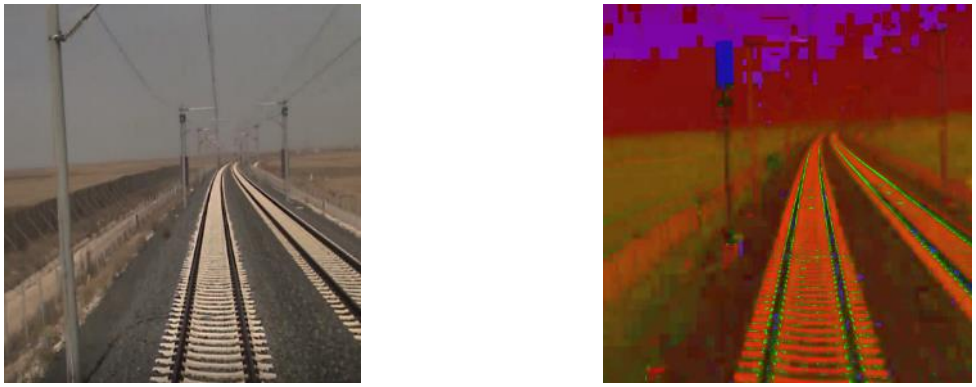
Bu çalışmada, ray tespiti için histogram eşiğine dayalı yöntem kullanılmıştır. Bu renk tabanlı yöntem, HSV renk uzayına ve histogram eşiğine dayanmaktadır. İlk olarak İHA üzerinde bulunan kameradan yol görüntüsü alınmıştır. Alınan renk tabanlı görüntü bölütleme işlemi için HSV renk formatına dönüştürülmüştür.

Demiryolundan alınan görüntüler RGB renk uzayındadır. Önerilen yaklaşımda, RGB renk uzayındaki ray görüntüsü HSV renk uzayına dönüştürülür ve ardından dönüştürülen görüntü yoğunluk ve renge bağlı olarak üç farklı bileşene bölünür. Her üç bileşen için (ton, doygunluk ve değer) histogram hesaplanır. Ardından, en düşük ve en yüksek eşik değeri seçilir ve eşik değerine göre ray görüntüsü çıkarılır. Ancak eşikleme işlemi tek başına yeterli olmayabilir. Elde edilen görüntü gürültülü bir görüntü olabilir. Böyle durumlar için morfolojik işlemler uygulanması gerekir. Morfolojik işlemlerden sonra maske elde edilir. Önerilen yaklaşımın akış diyagramı Şekil 3.10'da verilmiştir. Bu yöntemin ray görüntüsünün çıkarılmasında etkili olduğu gösterilmiştir.




Şekil 3. 10. Önerilen yöntemin akış diyagramı

Yapılan çalışmada İHA üzerine bağlı kameradan alınan görüntüler üzerinde işlemler yapılmıştır. Şekil 3.10'da verilen akış diyagramına göre raylar tespit edilmiştir. Alınan görüntü örneği Şekil 3.11.a' verilmiştir. Alınan görüntü HSV formatına çevrilmiştir (Şekil 3.11.b). HSV formatlı görüntüye yapılan eşikleme işlemi sonuçları Şekil 3.12.a'da verilmiştir. Eşikleme sonucu elde edilen görüntüye morfolojik işlemler uygulanmıştır (Şekil 3.12.b).



a) b)

Şekil 3. 11. a) Alınan görüntü b) HSV formatlı görüntü

a) b)

Şekil 3. 12. a) Eşikleme işlemi sonucu b) Morfolojik işlemler sonucu

Morfolojik işlemlerden sonra kenar çıkarma işlemi uygulanmıştır. Sonuçları Şekil 3.13.a'da gösterilmiştir. Kenar çıkarma Hough dönüşümü için yapılmaktadır. Hough dönüşümü, bir görüntüdeki belirli bir şeklin özelliklerini göstermek için kullanılan bir tekniktir. Hough dönüşümü, yaygın olarak çizgiler, daireler, elipsler gibi düzenli eğrilerin saptanması için kullanılır. Hough dönüşümü, bu çalışmada ray çizgilerini belirlemek için kullanılmıştır. İşlem sonrası Şekil 3.13.b'de verilen sonuç alınmıştır.

a) b)

Şekil 3. 13. a) Görüntüye uygulanan kenar çıkarma sonucu b) Hough Dönüşümü

#### 3.1.2.1.1. Ufuk Noktası Tespiti

Ufuk noktası tespiti, bilgisayarlı görme alanında önemli bir araştırma konusudur ve otonom bir İHA'nın görsel navigasyon sistemi için önemli bir bileşendir. Ufuk noktası tespiti, robot navigasyonu, hedef yeniden yapılandırma, kamera kalibrasyonu gibi alanlarda kullanılabilir. Ufuk noktası tespit edilerek, İHA'ların otonom navigasyon sistemlerinin en önemli parçası tamamlanabilir. Bu işlem için İHA'dan alınan görüntü kullanılarak ufuk noktası tahmin edilmelidir. Ufuk noktası tahmini sonrasında İHA'nın yönlendirilmesi ufuk noktasına göre yapılmalıdır.

Literatür incelendiğinde, ufuk noktası tahmini için kullanılan çeşitli teknikler vardır. Ancak kullanılan tekniklerin birçoğu yüksek hesaplama maliyetleri nedeniyle gerçek zamanlı kullanıma uygun değildir. Bu nedenle, değer çalışmalarda kıyaslandığında hızlı çalışan çizgilerin kesişimine dayalı yöntem kullanılmıştır. Rayların kesintisiz çizgi oluşturması kenar tabanlı ufuk noktası algılama yönteminin uygulanmasını kolaylaştırmaktadır. Önerilen yöntemde amaç, ray çizgilerine ait özellikleri bulmaktır.

### 3.1.2.2. Parrot Anafi ile Otonom Ray Takibi

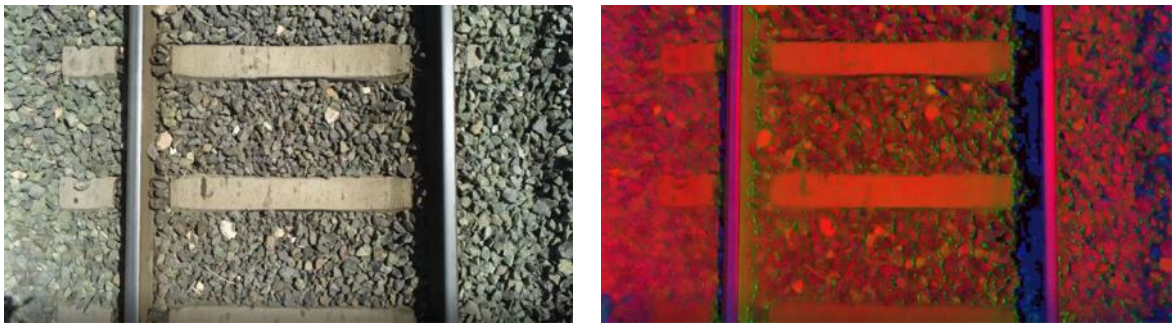
#### 3.1.2.2.1. Ray çizgilerinin tespiti için önerilen yaklaşım

Rayların tespiti için HSV'ye dayalı renk bölütleme kullanılmıştır. İHA tarafından alınan RGB görüntü 90 derece açıdan alındığı için sadece demiryolunu içermektedir. Böyle bir görüntüde, HSV tabanlı ray tespiti iyi sonuçlar vermektedir ve hız açısından avantajlıdır. Bu nedenle alınan görüntü ilk olarak HSV renk uzayına dönüştürülmüştür. Elde edilen sonuçlar Şekil 3.14'de gösterilmiştir. Ardından, raylara ait ikili maske elde edilmiştir.

Renk özü, doygunluk ve parlaklık değerlerine ait düşük eşik ve yüksek eşik değerleri uygulanmıştır. Uygulanan eşik değerleri, Tablo 3.1'de verilmiştir. Eşikleme işlemi uygulandıktan sonra rayın renk değerini içeren pikseller 0 olarak belirlenecek, kalan pikseller ise 1 olacaktır.

Tablo 3. 1. Ray tespiti için eşik değerleri

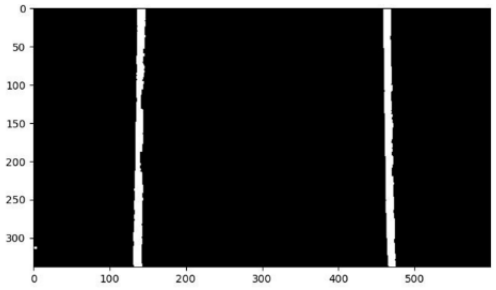
| **Kanal** | **H** | **S** | **V** |
|---|---|---|---|
| En Küçük | 85 | 0 | 61 |
| En Büyük | 121 | 255 | 255 |



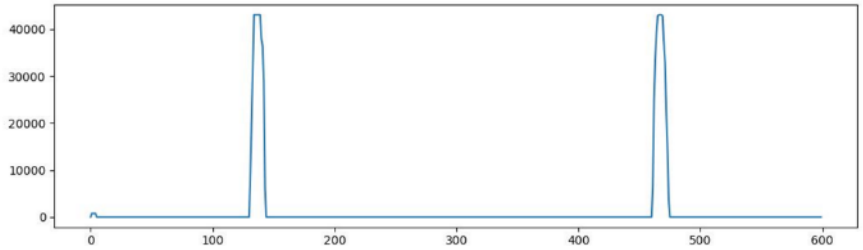
a) b)

Şekil 3. 14. a) RGB renk uzayı b) HSV renk uzayı

Alınan görüntüye eşikleme yapıldıktan sonra, Şekil 3.15.a'da görüldüğü gibi ray çizgilerini içeren net bir şekilde öne çıktığı ikili bir görüntü elde edilmiştir. Bu aşamadan sonra, ikili maskede bulunan beyaz piksellerin hangisinin sağ raya hangisinin sol raya ait olduğunun belirlenmesi gerekmektedir. Bu nedenle, beyaz pikselleri algılamak için bir histogram oluşturulmuştur. Şekil 3.15.b'de görüldüğü gibi histogram çizildiğinde iki tepe noktası elde edilecektir. Bu tepe noktaları, sol ray çizgisine karşılık gelen bir sol zirve ve sağ şerit çizgisine karşılık gelen bir sağ zirveyi belirtmektedir.




a)




b)

Şekil 3. 15. Ray konumlarının histogram tabanlı tespiti a) İkili görüntü b) Histogram pik noktaları

#### 3.1.2.2.2. İHA konumunun belirlenmesi

Parrot Anafi 4K'nın kamerası raya 90 derece ayarlandığı için alınan görüntüde raylar dikey bir biçimde görünmektedir. Bu durumda İHA'nın bulunması gereken konum iki rayın orta noktası olarak hesaplanabilir. Şekil 3.15.b'teki histogram grafiğinde görüldüğü gibi raylar iki adet pik noktası oluşturmaktadır. Elde edilen pik noktalarından sağ ve sol rayı belirlemek için ekranın yatay uzunluğu ikiye bölünmüştür. Yatay uzunluğun yarısı referans nokta olarak alınmıştır. Referans noktasının solunda kalan pik noktası sol rayı, sağında kalan pik noktası ise sağ rayı ifade etmektedir. İHA'nın bulunması gereken noktanın hesaplanması için aşağıdaki denklemler kullanılarak hesaplanmaktadır. İlk olarak sağ pik noktasının x koordinatı ile sol pik noktasının x koordinatı arasındaki fark Denklem 3.7'deki gibi hesaplanmıştır.

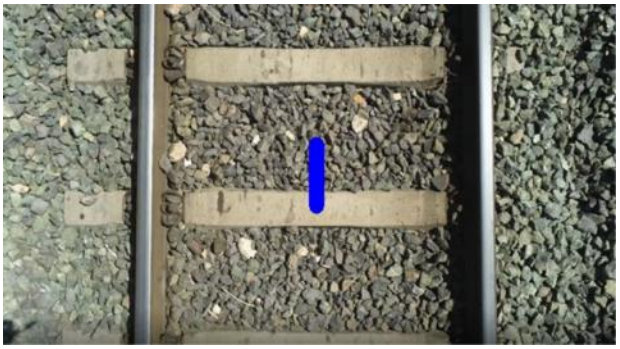
$$\text{fark} = \text{sagx} - \text{solx} \tag{3.7}$$

Ardından, aradaki farkın yarısı bulunarak sol pikselin x koordinatına eklenmesi gerekmektedir. Bunlar için denklemler aşağıda verilmiştir.

$$\text{orta} = \frac{\text{fark}}{2} \tag{3.8}$$

$$\text{konum} = \text{sol} + \text{orta} \tag{3.9}$$

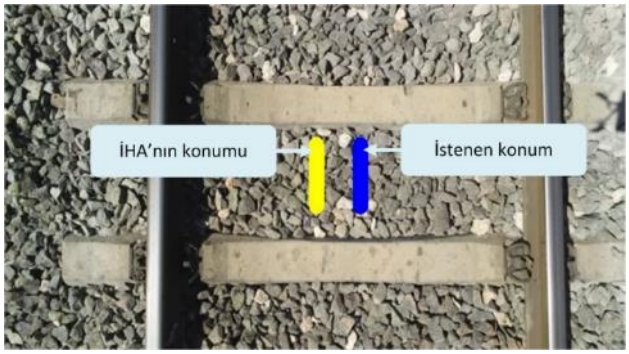
Şekil 3.16’da görüldüğü gibi İHA’nın o anda bulunması gereken bulunması gereken konum işaretlenmiştir. Rayların orta noktası ve İHA’nın anlık konum bilgisi arasındaki fark PID denetleyiciye aktarılacaktır. PID denetleyici ile her görüntü çerçevesi için oluşan hata değerine göre İHA kontrolü sağlanacaktır.



Şekil 3. 16. İHA'nın bulunması gereken konum

#### 3.1.2.2.3. Ray takibi

İHA’nın rayı takip etmesi, İHA’nın bulunduğu konumun x koordinatı ile istenen konum arasındaki mesafenin sıfıra indirilmesi ile sağlanacaktır. İHA’nın bulunduğu konum, kameradan alınan görüntünün x koordinatının orta noktasıdır. İstenen konum Denklem 3.9’da hesaplanan konumdur. Hesaplanan konumlar, her görüntü çerçevesi üzerinde Şekil 3.17’de gösterildiği gibi işaretlenmektedir. İstenen konum ile İHA’nın konumunun farkı sıfırda tutulurken İHA, ileri yönlü sabit hızla ilerleyecektir. Demiryolu takibi bu şekilde sağlanacaktır.




Şekil 3. 17. İHA’nın bulunduğu konum ile istenen konumun belirlenmesi

### 3.1.3. Gabor Filtresi ve Bulanık PID Tabanlı Ray Takibi

Ray görüntüsüne uygulanan Hough dönüşümü sonucunda elde edilen çizgiler ufuk noktasında kesişecektir. Gürültü gibi etkenler nedeniyle bu kavşakların tamamı aynı noktada olmayacaktır. Bu nedenle, bu kesişim noktalarından hangisinin gerçek ufuk noktası olduğunu belirlemek

zordur. Ufuk noktası tahmininin amacı bu kesişimlerden tek bir nokta elde etmektir. Bu nedenle, çizgilerin kesişiminde en küçük kareler çözümü kullanılmalıdır [47].
2 boyutlu bir çizgi üzerindeki nokta $a = (a_0, a_1)^T$ olarak ifade edilir. Yön vektörü $n = (n_0, n_1)^T$ olarak ifade edilir. Böylece çizgi $p = a + tn$ olarak gösterilir. Diyelim ki K-hattımız var. Tüm bu çizgiler ufuk noktasında kesişecek. Dolayısıyla birden fazla kesişme noktası. Tek kesişme noktası, en küçük kareler yöntemi kullanılarak bulunacaktır. Kare mesafelerin toplamı en aza indirilecektir. K çizgilerinin mesafesi şu şekilde hesaplanır:

$$\mathrm{D(p: A, N)} = \sum_{\mathrm{i=1}}^{\mathrm{K}} \mathrm{(p; a_i, n_i)} = \sum_{\mathrm{i=1}}^{\mathrm{K}} \mathrm{(a_i - p)^T \left(I - n_i n_i^T\right)(a_i - p)} \tag{3.10}$$

$$\mathrm{p'} = \mathrm{argmin D(p; A, N)} \tag{3.11}$$

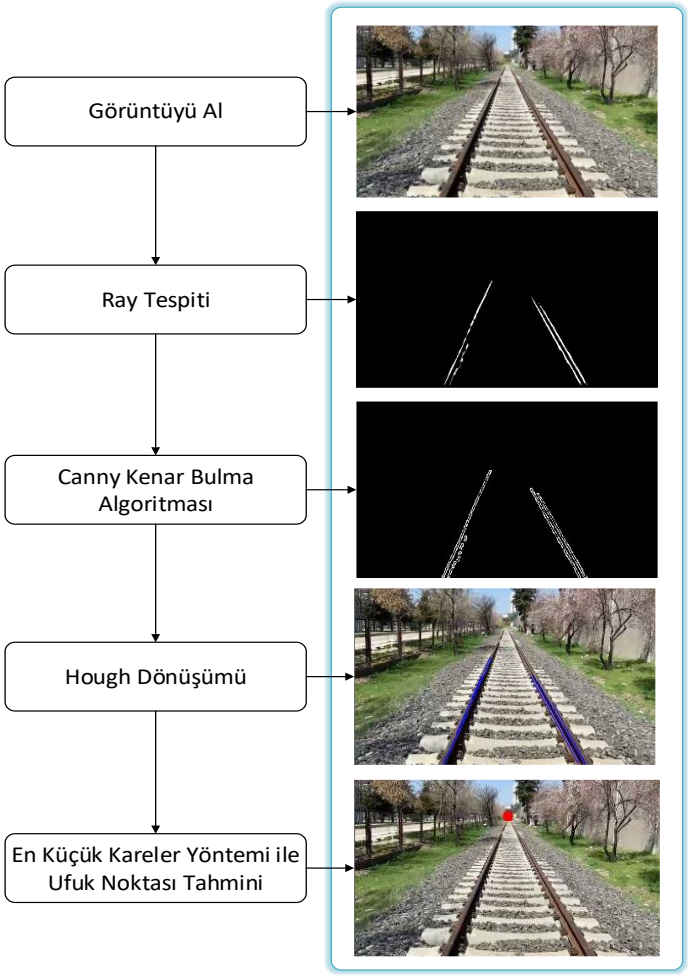
p'ye göre türev alınırsa aşağıdaki ifade elde edilir:

$$\frac{\partial \mathrm{D}}{\partial \mathrm{p}} = \sum_{\mathrm{i=1}}^{\mathrm{K}} -2\left(\mathrm{I} - \mathrm{n_i n_i^T}\right)\left(\mathrm{a_i} - \mathrm{p}\right) = 0 \tag{3.12}$$

Aşağıdaki doğrusal denklem yeniden düzenlenerek elde edilir.

$$\mathrm{R} = \sum_{\mathrm{i=1}}^{\mathrm{K}} \left(\mathrm{I} - \mathrm{n_i n_i^T}\right), \mathrm{q} = \sum_{\mathrm{i=1}}^{\mathrm{K}} \left(\mathrm{I} - \mathrm{n_i n_i^T}\right)\mathrm{a_i} \tag{3.13}$$

Ufuk noktası tahmini için uygulanan algoritma Şekil 3.18'de verilmiştir.



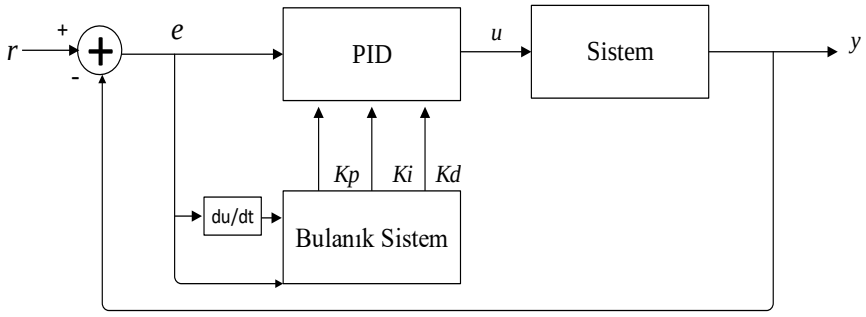

Şekil 3. 18. Ufuk noktası tahmini için uygulanan algoritma

#### 3.1.3.1. Bulanık PID Tabanlı Ray Takibi

Uygun PID parametrelerinin seçimi ile PID kontrolörün başarısı artırılabilir. Doğru PID kontrol parametrelerinin seçilmemesi performansı büyük ölçüde etkileyecektir. Bu nedenle, PID kontrol parametrelerinin ayarlanırken dikkat edilmelidir. PID kontrol parametreleri genellikle deneyimli uzmanlar tarafından belirlenir. Bu çalışmada, PID kontrol parametrelerini ayarlamak için bulanık mantık kullanılmıştır. Bulanık sistem için ilk olarak kurallar belirlenmiştir. Daha sonra, bu kurallar, PID kontrol parametrelerini çevrimiçi olarak ayarlamak için kullanılmıştır. Bulanık-PID denetleyicisi, PID denetim parametrelerini düzenlemek için bulanık sistem çıktısını kullanmaya dayanır. Bu yönüyle, geleneksel bir PID denetleyiciden farklıdır. Bu teknikle optimize edilmiş bulanık PID denetleyicisi, sistemdeki ani değişikliklere gerçek zamanlı olarak hızlı dinamik yanıt almayı sağlar.

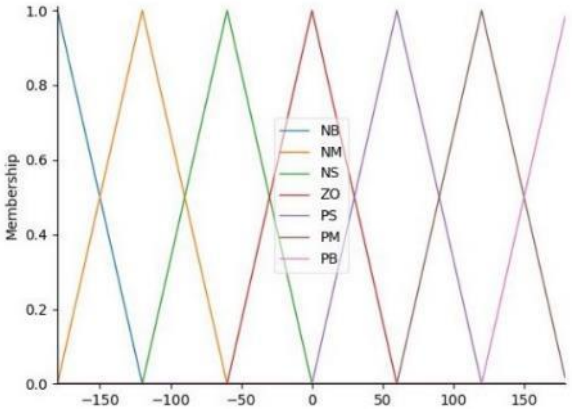
Kontrol parametrelerinin kendi kendine ayarlanmasıyla, sistemin çıktısına göre optimum değerler belirlenir. Bulanık PID denetleyici, sistemin mevcut hata değerine göre kontrol parametrelerini çevrimiçi olarak ayarlayabilir. Bu işlem, kontrolün hassasiyetini arttırır. Bu

nedenle, geleneksel PID denetleyiciden daha iyi bir performans sağlanır. Sistemin blok diyagramı Şekil 3.19'daki gibidir.

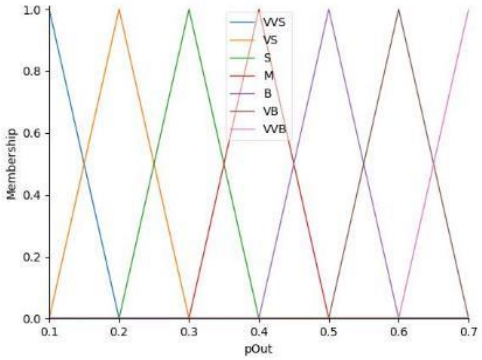



Şekil 3. 19. PID kontrol parametrelerini ayarlamak için kullanılan bulanık sistemin blok şeması

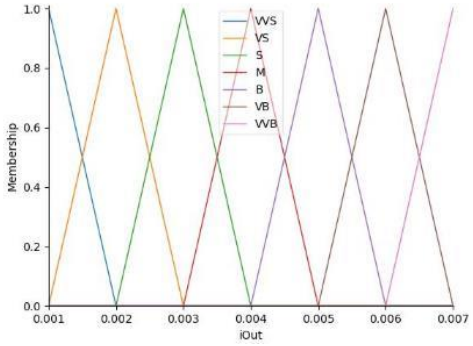
Bulanık sistem, hata (e) 'yi ve hatanın (ed) türevini girdi olarak alır. Hatanın türevi ile hata arasındaki ilişkiye göre Kp, Ki ve Kd değerleri için çıktı verir. Sitemde hata (e) ve hatanın türevi (ed) [-180,180] aralığında seçilmiştir. Bu aralığın seçilmesinin nedeni, İHA'dan alınan görüntünün yeniden boyutlandırılmasıyla ilgilidir. Çıktı olarak Kp [0.1, 0.8], Ki [0.001, 0.008] ve Ki [0.11, 0.18] seçilmiştir. Şekil 3.20, hata (e) ve hatanın türevi (ed) için üyelik fonksiyonlarını göstermektedir. Şekil 3.21, Kp, Ki ve Kd değerlerinin çıktısı için üyelik fonksiyonlarını gösterir. Ek olarak, kurallar Tablo 3.2 ve Tablo 3.3'te gösterilmektedir.



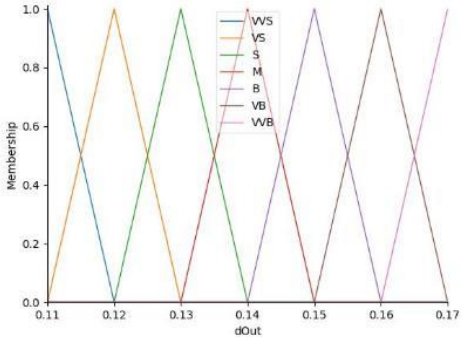

Şekil 3. 20. Hata (e) ve hatanın türevi (ed) için üyelik fonksiyonları




a) Kp




b) Ki




c) Kd

Şekil 3. 21. a) Kp için üyelik işlevleri b) Ki için üyelik işlevleri c) Kd için üyelik işlevleri

Tablo 3. 2. Kp ve Ki için kurallar

| **Kp,Ki** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Hatanın Türevi** | **Hata** | | | | | | |
| | **NB** | **NO** | **NK** | **N** | **PK** | **PM** | **PB** |
| **NB** | O | K | ÇK | ÇÇK | ÇK | K | O |
| **NO** | B | O | K | ÇK | K | O | B |
| **NK** | ÇB | B | O | K | O | B | ÇB |
| **N** | ÇÇB | ÇB | B | O | B | ÇB | ÇÇB |
| **PK** | ÇB | B | O | K | O | B | ÇB |
| **PO** | B | O | K | ÇK | K | O | B |
| **PB** | O | K | ÇK | ÇÇK | ÇK | K | O |

Tablo 3. 3. Kd için kurallar

| Kd | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Hatanın Türevi** | **Hata** | | | | | | |
| | **NB** | **NO** | **NK** | **N** | **PK** | **PO** | **PB** |
| **NB** | O | B | ÇB | ÇÇB | ÇB | B | O |
| **NO** | K | O | B | ÇB | B | O | K |
| **NK** | ÇK | K | O | B | O | K | ÇK |
| **N** | ÇÇK | ÇK | K | O | K | ÇK | ÇÇK |
| **PK** | ÇK | K | O | B | O | K | ÇK |
| **PO** | K | O | B | ÇB | B | O | K |
| **PB** | O | B | ÇB | ÇÇB | ÇB | B | O |

### 3.1.4. Yolact Tabanlı Ray Takibi

Otonom ray takibi algoritmasında rayların tespiti için derin öğrenme kullanılma amacı, İHA'nın ray çizgilerini algılamasını iyileştirmektir. Önerilen sistem, gerçek zamanlı ray çizgilerini algılar ve raylara ait maskeleri elde eder. Elde edilen maske bilgisi otonom ray takibi algoritmasında kullanılacaktır.

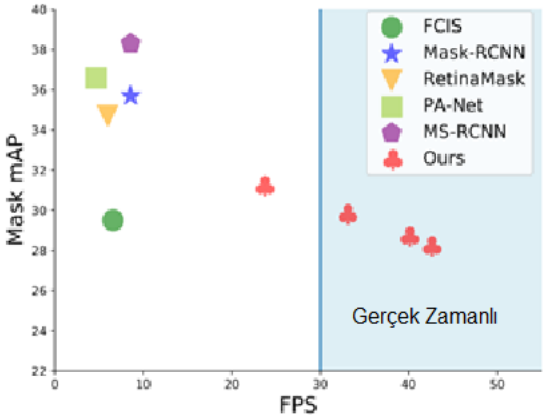
Son yıllarda, bilgisayar görmesi alanında örnek segmentasyonunda büyük ilerlemeler kaydedilmiştir. Mask RCNN ve Unet [48] gibi örnek segmentasyonuna yönelik son teknoloji yaklaşımlar, bu gelişmelerden bazılarıdır. İki aşamalı nesne tahmin algoritmalarını genişleterek ortaya çıkmıştır. Ancak, bu yöntemler hız yerine performansa odaklandığından gerçek zamanlı çalışma için uygun değildir. İki aşamalı tahmin algoritmaları, yüksek doğruluğa ancak düşük performansa sahiptir.

Hâlihazırda önerilen algoritmaların yavaş olmasının sebebi, örnek segmentasyonunun zor bir görev olmasıdır. YOLO [49] gibi tek aşamalı nesne tespit algoritmaları, Faster R-CNN gibi mevcut iki aşamalı dedektörleri, sadece ikinci aşamayı kaldırarak ve kayıp performansı başka yollarla telafi ederek hızlandırabilirken örnek segmentasyonunda böyle bir işlem kolayca uygulanamaz. Ayrıca, iki aşamalı tahmin algoritmaları, nesnelerin maskelerini oluşturmak için özellik yerelleştirmeye bağlıdır.

Bu çalışmada, gerçek zamanlı ray takibi yapılacağı için gerçek zamanlı çalışabilen bir segmentasyon yöntemi olan Yolact [50] kullanılmıştır. Yolact, örnek segmentasyonunu iki paralel görev ile yapar. İlk olarak giriş görüntüsü üzerinde prototip maskelerinden oluşan bir sözlük oluşturur. Ardından, örnek başına bir dizi doğrusal kombinasyon katsayısını tahmin

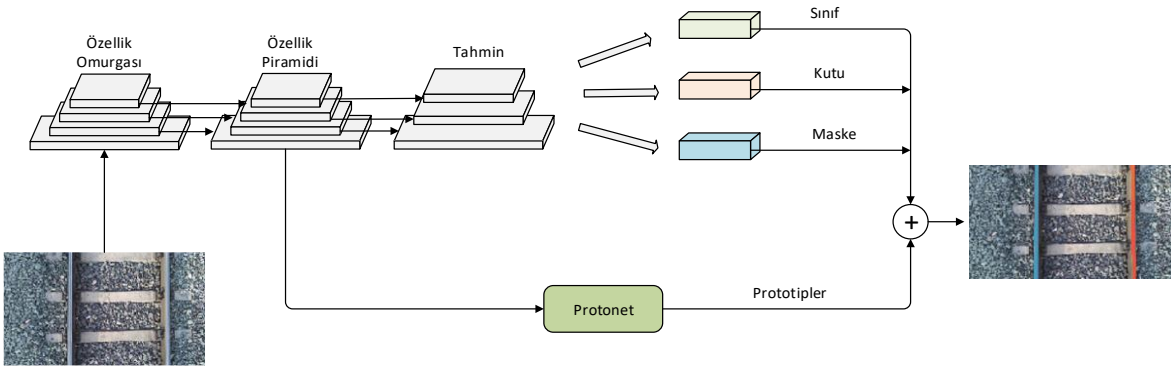
eder. Bu iki bileşenden segmentasyon oluştururken her örneğe karşılık gelen tahmini katsayıları kullanarak prototipleri doğrusal olarak birleştirir. Son aşamada ise tahmin bir sınırlayıcı kutu ile kırpılır.

Yolact, MS COCO veri kümesinde [50] test edildiğinde gerçek zamanlı kullanıma uygun örnek segmentasyon algoritması olduğu ispatlanmıştır. Diğer örnek segmentasyon algoritmaları ile hız-performans açısın karşılaştırılıp sonuçlar Şekil 3.22'de verilmiştir [50].




Şekil 3. 22. COCO veri kümesinde çeşitli örnek segmentasyon yöntemleri için hız-performans karşılaştırması

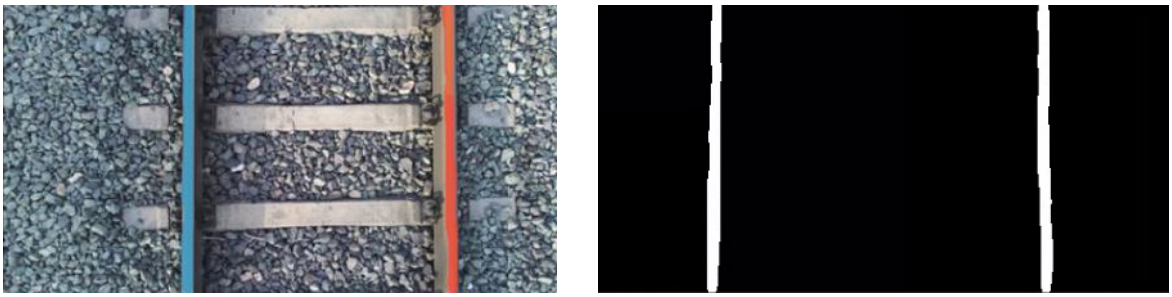
Yolact mimarisi Şekil 3.23'te gösterilmiştir. Özellik çıkarımı oluşturmak için farklı bir omurga kullanan FPN yöntemi ile başlar. Bunlar, ReLU [50] ve maske katsayılarını kullanarak prototip oluşturma dalları oluşturmak ve ardından bunları maske montajı için birleştirmek için kullanılır. Ardından, puan eşiğinde kullanılan birçok parametre görüntüyü filtreler ve bu parametreler çıktı görüntüsü olur ve nesnede bölümlenmiş bölgeler olarak görüntülenir.




Şekil 3. 23. Yolact mimarisi

Önerilen Yolact tabanlı ray tespit sistemini tasarımında ilk olarak İHA, el ile kontrol edilerek demiryolundan görüntü toplanmıştır. Görüntü toplama aşamasından sonra veriler ön işleme adımından geçirilir ve etiketlenir. Bu süreçten elde edilen veri kümesi, sistemi eğitmek için

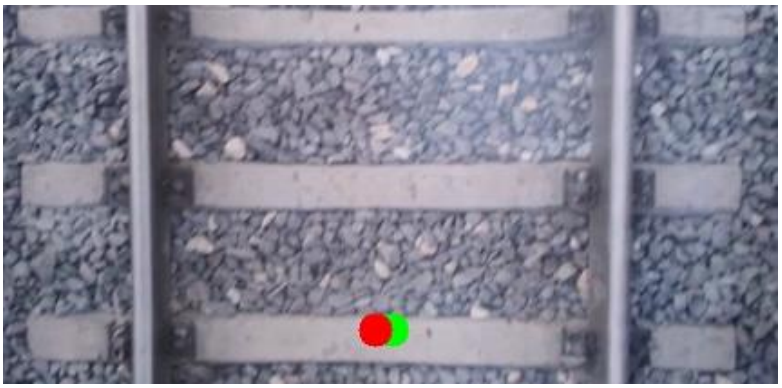
kullanılır. Sistemin yeterli doğrulukta algılama yapması için uzun bir eğitim sürecinden geçmiştir. Eğitim işlemi bittikten sonra, model doğruluk açısından kontrol edilmiştir. Elde edilen sonuç gerçek etiketle eşleştiğinde modelin kullanılabilir olduğuna karar verilmiştir. Elde edilen sonuç Şekil 3.24'te verilmiştir.



a) b)

Şekil 3. 24. Sonuçlar a) Maskelerin elde edilmesi b) İkili maskenin oluşturulması

Orta nokta tespiti için ilk olarak Şekil 3.24'te görülen sağ ve sol ray konturlarının merkez noktası hesaplanmıştır. Ardından elde edilen sağ ve sol raya ait merkez noktaların x koordinatlarının orta noktası kırmızı nokta ile işaretlenmiştir. Her görüntü çerçevesi için işaretlenen konum bilgileri, Şekil 3.25'te gösterilmiştir. Burada kırmızı nokta rayların orta merkezini gösterirken yeşil nokta referans noktasını gösterir. Referans noktası kameranın orta noktası olan 240. pikseldir. Tespit edilen iki ray çizgisinin orta konumu ve kameranın konumu kullanılarak, İHA'nın raylara göre konumu hesaplanmıştır. Bu işlem sonucu, İHA'nın merkezinin rayların ortasından ne kadar uzakta olduğunu hesaplanmıştır ve PID denetleyici ile ray takibi yapılmıştır.

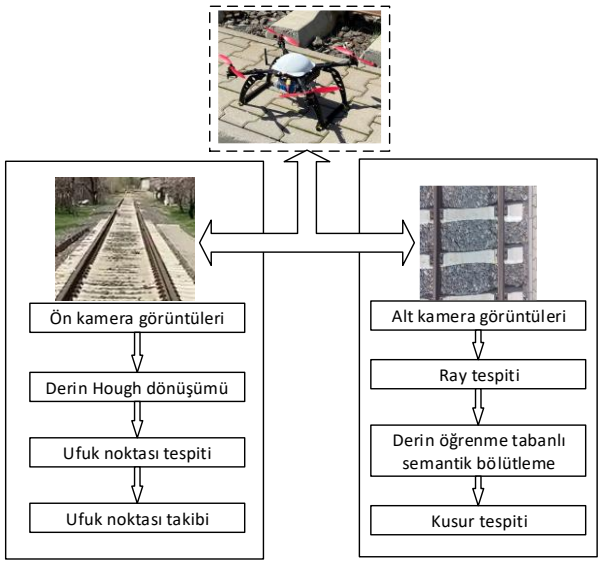


Şekil 3. 25. Orta nokta ve referans noktasının işaretlenmesi

### 3.1.5. Derin Hough Dönüşümü Tabanlı Otonom Ray Takibi ve Kusur Tespiti

Önerilen yöntem ilk olarak ray tespiti için derin Hough dönüşümünü kullanır. İHA'nın ön kamerasından alınan görüntüler kullanılarak derin Hough dönüşümü tabanlı iki ray belirlenerek

ufuk noktaları tespit edilir. İHA'nın ufuk noktasını takibi için PID tabanlı bir algoritma önerilmiştir. İHA'nın altında bulunan kamera ile elde edilen görüntülerden ray konumları tespit edilerek yüzey kusurları derin öğrenme tabanlı belirlenmektedir. Önerilen yöntemin blok diyagramı Şekil 3.26'da verilmiştir.

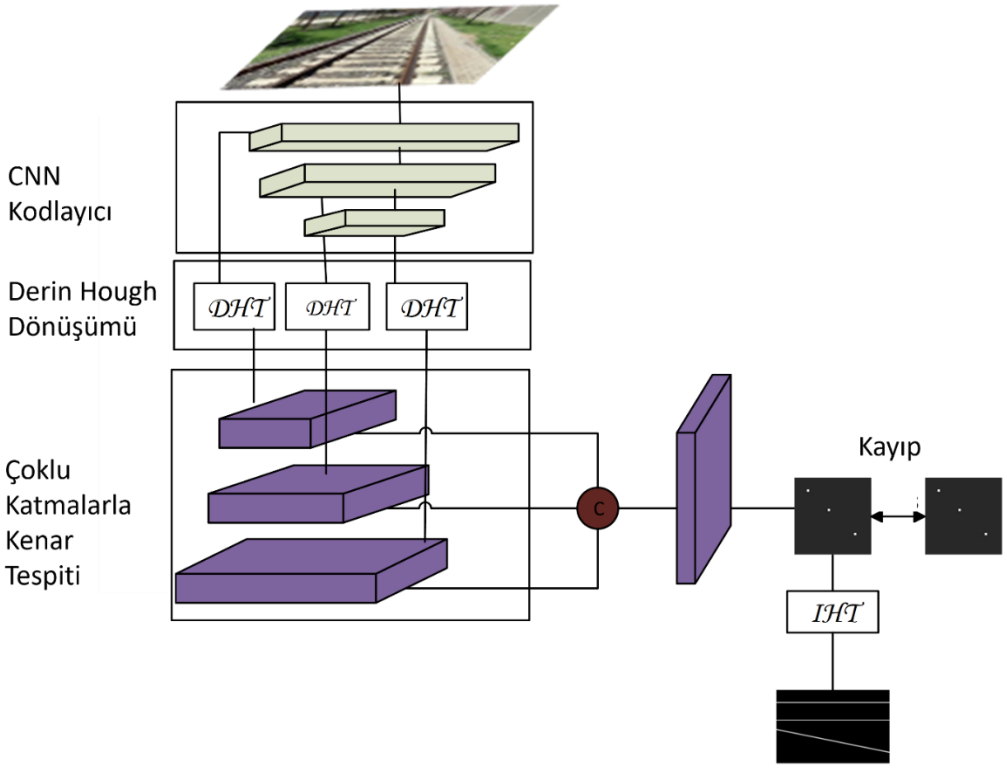



Şekil 3. 26. Önerilen derin öğrenme kontrollü kusur tespit sistemi

Şekil 3.26'da verilen sistem iki kameradan alınan görüntülere göre iki farklı algoritma çalışmaktadır. Ön kameradan alınan görüntülerden derin Hough dönüşümü ile raylar belirlenip otonom uçuş sağlanırken alt kameradan elde edilen görüntülerde ise ray yüzey görüntüleri kaydedilmektedir. Görüntü toplama işleminden sonra elde edilen ray görüntüleri kullanılarak ray yüzey kusurları bölütleme tabanlı derin öğrenme ile tespit edilmektedir.

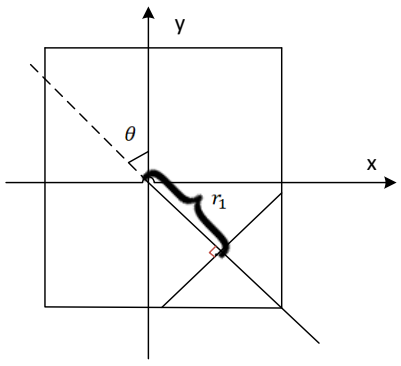
Hough dönüşümü kenar çıkarımı elde edilen görüntülerden çizgileri bulmak amacıyla geliştirilen bir yöntemdir [51]. Hough dönüşümü bilgisayarlı görü ve resim işleme için bir çizgi tespit algoritması olup kenar çıkarımı ile elde edilen ikili görüntülerde çizgileri bulak için kullanılır. Ayrıca Hough dönüşümü çizgilerin dışında elips ve çember gibi şekillerin tespitinde kullanılmıştır. Hough dönüşümünün çizgi tespiti üzerindeki başarımına rağmen yüksek hesaplama gücü istemesi, bulunan çizgilerin birbirinden ayrık olması ve çizgilerin bitiş noktalarının belirlenememesi en önemli dezavantajlarıdır. Ayrıca Hough dönüşümünün başarımı temelde kenar çıkarımı için kullanılan yönteme bağlıdır. Görüntü içerisindeki çizgilerin semantik olarak tespiti için çizgi segment tespit edici ve evrişimsel sinir ağları gibi teknikler geliştirilmiştir. Bu yöntemler her bir çizgi boyunca derin özellikleri birleştiren çizgisel öznitelikleri kullanmakta olup bu teknik yeterli kavram bilgisine sahip değildir. Derin Hough dönüşümü

Zhao ve ark. [52] tarafından önerilen ve semantik çizgilerin tespitinde kullanılan bir yöntemdir. Yöntemin genel şeması Şekil 3.27'de verilmiştir.




Şekil 3. 27. Derin Hough Dönüşümü Adımları

Şekil 3.27'de verilen derin Hough dönüşümü dört adımdan oluşmaktadır. İlk adımda bir evrişimsel enkoder piksel tabanlı derin gösterimi sağlar. Daha sonra derin Hough dönüşümü gösterimleri parametrik domain'e dönüştürür. Üçüncü aşamada parametrik uzayda çizgiler tespit edilerek dördüncü aşamada ters Hough dönüşümü ile tespit edilen çizgiler tekrar görüntü uzayına dönüştürülür. Verilen 2 boyutlu bir görüntüde görüntünün merkezi orijin olarak alınır. Bu uzayda bir l çizgisi x ekseni ile l çizgisi arasındaki açıyı ifade eden yönlendirmesi ve mesafe parametresi $r_l$ ile gösterilir. Şekil 3.28'de bahsedilen parametreler verilmiştir.




Şekil 3. 28. Hough dönüşümünde çizginin gösterimi

Şekil 3.28'de uygun parametreler belirlendikten sonra bu parametrelere göre semantik çizgiler belirlenmektedir. Derin Hough dönüşümü ray çizgilerini belirledikten sonra rayların kesiştiği nokta ufuk noktası olarak belirlenir. İHA'nın ufuk noktasını takip etmesi için bir PID sistem tasarlanmıştır [53]. PID sistemin denklemi aşağıda (3.14)'te verilmiştir.

$$u(t) = K\left(e(t) + \frac{1}{Ti}\int_0^t e(\tau)d\tau + Td\frac{de(t)}{dt}\right) \quad (3.14)$$

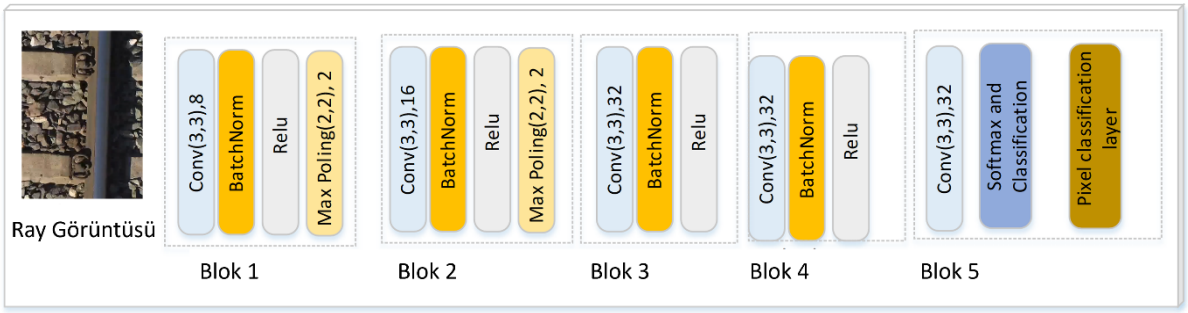
Burada referans nokta ufuk noktası olup ufuk noktası ile orta nokta arasındaki hatanın minimum yapılması istenir. Şekil 3.29'da derin Hough dönüşümü ile elde edilen ufuk noktası gösterilmiştir.

Şekil 3. 29. Derin Hough dönüşümü ile rayların ve ufuk noktasının tespiti

Şekil 3.29'da Hough dönüşümü ile elde edilen çizgiler iki boyutlu uzayda ufukta kesişmektedir. Bu kesişim noktası takip edilerek İHA'nın rayı takibi sağlanır.

### 3.1.5.1. Derin Öğrenme Tabanlı Ray Kusur Tespiti

İHA'nın ön kamerasından alınan görüntülerde ray takibi yapılırken, alt kameradan alınan görüntüler üzerinde ray kusur algoritmaları çalışmaktadır. Bu amaçla semantik bölütleme tabanlı bir kusur tespit algoritması önerilmiştir [54]. Önerilen derin öğrenme tabanlı bölütleme algoritmasının blok diyagramı Şekil 3.30'da verilmiştir.

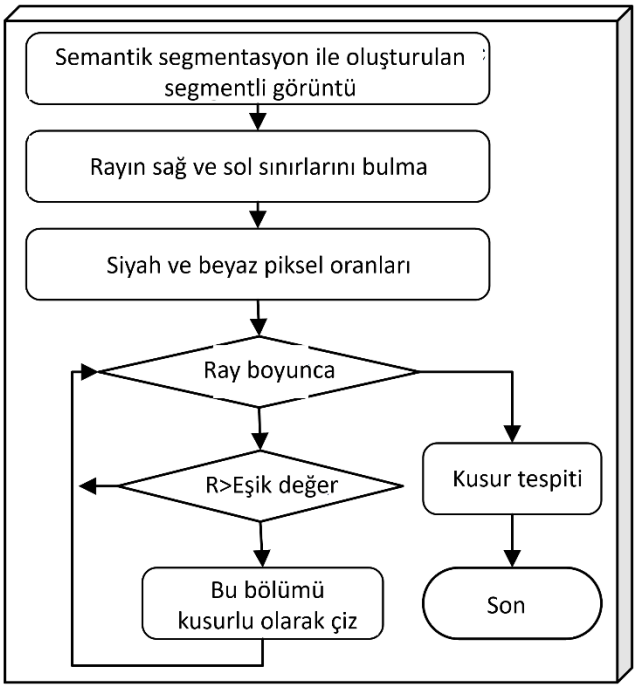



Şekil 3. 30. Derin Öğrenme Tabanlı Bölütleme Algoritması

Şekil 3.30'da toplamda 16 katmanlı bir semantik bölütleme derin öğrenme yapısı verilmiştir. Her bir blokta evrişim, batch normalizasyon, relu ve max pooling katmanı bulunmaktadır. En son katmanda ise piksel sınıflandırma katmanı ile bölütleme işlemi yapılmaktadır. Görüntü bölütleme algoritmasının etkinliği Jaccard ölçütüne göre hesaplanmaktadır. Rayı çıkarmak için denklem 3.14'teki indeks kullanılmıştır.

$$IoU = \frac{P_M \cap P_P}{P_M \cup P_P} \tag{3.14}$$

Denklemde $P_M$ bölütlenmiş görüntüde gerçekte rayı ifade eden pikselleri gösterirken $P_P$ ise derin bölütleme algoritması ile elde edilen pikselleri ifade eder. Daha sonra tespit edilen rayın kontürü ile olması gereken dikdörtgensel alan elde edilerek ikisi arasındaki fark alınarak kusur oluşup oluşmadığı belirlenir. Algoritma Şekil 3.31'deki gibi çalışmaktadır.

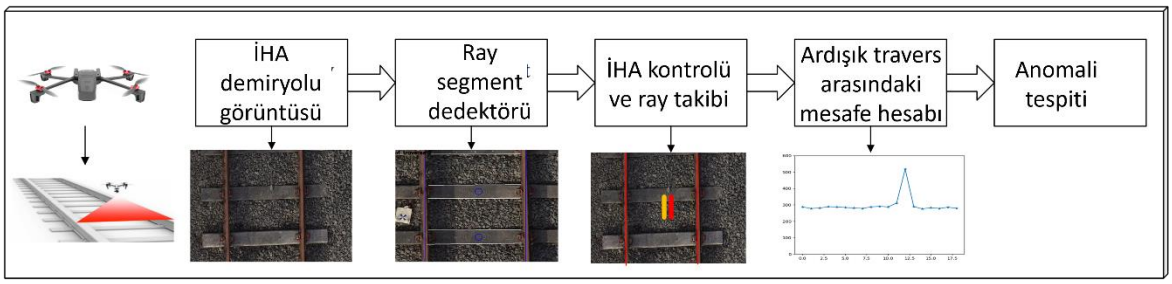



Şekil 3. 31. Kusur tespit algoritması

Şekil 3.31'de rayın sol ve sağ sınırları belirlendikten sonra her bir sütun için rayın siyah piksel oranı bir eşik değerden büyük ise kusurlu sütun olarak işaretlenir. Bu işlem bütün ray boyunca devam ettirilerek kusurlu bölgeler çıkarılmaktadır. Bölütleme tabanlı kusur tespit algoritmasının çalışma performansı kullanılan eşik değerine bağlıdır. Bu eşik değer çok küçük seçilirse bölütleme sonucu oluşan hatalı küçük alanların kusur olarak belirlenme ihtimali vardır. Bu değer büyük seçildiğinde ise küçük kusurların tespit edilmesi zorlaşacaktır. Bu yüzden eşik değeri ray genişliğinin bir oranı olarak alınmıştır.

### 3.1.6. Hafif Hat Segmentasyon Yaklaşımı ile Otonom İHA Tabanlı Ray Takip ve Travers Muayenesi

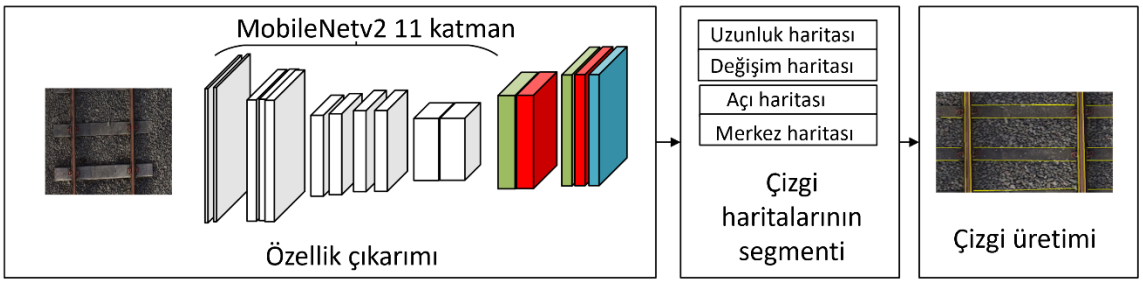
Bu çalışmada, otonom bir İHA ile bir demiryolu hattında travers bileşenlerinde ray takibi ve anormallik tespiti için görüş tabanlı bir yaklaşım önerilmiştir. Önerilen yaklaşım, ilk olarak, demiryolu hatları için derin öğrenmeye dayalı hafif bir hat bölütleme yöntemi sağlar. Sıralı traversler arasındaki mesafe, otonom İHA ile ray takibi yapılırken elde edilen görüntülerden hesaplanır ve zaman serisi olarak kaydedilir. Traverslerde meydana gelen anomali değişiklikleri, zaman serileri analiz edilerek belirlenir. Şekil 3.32, önerilen yöntemin blok diyagramını göstermektedir.




Şekil 3. 32. Travers durumu için önerilen yöntemin şeması

Şekil 3.32'de öncelikle İHA'nın alt görüş kamerasından alınan görüntüler üzerinde çizgi bölütleme algoritması kullanılarak yatay ve dikey çizgiler belirlenir. Ardından İHA, rayların takibi için PID kontrol algoritması ile kontrol edilebilir. Bu amaçla İHA merkezi ile iki rayın orta noktası arasındaki fark minimum olmalıdır. Aynı zamanda ardışık travers bileşenlerin pozisyonları bir zaman serisi olarak kaydedilmekte ve uçuş sırasında meydana gelen anormallikler zaman serilerinden elde edilmektedir.

Otonom yol takibi ve travers tespiti için hafif bir hat segmentasyon yöntemi kullanılmıştır. Bu yöntemin omurgası, hafif bir MobileNetv2 ve azaltılmış bir hat tahmin sisteminden oluşur [55]. Önerilen MobileNetv2 sistemi, kodlayıcı-kod çözücü mimarisine dayanmaktadır. Şekil 3.33, önerilen hat segmentasyon ağının mimarisini göstermektedir. Kod çözücü blokları ayrıca MobileNetV2'nin darboğaz katmanlarını kullanır. Bu katmanların eklenmesiyle ağdaki parametre sayısı 0,56 milyon olur. Daha sonra hat segment haritaları oluşturulur. Bu çalışmada hem travers hem de ray için elde edilen hatlar iki farklı amaç için kullanılmıştır. Ray için elde edilen hatlar raylar için kullanılmaktadır. Her iki ray için de birçok hat tespit edilebilir. Bu hatlardan sağ ve sol raylar için tek bir hat elde etmek için bulunan hatların x ekseninde ayrı ayrı ortalaması alınır. Travers için çok yakın çizgilerden biri ortadan kalkar. Bir sonraki aşamada PID tabanlı ray takibi yapılır.



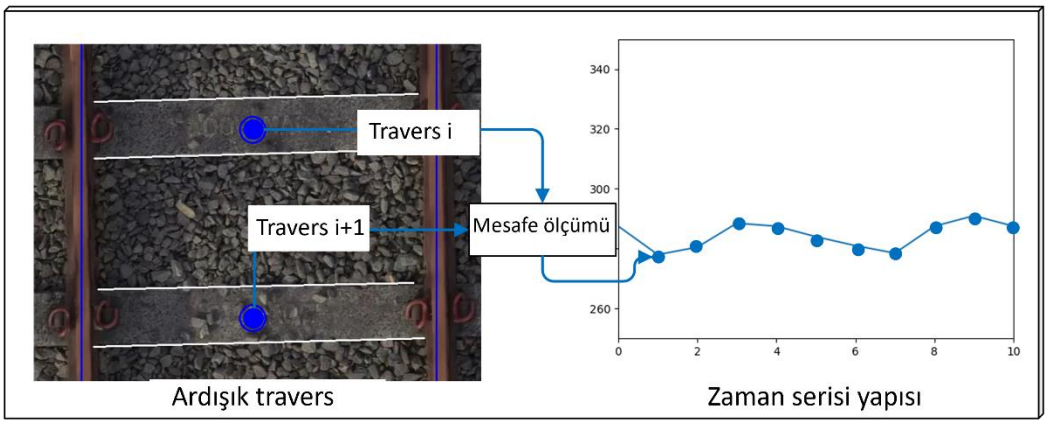

Şekil 3. 33. Hafif hat segmentasyon yöntemi

#### 3.1.6.1. İHA Kontrollü Otonom Ray Takibi

İHA'lar ile otonom ray takibi için PID tabanlı bir sistem önerilmiştir. İHA kamerası, raya 90 derecelik bir açıyla bakacak şekilde yerleştirilmiştir. Rayın merkezi, iki rayın orta noktasıdır. Amaç, ray merkezi ile İHA merkezi arasındaki farkı x ekseni boyunca en aza indirmektir. İHA'nın anlık konumu, alınan görüntünün x ekseni üzerindeki orta noktasıdır. İstenen konum, algılanan iki rayın orta noktası olarak hesaplanır. Bu nedenle amaç, İHA sabit bir yükseklik ve hızda hareket ederken anlık konum ile istenilen konum arasındaki mesafeyi en aza indirmektir. Bu amaçla bir PID kontrolör tasarlanmıştır. Önerilen yöntemde öncelikle bir önceki hata değeri (p_error) sıfır olarak alınmakta ve P ve I değerleri verilmektedir. Alınan her çerçeve için İHA'nın gerçek konumu ve hat segmentasyonu ile bulunan iki rayın orta noktası hesaplanır. Bulunan değer sıfırdan farklı ise yaw_velocity değeri PID ile hesaplanır ve hesaplanan hata bir önceki hata değerine atanır. Bu işlem yeni çerçeveler için tekrarlanır.

#### 3.1.6.2. Travers Sorunlarının Tespiti

Traverslerdeki sorunları tespit etmek için ardışık traversler arasındaki mesafe bir zaman serisi olarak kaydedilir. Daha sonra bu zaman serisindeki değişimler modellenerek traverslerde meydana gelen problemler anomali olarak alınarak modellenmiştir. Şekil 3.34'te travers orta noktalarının tespiti ve ardışık mesafelerin zaman serisine dönüştürülmesi gösterilmiştir.

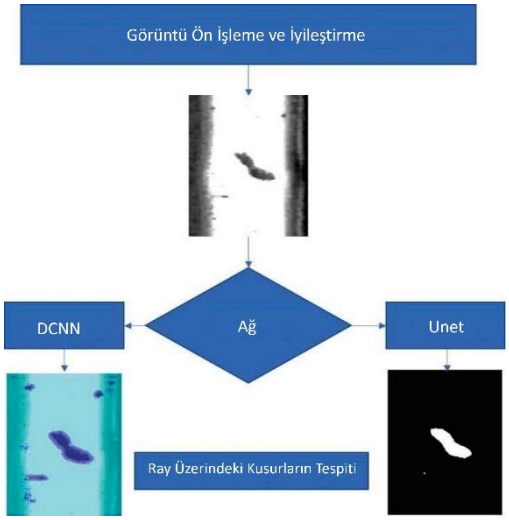


Şekil 3. 34. Sıralı traverslerin konumundan zaman serisi yapısı

Şekil 3.34'te her travers için x ekseninin orta noktası belirlenir. Bu noktalar arasındaki y ekseni boyunca olan mesafe daha sonra zaman serisinde bir nokta olarak kaydedilir. Zaman serisi elde edildikten sonra analiz edilir ve anomali noktaları belirlenir. Traverslerde oluşacak problemler, traversin üzerinin balast ile örtülmesi nedeniyle tespit edilememesi, travers kayması nedeniyle traversin tespit edilememesi veya travers kırılması olarak verilebilir.

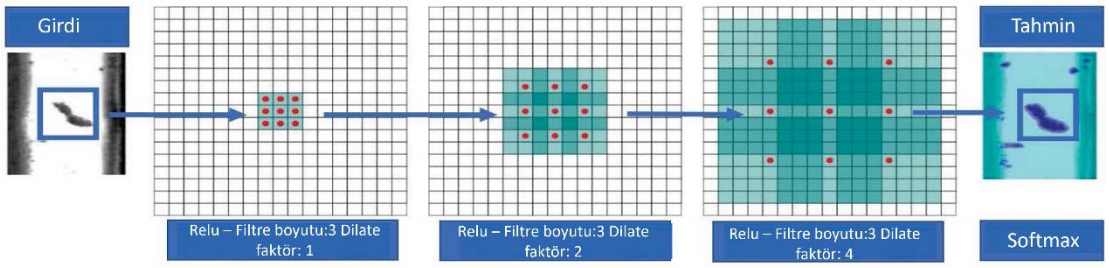
### 3.2. Semantik Segmentasyon Kullanarak Demiryollarındaki Kusurları Tespit Etme

Ray üzerindeki kusurları tespit etmek için görüntü işleme tabanlı bir model önerilmiştir. Yapay zekâ alanının alt dallarından biri olan derin öğrenme yöntemleri ile görüntülerden otonom şekilde kusurları tespit etmek oldukça mümkündür. Derin öğrenme teknikleri, geleneksel tekniklere kıyasla probleme özgü özelliklerin otomatik olarak öğrenilmesine yardımcı olur. Yani derin öğrenme, makine öğrenmesinde tanımlanması gereken parametreleri keşfeder ve makine öğrenmesine göre daha doğru sonuçlar verebilir. Evrişimli sinir ağları, derin öğrenme stratejisine dayanmaktadır. Normalde, evrişimli sinir ağları gibi geniş özellikli öğrenme ağları eğitim için büyük veri kümelerine sahip olması gerekir. Bir diğer önemli nokta ise klasik evrişimli sinir ağlarında genel görüntü eğitiminin sınıf etiketleri üzerinden yapılmasıdır. Ancak bazı problemler piksel tabanlı yaklaşımlar gerektirir. Sağlık veya güvenlik alanları gibi hassas yaklaşım gerektiren alanlarda her pikselin sınıf bilgisine ihtiyaç duyulmaktadır. Görüntü bölütleme burada devreye girmektedir. Bu çalışmada anlamsal bölütleme yöntemleri kullanılmıştır. Anlamsal bölütleme, bir görüntüdeki her pikselin hangi sınıfa ait olduğunu belirler [56]. Bölütleme görevi Unet ve DCNN ağları ile sağlanmıştır. Veri setindeki görüntüler, bölütleme işleminden önce görüntü ön işleme adımlarından geçmiştir. Bu aşamada veri setindeki ray görüntüleri kırpılmış ve boyutlandırılmıştır. Önerilen model Şekil 3.35'te verilmiştir.




Şekil 3. 35. Önerilen model

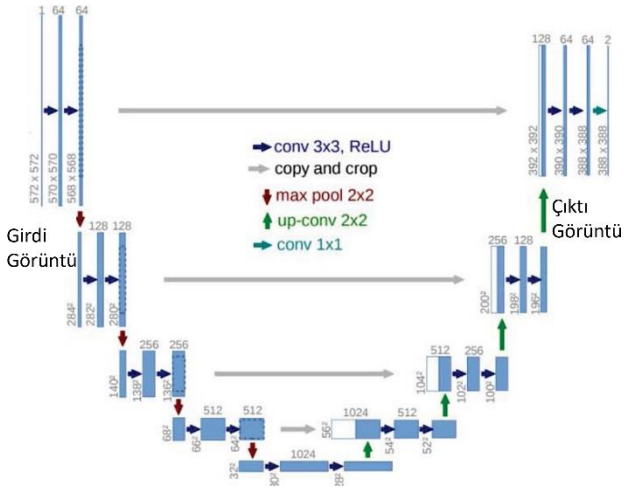
DCNN, her adımda parametre sayısı arttırılmadan alıcı alanın arttırılması yoluyla büyük vektör temsilini taramaktadır. Genişleme oranı 1 olan sıradan evrişimle kıyaslandığında parametre sayısı açışından yaklaşık olarak 3 kat azaldığı görülmektedir. Şekil 3.36.'da DCNN mimarisi modelinin her genişleyen evrişim filtresinin uygulanmasından sonraki dönüşümü gösterilmektedir.




Şekil 3. 36. DCNN mimarisi

Görüntüler arka plan ve kusur etiketlerinden oluşmaktadır. Piksel etiketlerinin çoğu arka plan olarak etiketlenmiştir. Bu durum sınıflar arasında dengesizliğe yol açabilir. Bunu düzeltmek ve sınıfları dengelemek için ters frekans sınıf ağırlıklandırması kullanılmıştır. İlk olarak, giriş görüntülerinin boyutuna karşılık gelen giriş katmanı kullanılarak piksel sınıflandırması için bir ağ oluşturulmuştur. Şekil 3.36'da görüldüğü gibi ikinci katmanda genişleme faktörü 1 olan 3x3 boyutunda 32 filtre içeren Relu katmanı kullanılmaktadır. Genişleme faktörü 2 ve 4 olan 2 Relu katmanı daha kullanılmıştır. Bu faktörler, 3x3 boyutlu filtreyi, çözünürlük veya kapsama kaybı olmadan sırasıyla 7x7 ve 15x15 boyutunda filtreler yapmıştır [57]. Ardından softmax katmanı ve ters sınıf ağırlıklarına sahip pixelClassificationLayer katmanı kullanılmış ve eğitim gerçekleştirilmiştir. Ardından, eğitilmiş ağın orijinal ve temel gerçek görüntüleri anlamsal bölütleme kullanılarak test edilmiştir. Ronneberger ve ark. Unet Evrişimsel Sinir Ağı

yaklaşımını, ilk kez 2015 yılında duyurmuşlardır [58]. Unet, otomatik bir bölütleme yöntemidir ve sınırı tanımlama ile ilgilidir. Unet mimarisi klasik modellere göre daha başarılı sonuçlar vermektedir. Ayrıca, az sayıda eğitim görüntüsü olsa bile iyi sonuçlar verebilmektedir. Unet, adını Şekil 3.37'de görüldüğü gibi U harfine benzer mimarisinden almaktadır.




Şekil 3. 37. Unet mimarisi

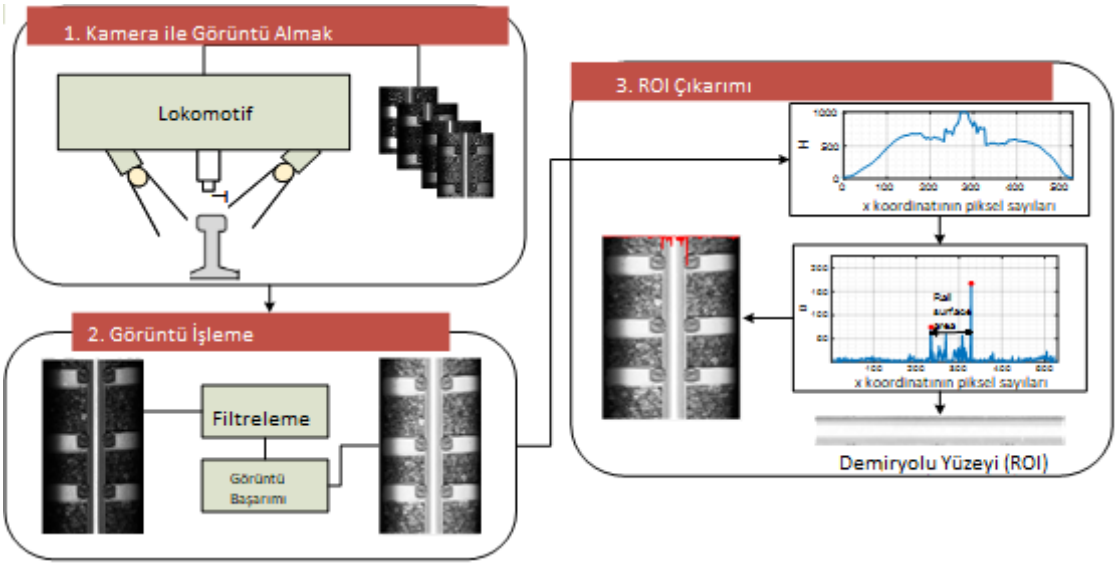
## 3.3. Sürdürülebilir Taşımacılıkta Demiryolu Hatları için Derin Özelliklere Dayalı Kusur Sınıflandırması

Bu çalışmada, iki derin öğrenme modelinin özelliklerinin birleştirilmesiyle ray yüzey kusurları tespit edilmiştir. Bu amaçla seçilen iki model olan SqueezeNet ve MobileNetV2, diğer derin öğrenme modellerine göre hem daha küçük hem de daha hızlıdır. Ancak, bu modellerin her ikisi de diğer modellerden daha az hassastır. Bu nedenle bu çalışmada, iki modelin özellikleri birleştirilerek yüksek doğrulukta bir füzyon modeli önerilmiştir. Önce rayın orijinal görüntüsüne kontrast ayarı yapılır ve ardından ray hattı konumu belirlenir. Daha sonra her ağdan en ağırlıklı özellikler seçilir ve azaltılmış özellikler Destek Vektör Makinelerine (SVM) verilerek kusurlar belirlenir. Deneysel sonuçlar, önerilen yöntemin, tek bir derin öğrenme modeli kullanmaktan ziyade, düşük kontrast altında çoklu ray yüzey kusurları için daha iyi sonuçlar verdiğini göstermektedir.

### 3.3.1. Görüntü Ön İşleme ve Ray Konumlandırma

Görüntü toplama sisteminde kamera tarafından toplanan ray bölgesine ait görüntülerde beklenen balastlar, bağlantı elemanları, traversler ve diğer parçalar bulunmaktadır. Ray yüzey kusurlarını tespit etmek için rayın görüntüdeki konumunun bulunması ve kırpılması gerekir. Ancak trenin altında çekilen görüntülerde genellikle gürültü ve ışıklandırma sorunları

yaşanmaktadır. Bu nedenle elde edilen görüntülere öncelikle aydınlatma ile ilgili problemlerin giderilmesi için ön işlem yapılması gerekmektedir. Daha sonra ray konumlandırma algoritması ile ray yüzeyi belirlenecektir. Önerilen ön işleme ve ray konumlandırma yönteminin blok şeması Şekil 3.38'de verilmiştir.




Şekil 3. 38. Görüntü ön işleme ve ROI çıkarma

Şekil 3.38'te, dikey olarak yerleştirilmiş iki kamera, ilk aşamada ray yüzeylerinin görüntülerini yakalar. Bu kameralar bir lokomotifin altına monte edilir ve her kamera rayın bir tarafını izler. Değişken aydınlatma koşullarının etkisini azaltmak için her ray için iki ışık kaynağı kurulur. İkinci aşamada, ray tespit algoritmalarının doğru çalışabilmesi için elde edilen görüntülere filtre ve kontrast iyileştirme gibi görüntü işleme yöntemleri uygulanmaktadır. Üçüncü adımda elde edilen görüntüden ROI içeren ray çıkarılır. ROI görüntüleri bir veri tabanına kaydedilir ve hata tanıma modülü için kullanılır. Görüntü ön işleme ve ROI çıkarmanın detayları aşağıdaki alt bölümlerde verilmiştir.

### 3.3.2 Görüntü İyileştirme ve Ön İşleme

Renkli bir görüntü, R, G ve B adı verilen üç matriste tutulur. Bu matrisler, bir görüntünün Kırmızı (R), Yeşil (G) ve Mavi (B) bileşenlerinin yoğunluk değerlerini tutar. Herhangi bir görüntü I'deki bir (x, y) pikselinin yoğunluk değeri, sırasıyla R (x, y), G (x, y) ve B (x, y) ile ifade edilir. Başka bir deyişle, bir piksel üç yoğunluk değerinden oluşur. Bir görüntünün genişliği ve yüksekliği, sırasıyla x=1,…,M ve y=1,…,N için M ve N ile gösterilir. Bir RGB görüntüsünü gri tonlama formatına dönüştürmek için birçok yöntem olmasına rağmen, genel olarak kullanılan denklem (3.15) [59]'de verilmiştir

$$I(x,y) = c_r R(x,y) + c_g G(x,y) + c_b B(x,y) \tag{3.15}$$

Denklem (3.7)'de $c_r$, $c_g$ ve $c_b$ değerleri, RGB renkli görüntünün her bir bileşeninin katsayı değerleridir. I (x, y), (x, y) noktasındaki gri görüntünün yoğunluk değerini temsil eder. Renkli bir RGB görüntüsü gri formata dönüştürüldüğünde $c_r$, $c_g$ ve $c_b$ katsayı değerleri sırasıyla 0.2989, 0.5870 ve 0.1140 olarak alınır [60]. Bu dönüşümden sonra parlaklık korunurken ton ve doygunluk bilgisi kaybolur [61]. İlk olarak görüntü normalizasyonu ile gri formatta 0 ile 255 arasında piksel değerine sahip görüntü (3.16)'da gösterildiği gibi 0 ile 1 arasında çift formata dönüştürülür.

$$J(x,y) = \frac{I(x,y) - I_{\min}}{I_{\max} - I_{\min}} \tag{3.16}$$

Denklem (3.16)'da, $I_{max}$ ve $I_{min}$ parametreleri sırasıyla görüntü I'deki minimum ve maksimum yoğunluk değerini gösterir. Normalleştirme, J görüntüsünün değerlerini 0 ile 1 arasında küçültür. Görüntünün uygun bir temsili, yüksek kontrast ve düşük homojenlik ile ölçülür. Bu nedenle, yüksek kontrastlı ve düşük homojenliğe sahip bir görüntünün varyansı yüksek olmalıdır. Bu amaçla J görüntüsünün varyansını maksimize edecek en uygun katsayılar belirlenmelidir. $c_b$ değeri genelleme kaybı olmadan 1 olarak alınabilir. Dönüşüm için J görüntüsünün varyansını maksimize etmek için $c_r$ ve $c_g$ değerleri belirlenmeli ve bu amaçla gradyan iniş yöntemi kullanılabilir. Bu yöntem, en uygun çözüm bulunana kadar, bir başlangıç değerinden başlayarak, parametrelere küçük bir değerin yinelemeli olarak eklenmesine dayanır. Gradyan inişi denklem (3.17)'da verilmiştir.

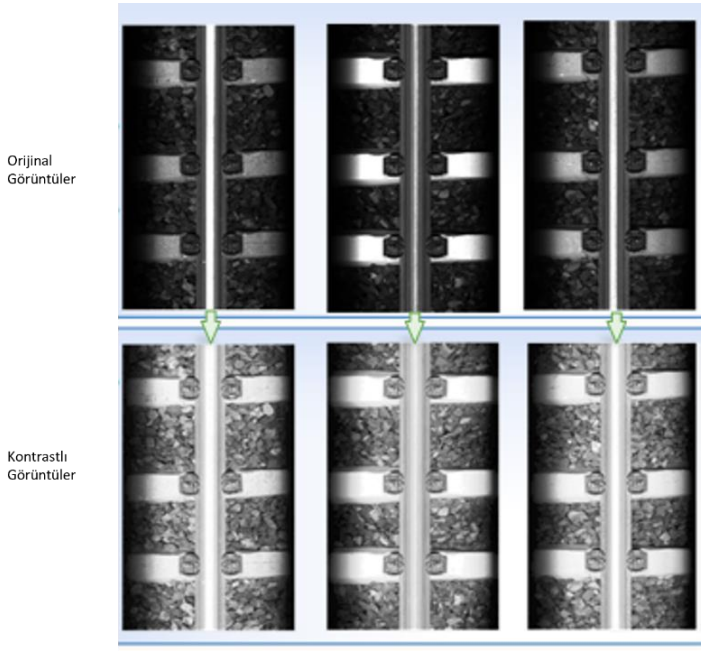
$$(c_r^{i+1}, c_g^{i+1}) = (c_r^i, c_g^i) + (\Delta_r^i, \Delta_g^i) \tag{3.17}$$

Denklem (3.17)'de $\Delta_r^i$ ve $\Delta_g^i$, sırasıyla varyansın gradyanı kullanılarak hesaplanır. Belirli bir optimum sonuç elde edildiğinde yineleme durdurulur. Bu amaçla MATLAB'da kullanılan *fminsearch* fonksiyonu, maliyet fonksiyonunu maksimize etmek için varyansın negatifi alınarak kullanılabilir [62]. Bu fonksiyon, Nelder-Mead yöntemini kullanan çok boyutlu kısıtsız bir optimizasyon tekniğine dayanmaktadır ve çok hızlıdır. Kontrast geliştirme yönteminin genel yapısı Şekil 3.39'da verilmiştir. Eğer tespit edilen nesnelerin pozisyonları temas telinin pozisyonu ile aynı ise ark olarak alınır.

```
Algorithm: J= function contrast_enhancement(I)
    Input: I→An RGB Image
    Output: J→High contrast gray image
1   (c_r, c_g)=arg  max c_r, c_g (σ_J^2)
2   R=I(:, :,1) %Red component of I
3   G=I(:,:,2) % Green component of I
4   B=I(:,:,3) % Blue component of I
5   if size(I, 3)==1 then
6       In=I;
7   else
8       In=imresize(In,[64 64]);
9       c=fminsearch('StdMonochrome', [1 1], [ ],In);
10      c_r=c(1)
11      c_g=c(2)
12      I=c_r*R+c_g*G+B
13  endif
14  J=(I-min(I(:)))/max(J(:));
15  endfunction
```

Şekil 3. 39. Kontrast geliştirme algoritması

Şekil 3.39'da yüksek kontrastlı bir görüntü elde edilmekte ve bu görüntüde ray konumu orijinal görüntüye göre daha kolay belirlenebilmektedir. Önerilen algoritma ile parametre olarak alınan RGB renkli görüntü gri görüntüye dönüştürülerek kırmızı ve yeşil kanalların optimum katsayıları hesaplanarak yüksek kontrastlı görüntü elde edilmiştir. Şekil 3.40, farklı ray görüntüleri için kontrast geliştirme algoritmasının sonuçlarını göstermektedir.




Şekil 3. 40. Üç ray görüntüsü için kontrast geliştirme

Şekil 3.40'ta görüldüğü gibi orijinal ray görüntüleri ölçüm treninin altına bir ışık kaynağı yerleştirilerek alınmasına rağmen ray yüzeyinde hala çok parlak bölgeler oluşmakta ve ışığın homojen bir dağılımı bulunmamaktadır. Bu şekilde rayı tespit etmek oldukça zordur. Ancak orijinal görüntüye kontrast iyileştirmesi uygulandığında ray yüzeyinin net bir şekilde ayırt edilebildiği görülmektedir.

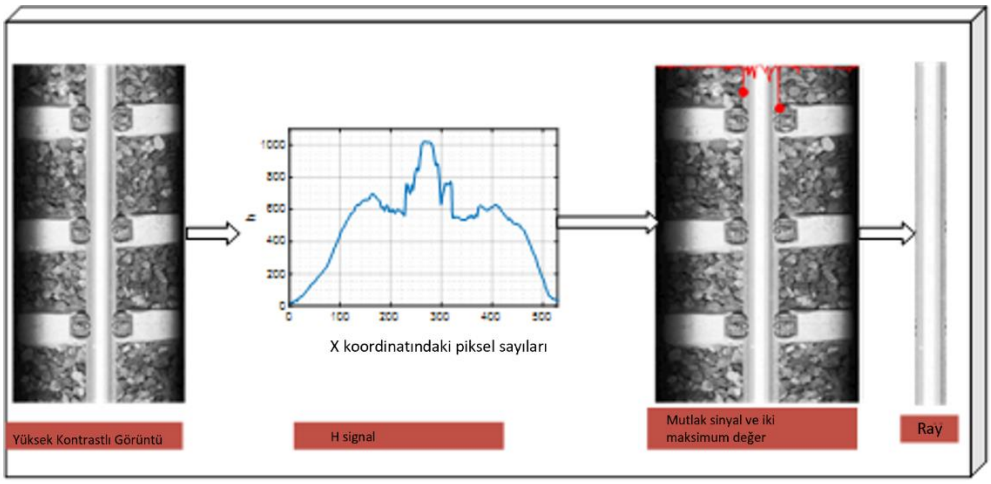
### 3.3.3 Ray Konumlandırma

Görüntü kontrastı iyileştirildikten sonra, ray yüzeyi kusurlarını belirlemek için ROI çıkarılmalıdır. Uygulamada, ölçüm treni ile ray görüntüleri alınırken bir ışık kaynağı kullanılmasına rağmen, ışık kaynağının yerleşiminden dolayı rayın bazı kısımları daha parlak, bazı kısımları ise daha koyu kalmaktadır. Bu nedenle görüntü kontrast ayarı ile bu sorun çözülmektedir. Bu adımdan sonra, görüntüden ray yüzeyinin kırpılması ve ilgili bölgenin çıkarılması için bir yöntem önerilmiştir. Önerilen yöntem, gri ölçekli bir ray görüntüsü üzerinde her satırda sütun toplamları yaparak bir vektör elde etmekte ve daha sonra bu vektörün elemanlarının mutlak farkını alarak sinyaldeki en büyük iki değeri bulmaktadır. Elde edilen değerler rayın sınırlarını temsil etmektedir. ROI ekstraksiyonunun sözde kodu Şekil 3.41'de verilmiştir.

```
Algorithm [L,R]=ROI_extraction(I)
        L, R→left and right boundaries of rail
        I→enhanced gray image
1       rows, cols←size(I,1),size(I,2)
2       h←array[1..cols]=0
3       for j=1 to cols do
4               for i=1 to rows do
5                       h(j)←h(j)+I(i, j)
6               endfor
7       endfor
8       dvp=array[1..cols-1]=0
9       for i=2 to  cols-1 do
10              dvp(i)=|h(i)-h(i-1)|
11      endfor
12      [val, index]←sort(dvp,'descending')
13      if index(2)<index(1) then
14              L←I(index(2))
15              R←I(index(1))
16      else
17              L←I(index(2))
18              R←I(index(1))
19      endif
20    endfunction
```

Şekil 3. 41. Roi çıkarma algoritması

Şekil 3.41’de, her satırdaki yüksek çözünürlüklü görüntünün yoğunluğu bir vektörde toplanmıştır. Daha sonra, kümülatif bir vektörün ardışık elemanları arasındaki fark alınarak bir fark vektörü elde edilir. Farkın maksimum olduğu iki değer, ray sınırlarını gösterir. Şekil 3.42, bir ray görüntüsü üzerinde ROI çıkarma yönteminin uygulama sonucunu göstermektedir.



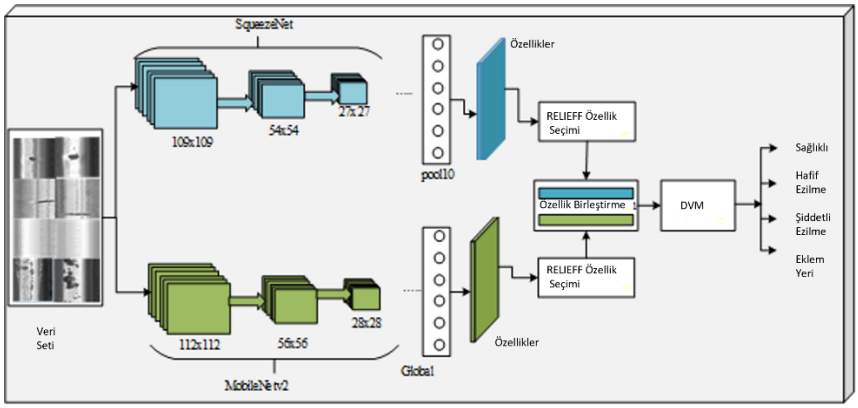

Şekil 3. 42. Raydan ROI çıkarımı

Şekil 3.42'de gösterildiği gibi, ray hattının sınırını tespit etmek için bir sinyal elde edilmektedir. Bu sinyalin ardışık elemanlarının farkı alınarak en yüksek iki nokta rayın başı ve sonu olarak alınır. Son olarak elde edilen ROI görüntü tanıma modülüne verilir ve kusurlar sınıflandırılır.

### 3.3.4 Derin Unsurlar Kullanılarak Ray Yüzey Hatalarının Sınıflandırılması

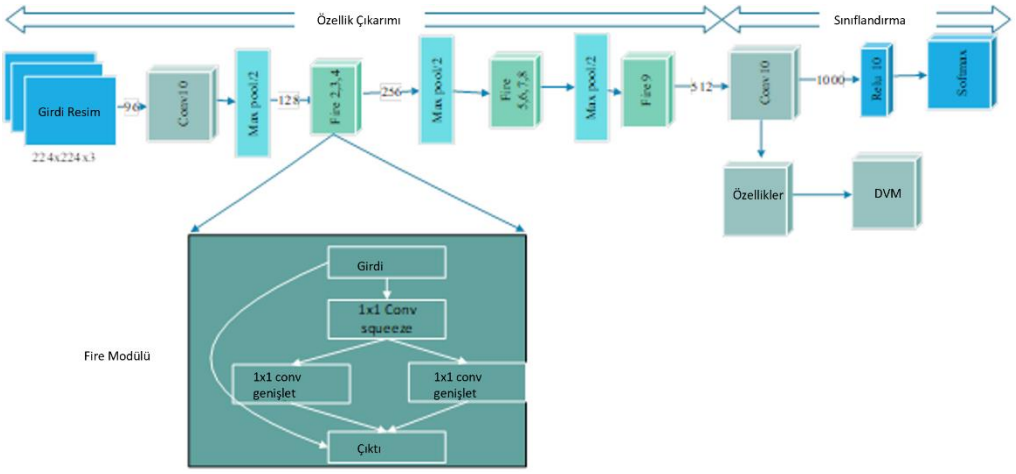
Bir önceki bölümde, görüntü iyileştirme ve ray konumlandırma problemi çözülmüş ve ortaya çıkan görüntü veri seti, diğer ray bileşenleri olmadan sadece ray yüzeylerinden oluşmaktadır. Bu bölümde derin öğrenme tabanlı öznitelik çıkarımı ve hata türlerinin sınıflandırılması sunulacaktır. Hata sınıflandırma modülü, yüksek doğruluk oranı elde etmek için iki derin öğrenme modülünün birleştirilmesiyle oluşturulur. Bu amaçla MobileNetV2 [63] ve SquezeNet [64] modülleri kullanılmıştır. Normalde derin öğrenme modellerini transfer öğrenme ile eğitmek oldukça uzun zaman alır ve test aşamasında çok sayıda hesaplama gerektirir. Önerilen yöntemle öznitelik çıkarımı yapıldıktan sonra ReliefF [65] ile öznitelik seçimi ile daha az öznitelik kullanımı sağlanacaktır. Bu iki derin öğrenme modelinin seçilmesinin nedeni, diğer derin öğrenme modellerine göre daha az parametre gerektirmeleri ve daha hızlı olmalarıdır. MobileNet, sınıflandırma, segmentasyon ve nesne tespiti için kullanılan düşük maliyetli donanım gerektiren bir CNN'dir. MobileNetV1'de önerilen derinlik bilgisine sahip evrişim katmanı hem model boyutunu hem de karmaşıklığı azaltmıştır. MobileNetV2'de ise, önerilen ters çevrilmiş artık bloklar, katmanlar arasındaki doğrusal olmayan ilişkiyi ortadan kaldırmıştır.

MobileNetV2 modeli, MobileNetV1'den geliştirilmiştir ve önceki modele kıyasla katmanlar arasında doğrusallaştırmayı sağlayan modüllere sahiptir. SqueezeNet, evrişim, havuzlama, ReLU ve Fire katmanlarından oluşan bir CNN'dir. SqueezeNet'in tam bağlantılı bir katmanı yoktur ve bu görev Fire katmanları ile yapılır. SqueezeNet, noktasal filtreler kullanarak hesaplama süresini azaltır ve giriş kanallarının sayısını 3x3 filtrelere indirerek ağ doğruluğunu korumaya çalışır. Ağda geç örnekleme yaparak evrişim katmanlarının büyük aktivasyon haritalarına sahip olmasını sağlar, bu da daha yüksek sınıflandırma doğruluğu ile sonuçlanır. İki ağ modelinin birleştirilmesi önerilmiştir, çünkü bu iki ağın her ikisi de daha az parametreye sahiptir ve mobil veya düşük maliyetli donanımda uygulanabilmektedir. Böylece hataların gerçek zamanlı olarak tespit edilebildiği bir ağ modeli oluşturulabilir. Önerilen hata sınıflandırma yönteminin blok şeması Şekil 3.43'te verilmiştir.




Şekil 3. 43. Önerilen kusur sınıflandırma yöntemi

Şekil 3.43'te her bir derin sinir ağına eğitim seti verilerek öznitelikler elde edilmiştir. Daha sonra elde edilen özniteliklerin ağırlıklarını belirlemek için ReliefF algoritması uygulanmıştır. Yüksek ağırlıklı özelliklerin bir bölümü seçilmiştir. Bu özellikler daha sonra birleştirilir ve kusurları sınıflandırmak için destek vektör makinesine verilir. Son yıllarda derin öğrenme yöntemlerinin ana odak noktası doğruluk olmuştur. Birden fazla CNN ile belirli bir doğruluk oranına ulaşmak mümkündür. Bununla birlikte, eşdeğer doğrulukla, daha küçük ağ mimarileri, gömülü olarak uygulanacak daha az bant genişliği, dağıtılmış bilgi işlem sırasında daha az iletişim gerektirir ve bir modeli buluttan otonom araca aktarabilir. SqueezeNet, Alexnet'e yakın bir doğruluk performansı sağlayan ve 50 kat daha az parametreye sahip bir CNN mimarisi olarak geliştirilmiştir. Ayrıca SqueezeNet, 3x3 filtre kullanımına kıyasla dokuz kat daha az parametre elde ederek parametre sayısını azaltmak için 3x3 filtreler yerine 1x1 filtreler kullanır. SqueezeNet derin öğrenme mimarisinin yapısı Şekil 3.44'te verilmiştir.




Şekil 3. 44. Squeezenet mimarisi

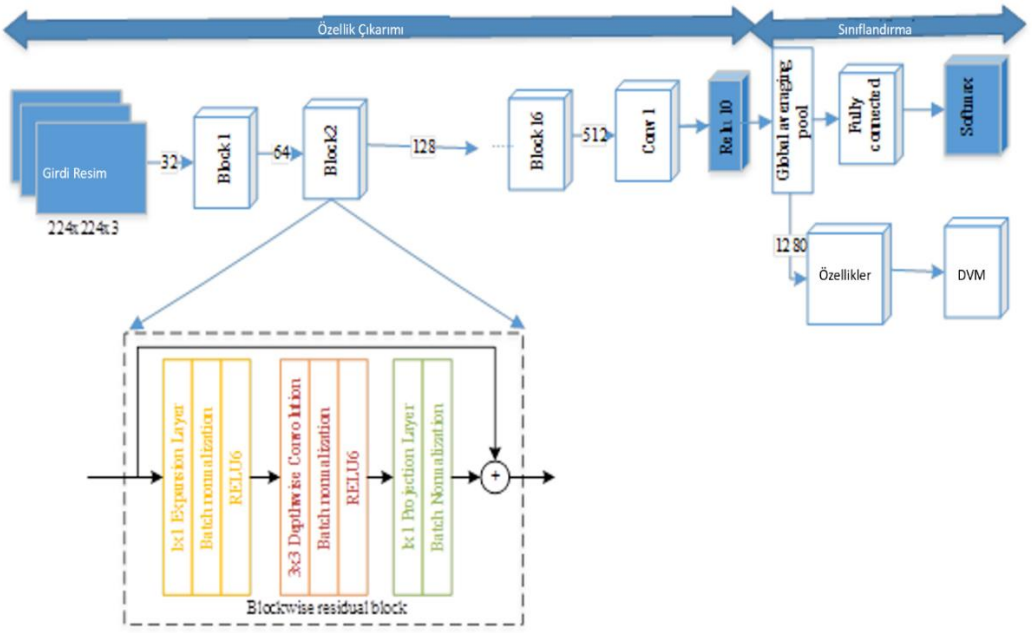
Şekil 3.44'te verilen SqueezeNet mimarisi 16 katmandan oluşmaktadır. İlk katmanda, girdi görüntüsü bir evrişim katmanına verilir ve havuz katmanından geçen görüntü sekiz Fire katmanına verilir. Ağ mimarisindeki son evrişim katmanından sonra ağ, ReLU ve Softmax katmanı ile sonlanır. Bu çalışmada SqueezeNet'ten öznitelik çıkarımı yapılmış ve bu öznitelikler pool10 adı verilen eğitilebilir son katmandan çıkarılmıştır. Bu katmanda 1.000 öznitelik elde edilmiştir. SqueezeNet parametreleri Tablo 3.4'te verilmiştir.

Tablo 3. 4. Squeezenet parametreleri ve mimari boyutlar

| **Katman** | **Çıktı Boyutu** | **Filtre Boyutu** | | |
|---|---|---|---|---|
| | | **1x1 squeze** | **1x1 expand** | **3x3 expand** |
| Input image | 224x224x3 | - | - | - |
| Conv1 | 111x111x96 | - | - | - |
| Maxpool1 | 55x55x96 | - | - | - |
| Fire2, Fire3 | 55x55x128 | 16 | 64 | 64 |
| Fire4 | 55x55x256 | 32 | 128 | 128 |
| Maxpool4 | 27x27x256 | - | - | - |
| Fire5 | 27x27x256 | 32 | 128 | 128 |
| Fire6, Fire7 | 27x27x384 | 48 | 192 | 192 |
| Fire8 | 27x27x512 | 64 | 256 | 256 |

| | | | | |
|---|---|---|---|---|
| Maxpool8 | 13x12x512 | - | - | - |
| Fire9 | 13x13x512 | 64 | 256 | 256 |
| Conv10 | 13x13x1000 | - | - | - |
| Avgpool10 | 1x1x1000 | - | - | - |

Özellik çıkarımı için kullanılacak diğer model MobileNetV2 olarak seçilmiştir. Bu model, düşük bellek kullanımı nedeniyle özellikle gömülü sistemlerde kullanılmaktadır. MobileNetV2 bir CNN modeli olup toplam 53 katmandan oluşmaktadır. Model mimarisi başlangıçta 32 filtreli tamamen evrişimli bir katmanla başlar, ardından 19 artık darboğaz katmanı gelir. Burada, ters çevrilmiş artık bloklar, noktasal ve derinlemesine evrişim katmanlarından oluşur. Noktasal evrişim katmanı, standart evrişimin özel bir şeklidir ve çekirdek boyutu, farklı giriş kanallarını doğrusal olarak birleştirmek için tek boyutludur. MobileNetV2'de kullanılan ters çevrilmiş artık katman, üç ayrı evrişimden oluşur. Birincisi, düşük boyutlu girdi özelliğini yüksek boyutlu bir alana genişletmek için 1x1 noktasal evrişim kullanmaktır. Bu işlemden sonra ReLU6 uygulanır. Ardından, 3x3 çekirdekler kullanılarak derinlemesine evrişim uygulanır ve yeniden ReLU6 uygulanır. Son adımda, uzamsal filtrelenmiş özellik haritası, başka bir noktasal evrişim kullanılarak düşük boyutlu bir alt uzaya dönüştürülür. MobileNetV2'nin yapısı Şekil 3.45'te verilmiştir.




Şekil 3. 45. MobileNetV2 mimarisi

Şekil 3.45'te gösterildiği gibi, girdi görüntüsü, birinci evrişim bloğundan sonra 16 blok halinde artık bloktan geçer. Daha sonra giriş bloğu, ReLU ve ortalama havuzlama bloğundan geçer ve ardından tam bağlantılı katmana ulaşır. Ortalama havuz bloğuna kadar olan kısım özellik çıkarımı için kullanılırken, bu aşamadan sonraki kısım sınıflandırma için kullanılır. Bu çalışmada, global ortalama katmanından 1.280 öznitelik elde edilmiş ve DVM için uygun bir forma dönüştürülmüştür. Tablo 3.5, katmanlardaki filtrelerin ve diğer özelliklerin sayısını göstermektedir.

Tablo 3. 5. MobileNetV2 parametreleri

| **Katman** | **Çıktı Boyutu** | **Filtre Boyutu** | **Sayı** |
|---|---|---|---|
| Conv2d | 224x224x3 | 32 | 1 |
| Bottleneck | 112x112x32 | 16 | 1 |
| Bottleneck | 112x112x16 | 16 | 2 |
| Bottleneck | 54x56x24 | 24 | 3 |
| Bottleneck | 28x28x32 | 32 | 4 |
| Bottleneck | 14x14x64 | 64 | 3 |
| Bottleneck | 14x14x96 | 160 | 3 |
| Bottleneck | 7x7x160 | 320 | 1 |
| Conv2d 1x1 | 7x7x320 | 1280 | 1 |
| Avgpool | 7x7x1280 | - | 1 |
| Conv2d 1x1 | 1x1x1280 | - | - |

Tablo 3.5'te bazı bloklar birden fazla uygulanmıştır. Bu durum tablonun son sütununda verilmiştir. Her iki CNN'den de öznitelikler elde edildikten sonra bu öznitelikler üzerinde öznitelik seçimi yapılır ve yeni öznitelikler bir araya getirilerek sınıflandırıcının girdileri elde edilir. mx1000 ve mx1280 özellik matrisleri, sırasıyla SqueezeNet'in pool10 katmanından ve MobileNetV2'nin ortalama havuz katmanından elde edilir. Matris boyutundaki m parametresi veri setindeki örnek sayısını gösterirken diğer parametre her örnek için elde edilen öznitelik sayısını gösterir. Öznitelik seçiminden sonra CNN'lerden n1 ve n2 özniteliklerinin elde edildiğini varsayarsak, birleştirme sonunda sınıflandırıcıya verilecek veri setinin boyutu mx (n1+n2) olur. Özellik seçimi için ReliefF yöntemi kullanılır. Bu yöntem, mesafeye dayalı seçim seçilerek

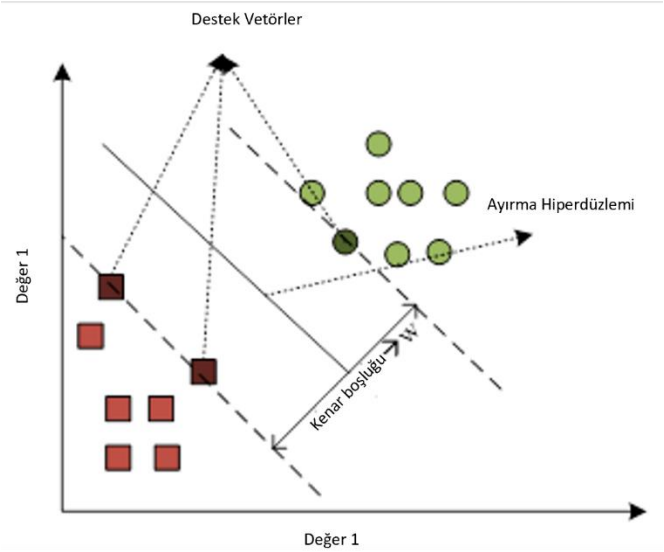
özellikler için negatif ve pozitif ağırlıklar üretir ve ardından negatif ağırlıklar elimine edilir. Bu yöntem, Rölyef yönteminin geliştirilmiş bir versiyonudur ve ağırlıkları hesaplamak için Öklid mesafesi yerine Manhattan mesafesini kullanır. Bu algoritma, çok sınıflı kategorik değişkenlerin olduğu bir durumda en yakın komşuluk değerine göre tahmin edicilerin ağırlıklarını bulur. Tahmin edicilerin sıralaması ve ağırlığı genellikle en yakın komşu sayısı k'ye bağlıdır. Başlangıçta tüm tahmin ediciler için ağırlık değerleri 0 olarak seçilir. Algoritma daha sonra rastgele bir örnek seçer ve k-en yakın komşuyu bularak ağırlıkları günceller. Her özelliğin ağırlık güncellemesi denklem (3.18)'de hesaplanır.

$$W[A] = W[A] - \sum_{j=1}^{k} \frac{d_A(R, H_j)}{m} + \sum_{c \neq class(R)} \left[ \frac{P_Q}{1 - P_R)} \right] \sum_{j=1}^{k} \frac{d_A(R, M_j)}{m} \tag{3.18}$$

Denklem (3.18)'de $W[A]$, bir A niteliğinin ağırlığını temsil eder. R, rastgele seçilmiş bir örneklemdir, Hj, aynı A niteliği sınıfına sahip en yakın örnektir ve Mj, farklı bir sınıftaki A niteliğine sahip en yakın örnektir. (3.18)'deki m parametresi yineleme sayısını, k ise en yakın komşuların sayısını belirtir. $P_Q$ ve $P_R$ değerleri sırasıyla mevcut örneklemin ve rastgele seçilen örneklemin sınıfa ait olma olasılığını gösterir. $d_A$ İşlevi, parametre olarak alınan iki örnek arasındaki mesafeyi gösterir ve (3.19)'daki gibi hesaplanır.

$$d_A(X, Y) = \frac{|X - Y|}{A_{\max} - A_{\min}} \tag{3.19}$$

Denklem (3.19)'da $A_{max}$ ve $A_{min}$, sırasıyla A niteliği için maksimum ve minimum değerleri temsil eder. ReliefF algoritması ağırlıkları yinelemeli olarak günceller ve öznitelikleri sınıflandırmada en ağırlıklıdan en azına doğru sıralar. Öznitelik seçimi sonrasında elde edilen indirgenmiş öznitelikler DVM'lere verilir. DVM, sınıflandırma amacıyla hiper düzlemler oluşturan sınıflandırma ve regresyon için kullanılan bir makine öğrenme yöntemidir. İyi bir sınıflandırma için, farklı sınıfların en yakın örnekleri arasındaki marj maksimum olmalıdır. Normalde, DVM'ler iki sınıflı bir sınıflandırma yöntemi olmasına rağmen çok sınıflı sınıflandırma için kullanılır. Çok sınıflı problemler için biri diğerine karşı ya da tüm yöntemlere karşı kullanılır [66]. Şekil 3.46, doğrusal ayrılabilir numuneler için DVM'lerde ayırma hiper düzlemini göstermektedir.




Şekil 3. 46. Destek Vektör Makinesinin karar hiper düzlemi

İki sınıfın destek vektörleri, Şekil 3.46'da gösterildiği gibi doğrusal verileri ayırmak için kullanılır. Destek vektörleri, ayırma hiper düzlemine en yakın örneklerdir. Amaç, iki sınıfın en yakın örnekleri arasındaki marjı maksimize etmektir [67]. DVM'lerin en iyi ve en uygun sınıflandırma parametrelerinin seçimi aslında bir optimizasyon problemi olarak görülmektedir. Eğitim örnekleri $x_i \in \Re^p$, i=1,…n, sınıf etiketleri y $\in \{-1 \quad 1\}$ olarak verildiğinde amaç w ve b değerlerini bulmaktır ve çoğu örnek için $sign(w^T x+b)$ değerinin doğru olmasını bekleriz. Bunun için aşağıdaki temel problem çözülmelidir.

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C\sum_{i=1}^{n} \zeta_i$$
$$subject\ to\ \ y_i(w^T x_i + b) \geq 1-\zeta_i, i = 1,...,n \qquad (3.20)$$

Denklem (3.20)'de w ve b sırasıyla n-boyutlu ağırlık vektörünü ve yanlılık değerini temsil etmektedir. $y_i$ parametresi, sınıf etiketi olarak -1 veya 1 değerini alır. Marj, $w^T w$ değeri minimize edilerek maksimize edilir. İdeal olarak, $y_i(w^T x_i + b)$'nin değeri her zaman >=1 olmalıdır. Bununla birlikte, kusursuz sınıflandırma her zaman yapılamayacağından, bazı numunelerin doğru marjlarının sınırında olmasına izin verilebilir. Bu değişkenler gevşek değişkenler ($\zeta_i$) olarak ifade edilir. Marj genişliğini belirleyen faktör ceza parametresi C'dir. Bu parametre kullanıcı tarafından uygun şekilde seçildiğinde sınıflandırma performansı artar [68]. Bu bir optimizasyon problemidir ve denklemi minimize edecek değerler Lagrange çarpanları ile çözülebilir. Lagrange çarpanları ikinci dereceden bir programlama problemi olarak ifade edilebilir ve aşağıdaki gibi verilebilir.

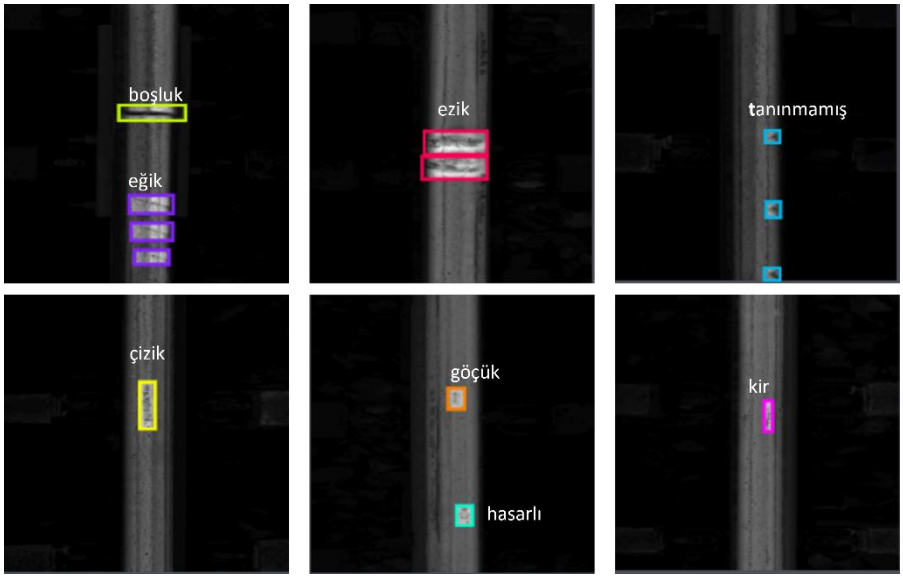
$$\left\{\begin{array}{ll} Maximize & L(\alpha)=\sum_{i=1}^{k}\alpha_i-\frac{1}{2}\sum_{i=0}^{k}\sum_{j=0}^{k}\alpha_i\alpha_j y_i y_j x_i x_j \\ Subject\ to & \sum_{i=1}^{k}\alpha_i y_i=0,\ for\ i=1....k \end{array}\right\} \quad (3.21)$$

Denklem (3.21)'de $\alpha_i$ , 0 ile C arasında pozitif değerler alan Lagrange çarpanlarıdır. $L(\alpha)$ değeri, pozitif $\alpha_i$ değerleri için maksimize edilmesi gereken Lagrange işlevidir. Veri setinde destek vektörü olan veriler için $\alpha_i$ değerleri pozitif değer alırken diğerleri 0 değerini alır. Bu optimizasyon problemini çözmek için sıralı minimizasyon algoritması kullanılır [69]. Problem lineer uzayda çözülemezse, o zaman problem uzayı kernel fonksiyonları kullanılarak lineer olmayan bir uzaya dönüştürülür. Popüler çekirdek işlevleri, polinom, radyal tabanlı ve tanh tabanlı çekirdek işlevleridir. Bu çalışmada, çok sınıflı sınıflandırma problemi için DVM'ler kullanılacaktır. Sorun doğrusal olmayan sınıflandırma olduğundan, çekirdek işlevi olarak radyal tabanlı işlev kullanılacaktır.

### 3.4. YOLOv5 ve Topluluk Öğrenme ile Ray Kusurlarının Tespiti

Önerilen yöntemde, ray üzerindeki kusurların tespiti için YOLOv4, YOLOv5 ve YOLOv6 modelleri uygulanmıştır. Ray üzerindeki kusurlar farklı kusur çeşitlerini içermektedir. Aynı zamanda tek görüntüde birden fazla kusur bulunabilmektedir. YOLO tabanlı modellerin tercih edilmesinin sebebi hızlı tahminler yaparken aynı zamanda da çok iyi sonuçlar verebilmeleridir. Örneğin R-CNN gibi bölge bazlı nesne tespit algoritmaları önce muhtemel alanları belirleyip ardından ayrı ayrı CNN (Evrişimsel Sinir Ağı) sınıflandırıcıları yürütmektedir. Bu yöntem her ne kadar iyi sonuçlar verse de bir görüntü iki ayrı işleme tabi tutulduğu için görüntü üzerindeki işlem sayısı artar ve düşük bir FPS (saniye başına kare) alınmasına sebep olmaktadır.
Çalışmada kullanılan veri seti 399 adet görüntüden oluşmaktadır [70]. Kusurlu demiryolu ray görüntüleri görüntü artırım teknikleriyle 939'a arttırılmıştır. Bu görüntü artırım teknikleri görüntü çevirme, parlaklık ve gürültüdür [71]. Görüntü çevirme, x ekseni veya y ekseni boyunca görüntünün çevrilmesiyle yeni bir görüntü elde eder. Gürültü ekleme işlemi, orijinal görüntüden farklı güçlendirilmiş bir görüntü elde etmek için orijinal görüntünün her pikseline rastgele ek bilgiler eklenerek görüntünün gürültü artırmasını amaçlar. Bir görüntünün parlaklık değişimi, görüntü örneğinin her pikseli üzerinde doğrudan bir doğrusal dönüşüm işlemidir. Görüntü parlaklığı dönüşüm faktörü olarak λ kullanılarak, parlaklık değişimiyle genişletilen görüntü örneği, I=λI olarak ifade edilebilir. Burada 0< λ<1 koyulaştırmayı ve λ>1 parlaklaştırmayı temsil eder. Veri setinde 810 görüntü eğitim, 90 görüntü değerlendirme ve 39 görüntüde test için

ayrılmıştır. Veri setinde toplam 8 çeşit ray kusuru bulunmaktadır. Bunlar; aşınma (scratch) ray kafasının yan düzlemlerinde küçük/hafif çizik, ezilme (crush) yani demiryolunun yanal düzlemlerinin büyük/şiddetli aşınması, kir (dirt) rayın yüzeyini kaplayan boya veya çamur, boşluk (gap) bir demiryolu hattında birbirini takip eden raylar arasında bırakılan boşluklar, bilinmeyen (unknown) tanımlanamayan kusur, zarar görmüş (damage), göçük (dent), slant (eğik) gibi kusurlardır. Şekil 3.47'de veri setinden örnek görüntüler gözükmektedir.

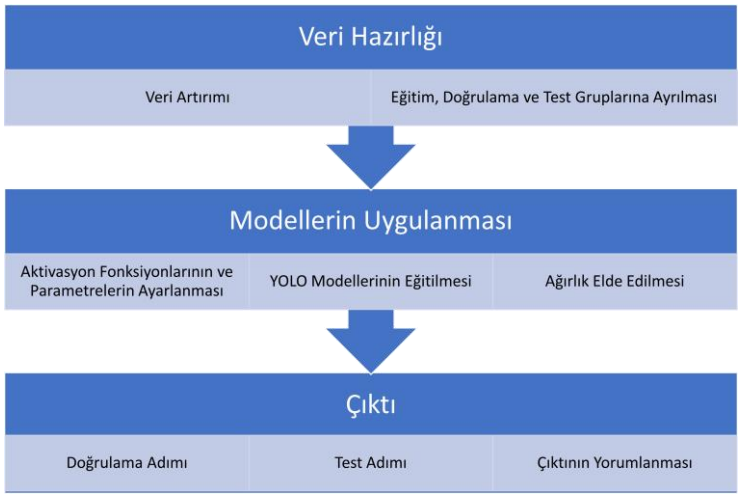



Şekil 3. 47. Veri setinden örnekler

Var olan 399 görüntüye görüntü çevirme, parlaklık ve gürültü ekleme işlemleri uygulanmış, bu sayede veri seti 939 görüntü içerecek şekilde genişletilmiştir. Daha sonra adım adım YOLO algoritmalarının kodları çalıştırılarak eğitim, doğrulama ve test işlemleri gerçekleştirilmiştir. Tüm modellerde kullanılan eğitim parametreleri ayarları:

- Girdi görüntü boyutu: 416
- İterasyon sayısı: 200
- Batch (bir iterasyon içerisinde her aşamada incelenen görüntü sayısı): 32

Önerilen modelin akış diyagramı Şekil 3.48'deki gibidir.




Şekil 3. 48. Önerilen modelin akış diyagramı

### 3.4.1. YOLO

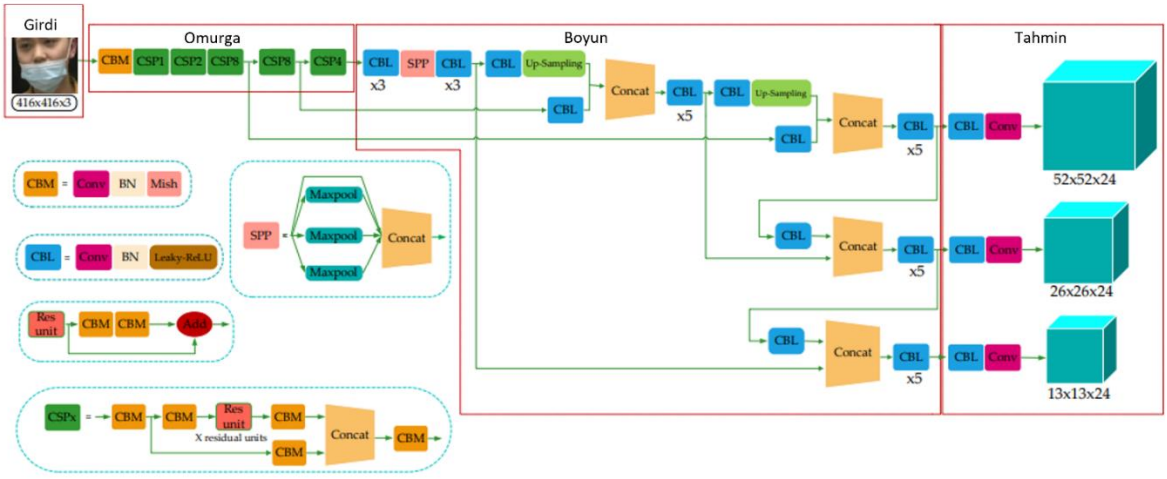
YOLO (Sadece Bir Kez Bak), sınırlayıcı kutuları ve sınıf olasılıklarını tahmin etmek için tek bir sinir ağı kullanan, GoogLeNet'ten ilham alan bir derin evrişimsel sinir ağıdır. Bu, görüntünün sadece bir kez ağ içinden geçip algılama görevini tamamladığı anlamına gelir. Literatürdeki en hızlı genel amaçlı nesne algılama mimarisidir. Ayrıca, rapor edilen ilk gerçek zamanlı evrişimsel sinir ağı tabanlı nesne algılama modelidir. Tüm algılama hattı tek bir ağda olduğundan doğrudan algılama performansı diğer sinir ağı mimarilerinin çoğundan yüksektir [72]. Şekil 3.49'da YOLO'nun katmanlı mimarisi görülmektedir.


Şekil 3. 49. YOLO mimarisi

YOLO mimarisi, resim sınıflandırması için GoogLeNet modelinden esinlenmiştir. YOLO sinir ağı modeli 24 evrişim katmanına ve ardından 2 tamamen bağlı katmana sahiptir. Ağın, başlangıçtaki evrişim katmanları görüntünün özelliklerini çıkartırken, tamamen bağlantılı katmanlar çıkış olasılıklarını ve koordinatları öngörürler. YOLO, C ve CUDA ile yazılmış açık kaynaklı bir sinir ağı uygulama çatısı olan "Darknet" üzerinde uygulanmaktadır. YOLO bu şekilde uygulanarak, gerçek zamanlı video akışını işlemek için GPU işlem gücünden yararlanmaktadır.

### 3.4.2. YOLOv4

YOLOv4, Nisan 2020'de yayınlanan ve COCO veri setinde üzerinde iyi bir performansa ulaşan gerçek zamanlı bir nesne algılama modelidir. YOLOv4 modeli YOLOv3'e dayalı optimize edilmiş bir modeldir. Şekil 3.50'de görüldüğü üzere üç kısımdan oluşmaktadır; omurga, boyun ve tahmin. Omurga kısmında CSPDarketNet53 ağı kullanılmıştır. Omurga kısmında ayrıca bag of freebies (BoF) ve bag of specials (BoS) adlı iki bölüm daha bulunmaktadır. Bu bölümler veri setini zenginleştirmek için kullanılmaktadır [73]. Boyun, nesneleri farklı ölçeklerde algılamak için kullanılır. Kafa bölümünü besleyen bilgileri zenginleştirmek için aşağıdan-yukarı, yukarıdan-aşağı akıştan gelen özellik haritaları kafa bölümünü beslemeden önce eleman bazında veya birleştirilerek birbirine eklenir [73]. Boyun yapısı SPP ve PAN'dan oluşmaktadır. Kafa bölümü ise tahmin katmanıdır.

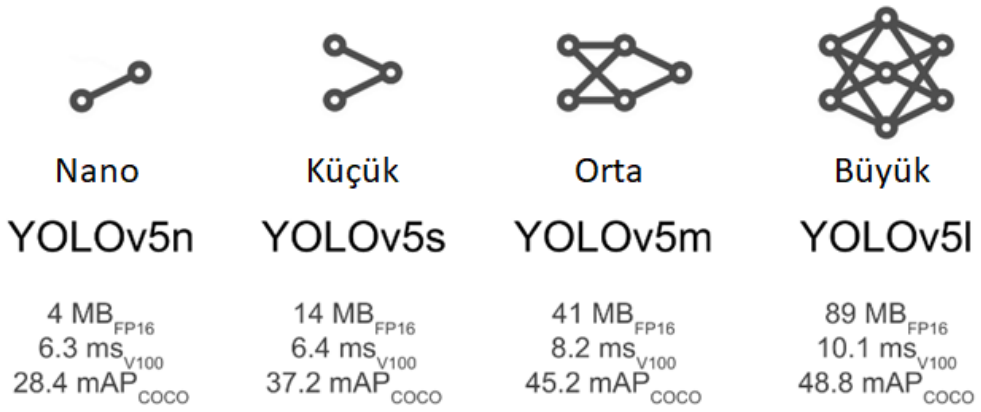



Şekil 3. 50. YOLOv4 mimarisi

### 3.4.3. YOLOv5

YOLOv4'ün yayınlanmasından kısa bir süre sonra Glenn ve ekibi tarafından mimarinin yeni sürümü olan YOLOv5 yayınlanmıştır. Mimarinin önceki sürümlerine göre işlem süresini önemli ölçüde düşürmesiyle birlikte YOLOv5, Pytorch'ta yeni bir eğitim ortamı altında derlenerek eğitim sürecini Darknet'ten daha kolay bir hale getirmiştir. Bu ağ modelinin algılama doğruluğu ve hızı önceki sürümlere kıyasla oldukça yüksektir. Buna ek olarak, YOLOv5'in ağ modelinin ağırlık dosyasının boyutları da önceki sürüm olan YOLOv4'ten yaklaşık %90 oranla daha küçüktür. Bütün bu avantajlar YOLOv5 modelinin gerçek zamanlı tespit uygulamaları için gömülü cihazlar üzerinde çalışması için uygun olduğunu göstermektedir. YOLOv5 mimarisi boyut ve model parametrelerinin miktarına göre artan dört alt mimariyi içermektedir. Bunlar;

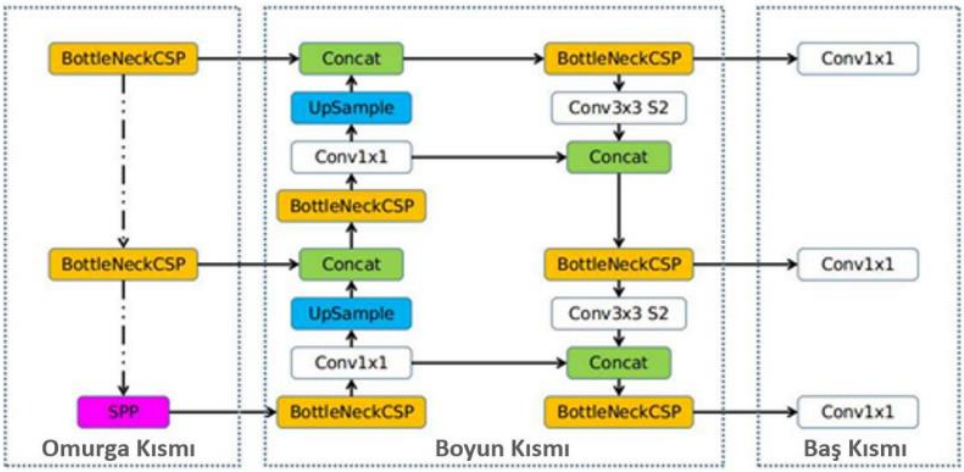
YOLOv5s, YOLOv5n, YOLOv5m, YOLOv5l ve YOLOv5x mimarileridir. Bu mimariler sinir ağının belirli bir yerindeki özellik çıkarma modüllerinin ve evrişim çekirdek miktarlarının farklı olmasıyla birbirinden ayrılırlar. Model boyutları ve parametre sayısı da sırayla artmaktadır. Bu artış Şekil 3.51'de görülmektedir.




Şekil 3. 51. YOLOv5 alt mimarileri

Bu çalışma kapsamında geliştirilen sürücü asistan sistemi uygulamaları gömülü platform üzerinde gerçek zamanlı olarak çalışacağından ve sistemin minimum kaynak kullanımı ile maksimum performansa ulaşması istendiğinden ağın eğitimi YOLOv5s [74] ile gerçekleştirilmiştir. Genel yapısı Şekil 3.52'de verilen YOLOv5 ağ mimarisi omurga, boyun ve baş kısmı olmak üzere üç bölümden oluşmaktadır.

Omurga kısmında CSPDarknet [74], boyun kısmında PANet [74] ve baş kısmında ise YOLO katmanı kullanılmaktadır. Özetle, veriler özellik çıkarımı için ilk olarak CSPDarknet'den geçirilerek özellik birleştirme için PANet'e verilir. Bunun sonucunda da YOLO katmanı sınıf, skor, konum ve boyut bilgisi gibi çıktıları vermektedir.




Şekil 3. 52. YOLOv5 mimarisi

Bu çalışmada ayrıca YOLOv5 ile eğitilmiş ağırlıkları kullanarak topluluk modellemesi yöntemi kullanılmıştır. Topluluk modelleme işleminin mAP.5 ve geri çağırma değerleri üzerinde pozitif etki yaptığı saptanmıştır. Topluluk modellemesi birçok farklı modelleme algoritması kullanarak ya da farklı eğitim veri setleri kullanarak bir sonucu tahmin etmek için çok çeşitli modellerin oluşturulduğu bir süreçtir. Topluluk modeli her bir temel modelin tahminini toplar ve görünmeyen veriler için nihai bir tahminle sonuçlanır. Topluluk modellerini kullanma amacı, tahminin genelleme hatasını azaltmaktır. Temel modeller çeşitli ve bağımsız olduğu sürece, topluluk yaklaşımı kullanıldığında modelin tahmin hatası azalmaktadır. Topluluk modeli, model içinde birden fazla temel modele sahip olsa da tek bir model olarak hareket eder ve çalışır [75]. Çalışmada YOLOv5'in alt mimarileri olan YOLOv5s, YOLOv5n, YOLOv5m, YOLOv5l kullanılarak veri seti eğitilmiştir. Eğitim sonucu oluşan ağrılıklar kullanarak topluluk modellemesi yapılmıştır.

### 3.4.4. YOLOv6

YOLOv6 modeli Haziran 2022'de Meituan tarafından piyasaya sürülmüştür. COCO veri kümesi karşılaştırmasında iyi sonuçlar vermiştir. YOLOv6, modeli YOLO mimarisi temelinde inşa edilmiş ve YOLO ailesinin diğer modellerine göre çeşitli iyileştirmeler ve yeni yöntemler sunmuştur. YOLOv6, PyTorch'ta yazılmıştır. YOLOv6'nın üç önemli güncellemesi var: donanım dostu omurga ve boyun tasarımı, verimlilik için ayrılmış baş ve etkili eğitim stratejileri. YOLOv6'nın nesne algılama performansının, algoritmanın aşamalı sürümleriyle hem hız hem de doğruluk iyileştirmeleri ile diğer CNN tabanlı algoritmalarla karşılaştırılabilir olduğu gösterilmiştir [76]. Çalışmada farklı görüntü boyutlarının YOLOv6 alt mimarilerinin performanslarına etkisi gösterilmiştir. Çalışmada YOLOv6 modeli için kullanılan veri seti 416x416 ve 640x640 boyutlarında iki versiyona türetilmiştir.

### 3.4.5. Metrikler

Model eğitimi tamamlandıktan sonra modelin test edilmesi için eğitilen ağırlıklar kullanılır ve model birçok yönden değerlendirilir. Ray kusurlarını içeren veri setimiz için test sonuçları üç kategoride sınıflandırılabilir: TP (gerçek pozitif), test setindeki kategorilerin test sonuçlarıyla aynı olduğu anlamına gelir; FP (yanlış pozitif), tespit edilen nesne kategorisindeki örneklerin sayısının gerçek nesne kategorisiyle tutarsız olduğu anlamına gelir ve FN (yanlış negatif), gerçek numunenin ters sonuç olarak veya tespit edilmeyen kategoride tespit edildiğini gösterir. Model tarafından değerlendirilen tüm pozitif vakalar için sayı (TP + FP), bu nedenle gerçek vakaların (TP) oranına kesinlik (precision) oranı denir ve bu, pozitif vakalardaki gerçek vaka

örneklerinin, tarafından tespit edilen numuneler arasındaki oranını temsil eder. Denklem (3.22)'de gösterilmiştir.

$$Precision = \frac{TP}{TP+FP} \tag{3.22}$$

Test setindeki tüm pozitif örnekler için sayı (TP + FN)'dir. Bu nedenle, denklem (3.23)'de gösterildiği gibi, geri çağırma (recall) oranı, modelin test setindeki gerçek durumları tespit etme yeteneğini ölçmek için kullanılır [77].

$$Recall = \frac{TP}{TP+FN} \tag{3.23}$$

Modelin kesinliğini karakterize etmek için bu makale, Denklemler (3.24) ve (3.25)'da gösterildiği gibi modelin doğruluğunu değerlendirmek için AP (ortalama kesinlik) ve mAP (ortalama kesinlik değerlerinin ortalaması) göstergelerini tanıtmaktadır [77].

$$AP = \int_0^1 P(R)dR \tag{3.24}$$

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{3.25}$$

Bunlar arasında sırasıyla P, R, N, tüm kategorilerdeki kesinliği, geri çağırma oranını ve toplam nesne sayısını temsil eder [77].

## 3.5. Bulanık Ölçüm ve Evrişimli Sinir Ağlarına Dayalı İki Aşamalı Ray Kusur Sınıflandırması

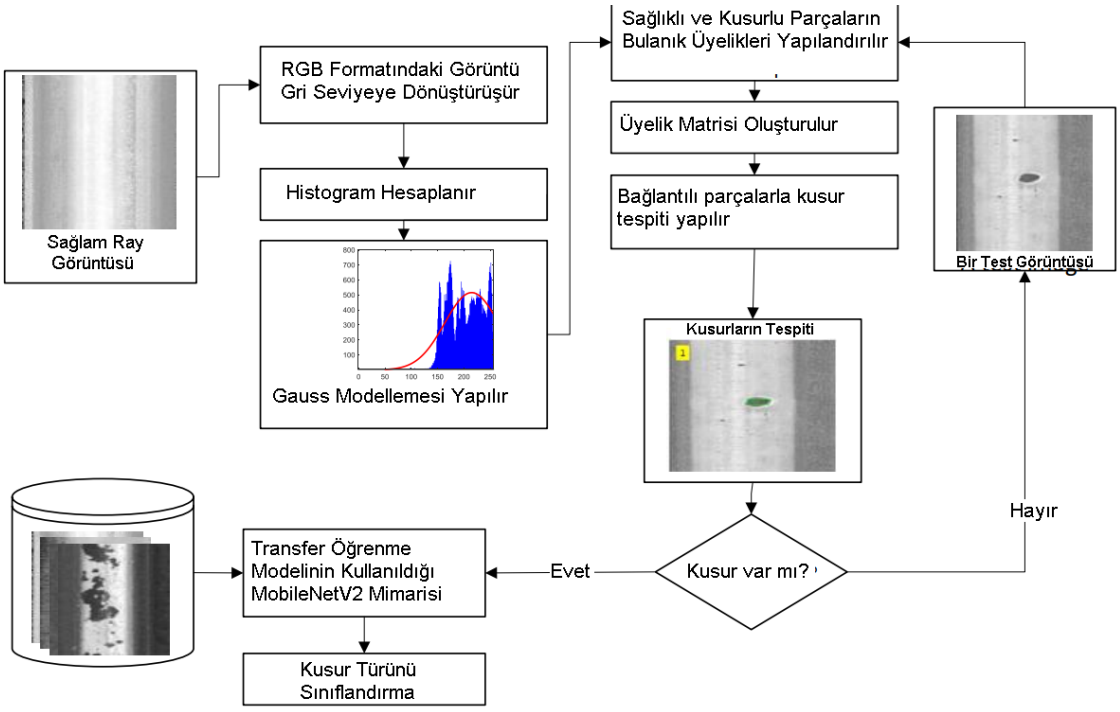
Bu çalışmada, ray yüzey kusurlarının tespiti ve sınıflandırılması için iki aşamalı bir yaklaşım sunulmaktadır. İlk aşamada, sağlıklı ray yüzeyini temsil eden görüntünün histogramı bir Gauss fonksiyonu ile modellenir ve temsil eden bulanık üyelik fonksiyonu oluşturulur. Aynı şekilde kusurlu bölge için bir bulanık üyelik fonksiyonu oluşturulur ve kusurlar segmentlere ayrılır. İkinci adımda bir derin sinir ağı ile bir kusur tespit edilirse kusur tipi belirlenir. Önerilen yöntemin avantajları aşağıdaki gibidir.

- Hata tespiti için gerçek zamanlı bir çalışma yöntemi sağlanması,
- Derin öğrenme yöntemi ile farklı kusur türlerinin sınıflandırılması,
- Gürültüye duyarlı bir segmentasyon ile kusurlu bölgelerin hassas tespiti.

Önerilen yaklaşım, kusur türlerini belirlemek için bulanık ölçüm, klasik görüntü işleme ve derin öğrenmeyi birleştirir. Yaklaşım, kusur türünü tespit etmek ve belirlemek için yüksek performans ve gerçek zamanlı çalışma sunar.

### 3.5.1. Ray Yüzey Hatalarının İki Adımda Tespiti için Önerilen Yaklaşım

Kusur tespiti için önerilen yöntem, bulanık ölçüme dayalı arka plan modellemesine dayanmaktadır. Arka plan, sağlıklı ray için elde edilen histogram için bir Gauss fonksiyonu ile modellenmiştir. Ek olarak, arka planın dışındaki bölge ayrı bir Gauss fonksiyonu olarak modellenmiştir. Daha sonra, bir pikselin ait olduğu üyelik fonksiyonuna göre bir pikselin kusurlu olup olmadığı belirlenir. Arıza tespit edildikten sonra arıza tipi MobileNetv2 konvolüsyonel sinir ağı tarafından sınıflandırılır. Önerilen algoritmanın blok diyagramı Şekil 3.53'te verilmiştir. Şekil 3.53'te yüzey kusurlarını tespit etmek için önce bulanık ölçüme dayalı kısım işlenir. İlk modül bir kusur tespit ederse, eğitilmiş derin sinir ağı devreye girer ve kusurun türünü belirler




Şekil 3. 53. Önerilen yaklaşımın mimarisi

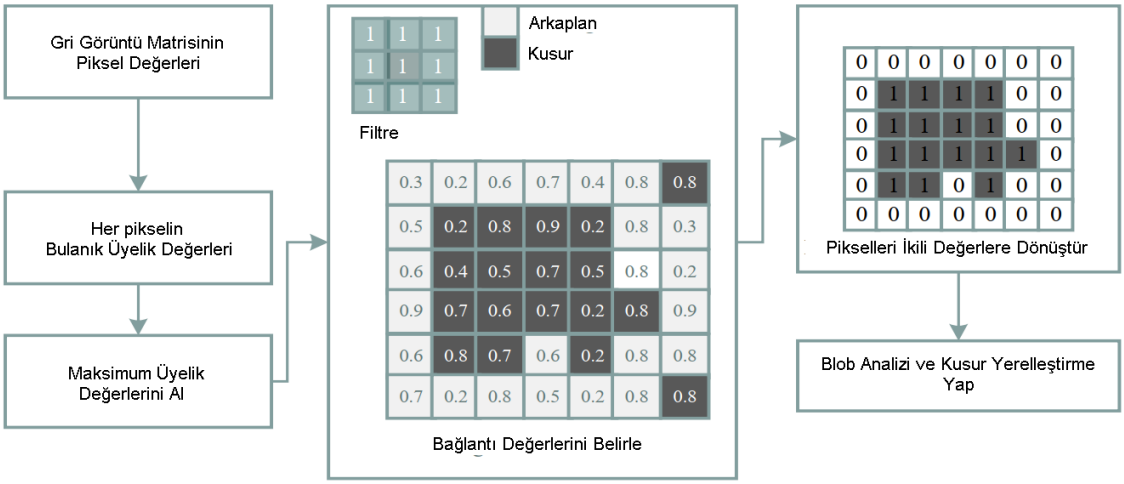
#### 3.5.1.1. Bulanık Ölçüme Dayalı Hata Tespiti

Hata tespiti için öncelikle sağlıklı raydan alınan RGB görüntüsü gri bir görüntüye dönüştürülür. Sağlıklı görüntünün gri histogramı daha sonra bir Gauss fonksiyonu olarak modellenir. Gauss fonksiyonu denklem (3.26)'da gösterilmiştir.

$$f(x,y) = \frac{y}{\sqrt{\pi\sigma}} e^{-\frac{(x-\mu)^2}{\sigma^2}} \quad (3.26)$$

Denklem (3.26)'te verilen Gauss fonksiyonunda μ ve σ değerleri sırasıyla ortalama ve standart sapma değerlerini gösterir. Denklemdeki x ve y değerleri sırasıyla giriş verilerini ve verilerin genliğini temsil eder. Histogram modellemede, x ve y değerleri sırasıyla histogramdaki piksel değerlerini ve her piksel değerinin oluşum sıklığını temsil eder. Histogramı Gauss fonksiyonuyla modellemek için μ ve σ parametrelerinin tahmin edilmesi gerekir. Bu iki parametrenin tahmini için Nelder-Mead simpleks yöntemine dayalı yöntem kullanılmıştır. Bu yöntem, n boyutlu bir vektör x için n+1 nokta kullanır. Başlangıçta en düşük fonksiyon değeri f(x1) ile başlar ve en kötü değeri simpleksten çıkarır ve yeni bir nokta ekler. Gauss fonksiyonunun genlik, ortalama ve standart sapma değerleri için başlangıç değerleri verilerek başlanır ve ardından optimum değer elde edilir.

Sağlıklı ray için histogramı temsil eden üyelik fonksiyonu elde edildik-ten sonra, sağlıklı histogramın dışındaki alanda hatalı ray için bir üyelik fonksiyonu oluşturulur. Bir sonraki adımda, herhangi bir test görüntüsünün piksel değerlerinin ait olduğu üyelik fonksiyonu belirlenir. Bu nedenle, bir piksel bir arka plan veya kusur olarak etiketlenir. Şekil 3.54, üyelik fonksiyonuna ait olup olmadıklarına bağlı olarak piksellerin arka plan veya kusur olarak belirlenmesi sürecini temsil etmektedir.




Şekil 3. 54. Arıza tespit adımları

Şekil 3.54'te, ilk olarak gri görüntü matrisinin iki üyelik fonksiyonuna atanmasıyla üyelik değerleri hesaplanmaktadır. Daha sonra Gauss bulanık fonksiyonu kullanılarak üyelik değerleri belirlenir ve bağlantılı kısımları bulmak için üyelik matrisi üzerinden 3x3 filtre kaydırılır. Bir pikselin 3x3 komşuluğundaki tüm pikseller kusur üyeliğine aitse, kusur olarak etiketlenir. Aksi takdirde ilgili piksel ikili görüntüde arka plan olarak alınır. Görüntü ikili

görüntüye dönüştürüldükten sonra blob analizi ile kusurlu bölgeler ve sayıları belirlenir. Bir kusur tespit edilirse kusur türünü belirlemek için eğitilmiş derin öğrenme yöntemi etkinleştirilir.

#### 3.5.1.2. MobileNetv2 Tabanlı Hata Türlerinin Sınıflandırılması

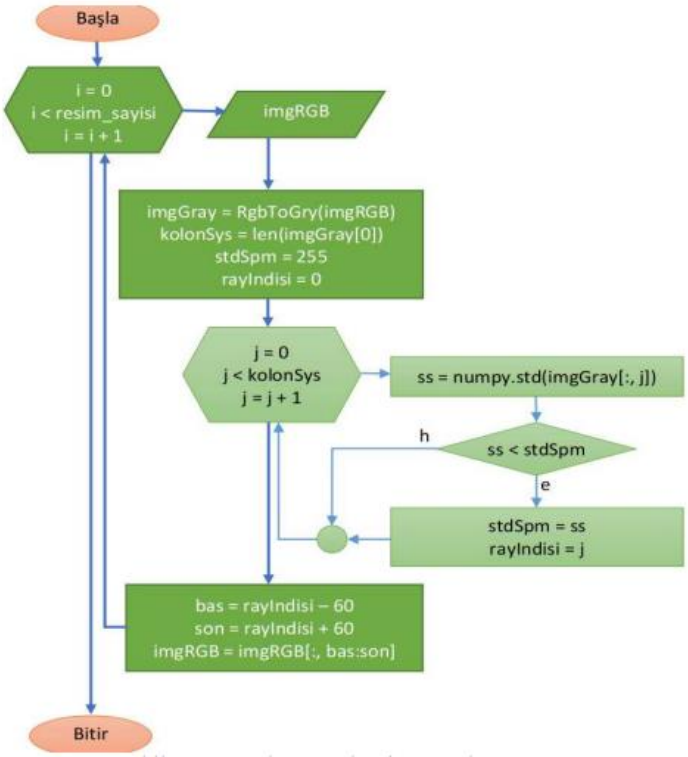
Bulanık ölçüm ile hata tespit edildikten sonra hatanın türünü belirlemek için transfer öğrenme yöntemi kullanılır. Bu amaçla düşük ağırlıklı bir ağ olan MobileNetv2 kullanılmıştır [78]. MobileNetv2 modeli toplamda 53 katmandan oluşsa da düşük parametre sayısı ve yüksek performansı nedeniyle tercih edilmektedir. Bu derin öğrenme ağı, derinlemesine evrişim katmanları ve ters çevrilmiş artık bloklar kullandığından, parametre sayısı çok düşüktür.

## 3.6. İki Derin Öğrenme Yöntemiyle Ray Yüzeyi Kusurlarının Tespiti: Karşılaştırmalı Analizi

Çalışmada drone yardımı ile elde edilen ray görüntüleri kullanılmıştır. İmgeler 300px eninde, 540px boyunda ".png" formatındadır. Toplam 2000 adet imge, kırık veya sağlam olmak üzere iki sınıftan birine dahil olmaktadır. İmgeler "raylar" klasörü altında kendi sınıfına ait "Kırık" ve "Sağlam" alt klasöründe barındırılmaktadır. "Kırık" sınıfı 1010, "Sağlam" sınıfı 990 imgeden oluşmaktadır. Bu imgelerin rastgele 400 tanesi test için ayrılmıştır. Test için ayrılan imgelerin 202 tanesi "Kırık", 198 tanesi "Sağlam" sınıfına aittir. İmgeler "raylar" klasörü altında kendi sınıfına ait "Kırık" ve "Sağlam" alt klasöründe barındırılmaktadır. "Kırık" sınıfı 1010, "Sağlam" sınıfı 990 imgeden oluşmaktadır. Bu imgelerin rastgele 400 tanesi test için ayrılmıştır. Test için ayrılan imgelerin 202 tanesi "Kırık", 198 tanesi "Sağlam" sınıfına aittir.

### 3.6.1. Görünü Ön İşleme

ESA, küçük görüntüler üzerinde daha hızlı çalışmaktadır. Öncelikle rayın imge üzerindeki yerinin tespit edilmesi gerekir. Ardından yeri tespit edilen rayın sol ve sağındaki gereksiz alanlar belirli bir miktarda kırpılarak temizlenebilir. Rayın tespiti ve kırpılacak indislerin belirlenmesi için geliştirilen algoritmanın akış şeması Şekil 3.55'te görülmektedir




Şekil 3. 55. Ray çıkarma algoritması akış şeması

Ray tespiti için geliştirilen algoritmada imgeler gri seviye olarak alınmaktadır. Gri seviye imge üzerinde histogram eşitleme yapılarak imge iyileştirilir. Bu sayede imge üzerindeki ayrıntılar daha belirgin hale gelmektedir. İyileştirilen gri seviye imge 2 boyutlu bir matristir. İmge matrisinin her sütunundaki piksel değerlerinin standart sapması hesaplanır. İmge üzerinde dikey görünen ray piksellerinin sayısal değerleri birbirine yakındır. Bu nedenle ray üzerinden geçen sütunlardaki piksel değerlerinin standart sapması düşüktür.

### 3.6.2. Çalışmada Kullanılan ESA Modelleri

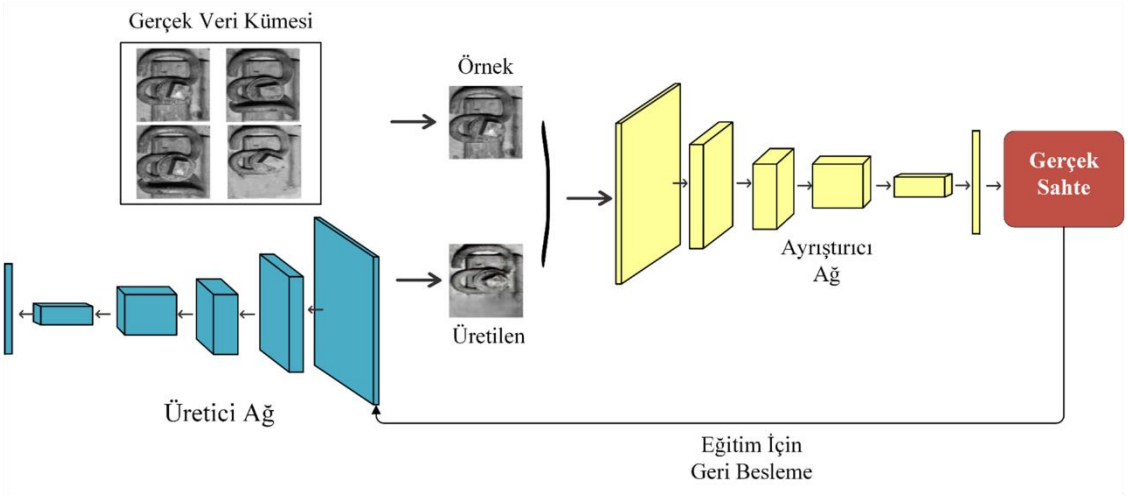
Ön işleme neticesinde ESA için daha kullanışlı hale getirilen veriler, GoogleNet ve SqueezeNet mimarileri ile denenmiştir. SqueezeNet 2016 yılında Iandola ve ark. tarafından geliştirilmiş bir mimaridir. SqueezeNet tasarlanırken, azaltılmış parametreler ile daha hızlı çalışabilecek bir ESA yapısı hedeflenmiştir. AlexNet'ten 50 kat az parametre ile çalışmakta ve 0.5 MB'den daha az bellek alanı kaplamaktadır [64].

## 3.7. DCGAN ve Siyam Sinir Ağını Kullanarak Demiryolu Bağlantı Elemanlarındaki Kusurların Tespiti

Geleneksel bir evrişimli sinir ağı küçük bir veri seti ile özellikleri öğrenemez. Eğitim işlemi için sağlam bağlantı elemanlarından oluşan veri setini oluşturmak kolay olmasına rağmen kusurlu bağlantı elemanlarından oluşan veri setini oluşturmak oldukça zordur. Bu tür veri setini oluşturmak için yüzlerce kilometre demiryolundan görüntü toplanması gerekebilir. Bu nedenle bu çalışmada, DCGAN kullanılarak yapay deforme bağlantı elemanı görüntüleri oluşturulup veri seti çoğaltılmıştır. Ardından, siyam sinir ağı ile bağlantı elemanlarının kusur durumu incelenmiştir. Çalışmada, sağlam ve deforme olmak üzere iki bağlantı elemanı sınıfı bulunmaktadır. Her sınıf için farklı sınıfların görüntüleri arasındaki benzerlik puanları hesaplanmıştır. Temel fikir, bağlantı elemanlarını benzerlik puanlarını kullanarak ve karşılaştırma yaparak tanımlamaktır. Deneysel sonuçlarda, önerilen yöntem için %98,23 doğruluk oranı elde edilerek, geleneksel yöntemlere göre avantajı gösterilmiştir.

### 3.7.1. DCGAN ile Eğitim Görüntülerinin Çoğaltılması

Derin Evrişimli Çekişmeli Üretici Ağlar (DCGAN), görüntü işleme yeteneklerinden yararlanmak için Üretici-Ayrıştırıcı çerçevesini kullanarak gürültülü verilerden belirli bir dağıtıma ait görüntüleri oluşturmak için esasen derin evrişimli sinir ağlarından yararlanır. DCGAN, GAN ağının bir varyasyonudur. DCGAN yapısı Şekil 3.56'da verilmiştir. DCGAN yapısında, çekişmeli olarak aynı anda iki model eğitilir. Eş zamanlı olarak eğitilen üretici ve ayrıştırıcı ağ olmak üzere iki bölümden oluşur. Ayrıştırıcı model, örneğin gerçek mi yoksa sahte mi olup olmadığını belirler.




Şekil 3. 56. DCGAN yapısı

Ayrıştırıcı model, eğitim sisteminde üretici modelin verilerini gerçek ve sahte olarak sınıflandıran ikili bir sınıflandırıcıdır. DCGAN modelinin çekişmeli eğitim süreci denklem (3.27) kullanılarak hesaplanır.

$$minmax(D,G) = E_{x-pdata(x)}[logD(x)] + E_{x-p(x)}log[1 - D(G(z)))] \quad (3.27)$$

Üretici, girdi verisi ve ayırıcıdan geri bildirim olarak rastgele oluşturulmuş bir gürültü vektörü alır ve mümkün olduğunca gerçek görüntülere yakın yeni görüntüler üretir. Ayrıştırıcı, üreticinin çıktısını eğitim verisi olarak kullanır. Üretici, ayrıştırıcıdan geri bildirim alır. Böylece her model bu süreçte daha güçlü hale gelir. Üretici, yeni görüntüler oluşturmaya ayrıştırıcı ise oluşturulan görüntüler ile gerçek eğitim görüntüleri arasındaki farkı artık belirleyemeyene kadar sürecini iyileştirmeye devam eder. Bu süreç dengeye ulaştığında ayrıştırıcı artık gerçek görüntüleri sahtelerden ayırt edemez. Bu şekilde, en gerçekçi yapay görüntüler elde edilmiş olur.

### 3.7.2. Siyam Ağı Modeli

Kusur tespiti görevlerinde, sinir ağları çoğu zaman iyi sonuç verir, ancak sinir ağları iyi performans gösterebilmek için daha fazla veriye ihtiyaç duyar. Daha önce belirtildiği gibi, deforme bağlantı elemanlarından oluşan veri kümesi oluşturmak zordur. Bu tür az sayıda örnek içeren veri kümelerinde görevleri çözmek için Siyam Ağları adı verilen mimari kullanılır. Az veriden öğrenme yeteneği, Siyam ağlarını son yıllarda daha popüler hale getirmiştir. Siyam Sinir Ağı, iki veya daha fazla aynı alt ağ içeren bir sinir ağı mimarisidir (Şekil 3.57). Alt ağlar, aynı parametreler ve ağırlıklarla aynı konfigürasyona sahip oldukları anlamına gelir. Parametre güncellemesi her iki alt ağa da yansıtılır. Özellik vektörlerini karşılaştırarak girişlerin benzerliğini bulmak için kullanılır, bu nedenle bu ağlar birçok uygulamada kullanılır. Siyam ağlarında sinir ağı, özellik vektörlerini kullanarak iki görüntü arasında ayrım yapmayı öğrenir. Siyam ağının amacı, benzerlik skorunu kullanarak iki girdinin aynı mı yoksa farklı mı olduğunu sınıflandırmaktır. Benzerlik puanı, genel uzaktan metrik öğrenme yaklaşımı için teknikler olan ikili çapraz entropi, karşıtlık işlevi veya üçlü kayıp kullanılarak hesaplanabilir. Siyam ağının eğitimi için ilk olarak farklı sınıfları içeren veri kümesi eklenir. Ardından, pozitif ve negatif veri çiftleri oluşturulur. Pozitif veri çifti, her iki girişin de aynı olması, negatif veri çifti ise iki girişin farklı olması durumudur. Tam bağlantılı bir katman kullanarak özellik kodlamasının çıktısını veren evrişimli sinir ağını oluşturulur. Bu ağ, iki girişin seçildiği kardeş CNN'dir. Kardeş CNN'ler aynı mimariye, hiperparametrelere ve ağırlıklara sahip olmalıdır. Çıktıyı kodlayan iki kardeş CNN ağı arasındaki Öklid mesafesini hesaplamak için fark oluşturma katmanı oluşturulur. Son olarak, benzerlik puanının çıktısını almak için sigmoid etkinleştirme işlevini kullanan tek bir düğüme sahip tam bağlantılı bir katman oluşturulur ve model ikili çapraz entropi kullanılarak derlenir. Siyam ağlarını eğitirken aynı çiftlere ve farklı çiftlere ihtiyaç vardır. Bu iki görüntü, sinir ağı katmanından geçtikten sonra her görüntü için h1 ve h2 özellik vektörleri elde edilir. Özellik vektörleri, iki görüntünün benzerliğini belirlemek için karşılaştırılır. Karşılaştırma işleminde

özellik vektörleri arasındaki mesafe ölçülür. İki görüntü arasındaki mesafe küçükse vektörler benzerdir, mesafe büyükse vektörler farklıdır. Mesafe ölçümü Öklid uzaklığı ile yapılmıştır. Öklid formülü denklem (3.28)'de verilmiştir. Denklemde, x ve y modelin ürettiği iki vektöre karşılık gelmektedir.

$$D = \sqrt{\sum_{i=1}^{n}(x_{i-}y_i)^2} \tag{3.28}$$

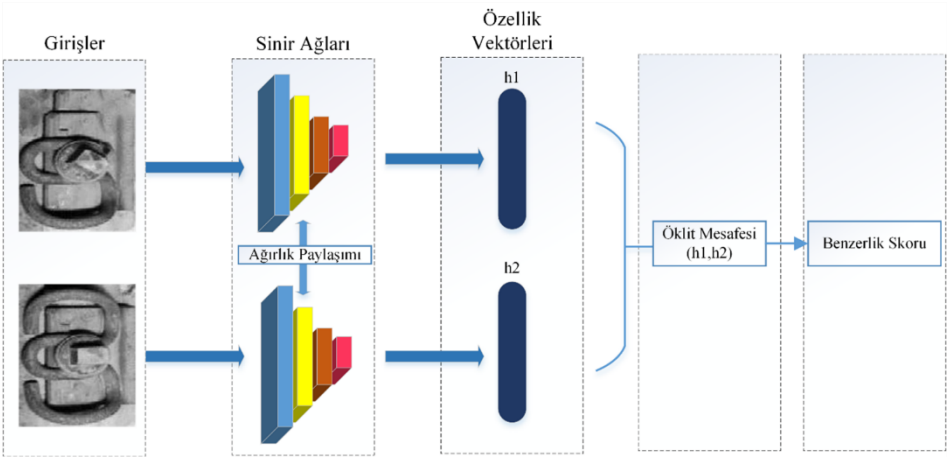
Eğitim sırasında ikili çapraz entropi, karşıtlık işlevi veya üçlü kayıp gibi farklı hata fonksiyonları kullanılabilir. Hata fonksiyonlarının temel amacı, ağın iki görüntü çiftini ne kadar iyi ayırt ettiğini değerlendirmektir. Bu çalışmada, kullanılan hata fonksiyonu ikili çapraz entropidir. Amaç, girdilerin aynı sınıfta olup olmadığını sınıflandırarak çıktıda ikili sınıflandırma gerçekleştirmektir. İkili çapraz entropi'de kayıp denklem (3.29)'daki gibi hesaplanır.

$$L = -y \log p + (1 - y) \log(1 - p) \tag{3.29}$$

Burada L kayıp fonksiyonu, y 0 veya 1 olmak üzere sınıf etiketi ve p tahmini ifade etmektedir. Denklem (3.30)'a göre ağı, benzer ve farklı görüntüler arasında ayrım yapacak şekilde eğitmek için, her etapta bir pozitif ve bir negatif örnek beslenebilir ve kayıplar toplanabilir. Bu kayıp işlevi, ağı eğitmek için kullanılır.

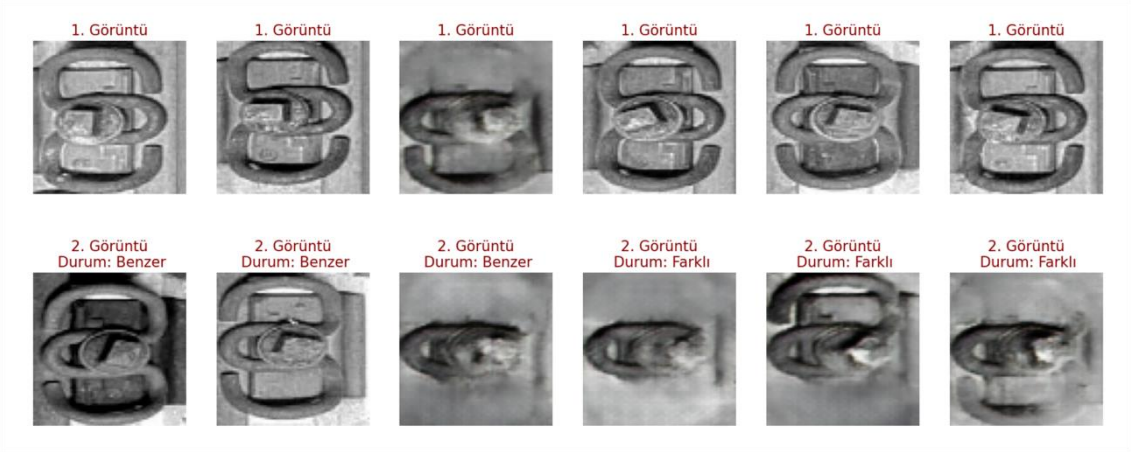
$$L = L_{pozitif} + L_{negatif} \tag{3.30}$$

Siyam ağlarını eğitirken aynı çiftlere ve farklı çiftlere ihtiyaç vardır. Aynı çiftler, aynı veri kümesine ait iki görüntüden oluşur. Farklı çiftler ise farklı veri kümesine ait iki görüntüden oluşur. Deforme bir bağlantı elemanı ile sağlam bağlantı elemanı negatif çift oluştururken deforme iki bağlantı elemanı pozitif çift oluşturmaktadır. Siyam ağını eğitmek için, görüntü çifti örnekleri rastgele seçilmiştir. Siyam ağı bu çiftleri kullanarak, benzerliği öğrenmektedir.



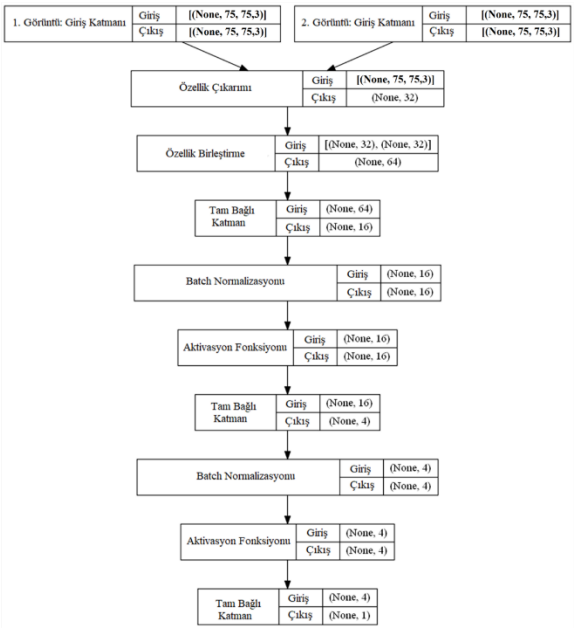

Şekil 3. 57. Siyam sinir ağı yapısı

Eğitim için pozitif ve negatif çift örnekleri oluşturulurken veri kümesinden rastgele örnekleme yapılır. Şekil 3.58'de rastgele seçilen bazı görüntü çiftleri verilmiştir. Seçilen görüntüler, benzer ise pozitif çift, farklı ise negatif çift oluşturur.




Şekil 3. 58. Rastgele seçilen bazı görüntü çiftleri

Ağlar için giriş görüntülerinin uzamsal boyutları genişlik, yükseklik ve sayı kanalı şeklindedir. Veri kümemiz için giriş görüntüleri 75x75x3 şeklindedir. Evrişimli sinir ağları, sınıflandırma ve özellik çıkarma kullanılır. Bu çalışmada, önerilen sinir ağı sınıflandırma için kullanılan sinir ağından farklıdır. Önerilen sinir ağının çıktısı bir özellik vektörüdür. Aktivasyon fonksiyonu olarak Relu fonksiyonu kullanılmıştır. Siyam ağında kullanılan ağın yapısı Şekil 3.59'da verilmiştir.

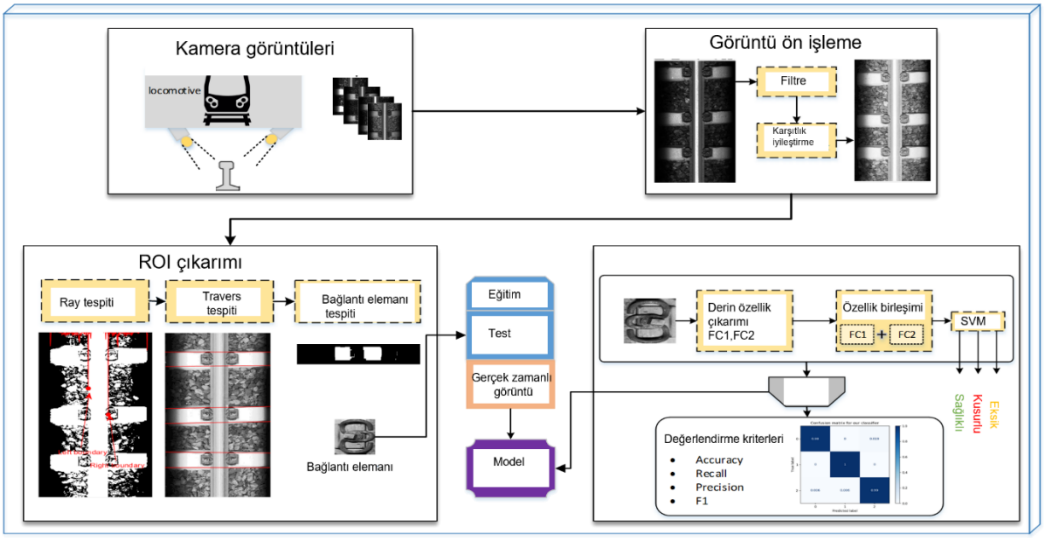



Şekil 3. 59. Siyam sinir ağı modelinde kullanılan katmanlar ve parametreler

Veri kümesi, demiryolu sahasından elde edilen görüntülerden bağlantı elemanları kırpılarak oluşturulmuştur. Veri kümesinde, eğitim için 400 sağlam ve 400 kusurlu olmak üzere toplam 800 adet, test için ise 100 sağlam ve 100 kusurlu olmak üzere toplam 200 adet bağlantı elemanı görüntüsü bulunmaktadır. Eğitim veri kümesine yapay bağlantı elemanı görüntüleri dâhildir. Test veri kümesine ise yapay bağlantı elemanı görüntüleri dâhil değildir. Veri kümesinde toplam 1000 görüntü vardır ve görüntülerin çözünürlüğü 227×227 pikseldir.

## 3.8. Görüntü Ön İşleme ve Hafif Evrişimsel Sinir Ağı Kullanarak Demiryolu Bağlantı Elemanlarının Kusur Sınıflandırması

Bu çalışmada, ray bağlantı elemanlarındaki kusurların sınıflandırılması için dört önemli adımdan oluşan yeni bir yaklaşım önerilmiştir. İlk adımda, bir kamera yardımıyla demiryollarından görüntüler toplanır. Ray görüntüleri genellikle trenin altına monte edilmiş bir kameradan alındığından aydınlatma koşulları optimal değildir, bu nedenle ray görüntüleri çok fazla gürültü içerir. İkinci adımda, yüksek kontrastlı bir ray görüntüsü elde etmek için bir görüntü iyileştirme yöntemi uygulanır. Elde edilen görüntüde gürültü ve ışık efektleri azaltılır. Üçüncü adımda, bağlantı elemanı bileşeni, rayları, traversleri, altyapıyı ve bağlantı elemanlarını içeren genel görüntüde tam olarak algılanır. Ray ve travers konumu belirlendikten sonra travers kısmına LLBP yöntemi uygulanarak bağlantı elemanının konumu elde edilir. Son adımda, bir LCNN tarafından elde edilen özellikler kullanılarak sağlıklı, hasarlı ve eksik bağlantı elemanları türleri sınıflandırılır. Önerilen yöntemin blok şeması Şekil 3.60'ta verilmiştir.




Şekil 3. 60. Önerilen yöntemin akış şeması

Şekil 3.60'ta ilk olarak, ray, travers ve bağlantı elemanı dahil olmak üzere ray bileşenlerinin, trenin altına monte edilmiş yüksek hızlı bir kamera ile toplanan görüntüleri gösterilmektedir. Filtreleme adımında yakalanan görüntülere histogram eşitleme uygulanır. Bu işlemle, en yaygın piksel değerleri tam gri ton aralığına yayılır, böylece görüntünün yoğunluk aralığı genişletilir. Daha sonra ray görüntüsü üzerinde görüntü iyileştirme işlemi yapılır ve gürültü giderilir. Üçüncü aşamada, ray ve traverslerin konumları belirlenir ve travers bloklarını içerdiği belirlenen görüntülere LLBP algoritması uygulanır. LLBP algoritması bağlantı elemanı bileşenlerini ortaya çıkarsa da herhangi bir deliği kapatmak için bir morfolojik operatör uygulanır. Son aşamada, yeni geliştirilen LCNN'ye konumlanan bağlantı elemanı bileşenleri verilir ve elde edilen özellikler destek vektör makineleri kullanılarak sınıflandırılır.

### 3.8.1. Görüntü Ön İşleme ve Geliştirme

Renkli bir görüntü, Kırmızı (R), Yeşil (G) ve Mavi (B) olmak üzere üç bileşenden oluşur. Bu matrislerin her biri, görüntüyle ilişkili yoğunluk değerlerini tutar. Görüntüdeki herhangi bir (x, y) konumundaki pikselin değeri, R(x, y), G(x, y) ve B(x, y) içindeki değerlerle temsil edilir. Genel olarak, aşağıdaki dönüştürmeyi kullanarak renkli bir görüntüyü gri tonlamalı bir görüntüye dönüştürürken üç katsayı gerekir denklem (3.31)'deki gibi.

$$L = -y \log p + (1 - y) \log(1 - p) \tag{3.31}$$

Bu katsayıların kr, kg ve kb değerleri genellikle kr = 0.2989, kg = 0.5870 ve kb = 0.1140 olarak seçilir. Trenin alt kısmından elde edilen görüntülerin kontrast değeri düşük olduğu için ilk adım olarak görüntü iyileştirme işlemi uygulanmaktadır. Yüksek kontrastlı ve düşük homojenliğe

sahip bir görüntü elde etmek için uygun katsayılar belirlenmelidir. Denklem (3.31), bu genellemeyi kaybetmeden kb değeri 1 olarak seçilebilir. Elde edilen yüksek kontrastlı görüntüde küçük detaylar kaybolur, ancak bu, ray ve travers bileşenlerinin daha kolay algılanmasını sağlar. Homojenliği düşük bir görüntü elde etmek için yüksek varyanslı monokrom bir görüntü oluşturulmalıdır. İlk olarak, görüntünün piksel değerleri denklem (3.32)'ye göre 0 ile 1 arasında olacak şekilde normalize edilir.

$$J(x,y) = \frac{I(x,y) - I_{min}}{I_{max} - I_{min}} \tag{3.32}$$

Denklem (3.32), Imax ve Imin, sırasıyla monokrom görüntünün maksimum ve minimum değerlerini temsil eder. Görüntünün kontrast değerini artırmak için homojenliğinin düşük olması gerekir. Bu nedenle görüntünün standart sapmasını maksimize edecek kr ve kg değerlerinin bulunması gerekmektedir. Bu amaç fonksiyonu denklem (3.33)'te verilmiştir.

$$(\hat{k}_r, \hat{k}_g) = \arg\max_{k_r,k_g} (\sigma_J^2) \tag{3.33}$$

Nelder-Mead Simplex yöntemi, denklem (3.33)'de verilen standart sapmayı maksimize etmek için kullanılabilir. Amacımız için, $-\sigma_J^2$ 'yi minimize eden kr ve kg değerleri bulunmalıdır. Nelder-Mead yöntemi, doğrusal olmayan sınırsız bir optimizasyon yöntemidir ve Matlab'da fminsearch işlevi aracılığıyla uygulanır. fminsearch işlevi için kr ve kb'nin başlangıç değerleri 1 olarak alınır. Yüksek kontrastlı görüntü 0 ile 1 arasındaki değerlerden oluşur. Bu görüntünün tersini elde etmek için tüm piksel değerlerini 1 değerinden çıkarmak gerekir. Bu işlem sonucunda kontrast iyileştirilmiş ve aydınlatma sorunları azaltılmış bir görüntü elde edilerek bir sonraki adımda segmentasyon için ray bileşenlerini daha kolay tespit edebilmemiz sağlanmıştır.

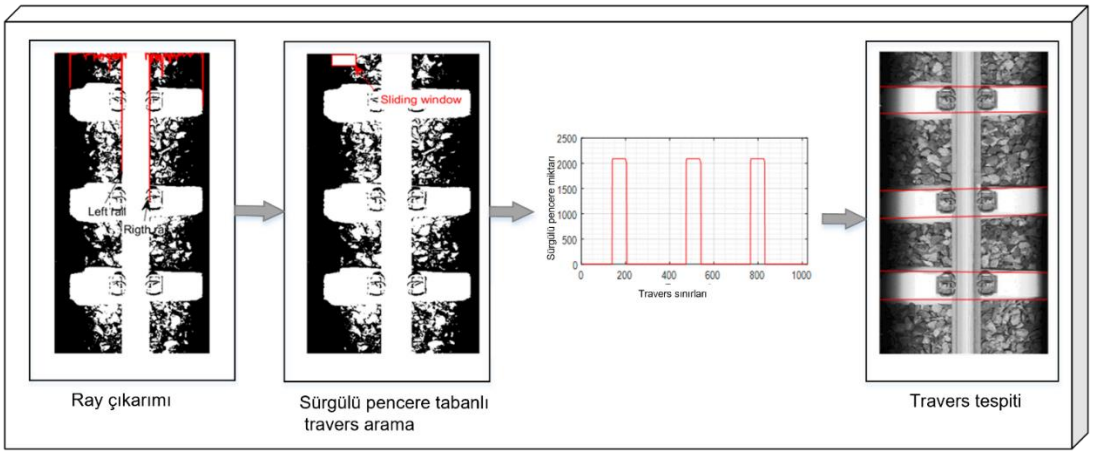
### 3.8.2. İlgi Alanı Çıkarma ve Bağlantı Elemanı Algılama

Kontrastlı görüntü elde edildikten sonra ray ve travers bileşenlerinin algılanması gerekir. Bu amaçla elde edilen görüntü Otsu yöntemi ile bir eşik noktası belirlenerek ikili görüntüye dönüştürülür. Daha sonra yatay iz düşüm yöntemi kullanılarak ray konumu belirlenir. Sol ve sağ rayı algılamak için Şekil 3.61'de gösterilen bir ray çıkarma algoritması uygulanır.

```
Function [lefboundary, rightboundary] = railextraction(S)
[w, h]←size(S) // w: genişlik, h: yükseklik
P←zeros(1,h)
For i←1 to h do
  For j←1 to w do
    P(i)←P(i)+S(i, j)
  End for
End for
DVP←zeros(1,h-1)
For I←2 to h do
  DVP(i-1)←P(i)-P(i-1)
Endfor
[leftboundary, rightboundary]← DVP'nin ilk iki maksimum
değeri
```

Şekil 3. 61. Ray çıkarma algoritması

Şekil 3.61'de S(i, j), parçalı görüntünün (x, y) konumundaki piksel değerini temsil eder. Parçalı görüntü, sıfırlardan veya birlerden oluşan ikili bir görüntüdür. Algoritma ilk olarak vektör P'deki ikili ray görüntüsündeki her bir sütunun ortalama yoğunluğunun histogramını hesaplar. Daha sonra, bu vektörün histogramlarındaki ardışık sütunlar arasındaki fark hesaplanır ve DVP vektöründe saklanır. Görselde sadece bir ray olduğu için rayın başlangıç kolonu ile bir önceki kolon ve rayın bitiş kolonu ile bir sonraki kolon arasındaki histogram farkları diğer farklardan daha büyük olacaktır. Bu nedenle, bu iki maksimum değer, rayın başlangıç ve bitiş noktalarını gösterir. Travers bileşeninin tespiti, ray bileşeninden biraz farklıdır. Bunun nedeni, görüntüde birden fazla travers olmasıdır. Her traversin başlangıç ve bitiş noktası, bağlantı elemanının bu bölgede olabileceği ilgili bölgeyi (ROI) ifade eder. Her travers için başlangıç ve bitiş konumları, rayın sol tarafına yakın bir noktadan yukarıdan aşağıya kayan bir pencere kullanılarak belirlenir. Ray ve travers yerleşimlerinin çizimleri Şekil 3.62'de verilmiştir.



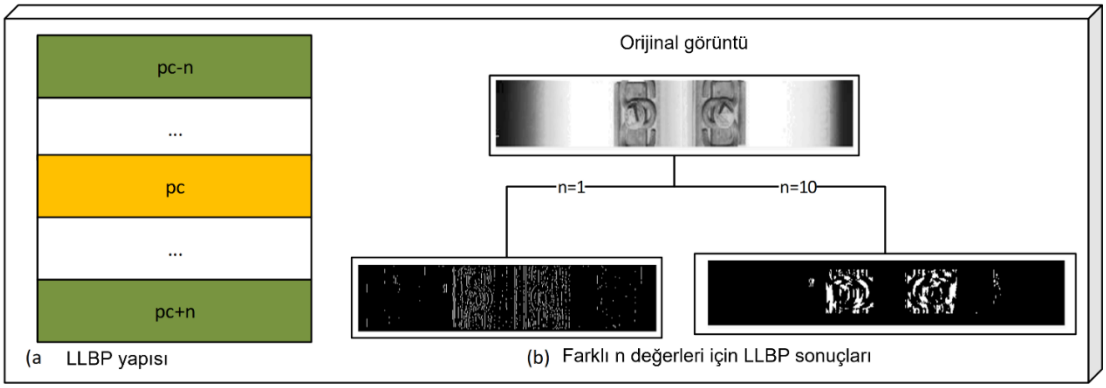

Şekil 3. 62. Ray ve travers algılamasının gösterimi

Şekil 3.62'de görüldüğü gibi ray ve travers bileşenleri belirlendikten sonra görseldeki bağlantı elemanlarının yerleri belirlenmelidir. Bu amaçla LLBP yöntemi kullanılmıştır. Orijinal yerel ikili model yöntemi, merkezi piksel ve komşu pikselleri karşılaştırarak bir değer üretir. Merkezi

piksel değeri pc ve komşu piksel değeri pn ise, pn>pc ise karşılık gelen piksel 0 değerini, değilse 1 değerini alır. Ardından, merkez pikselin etrafındaki her bitin ağırlıklı toplamı, merkez pikselin değerini verir. Yerel ikili desenle elde edilen görüntüdeki bağlantı elemanını tespit etmek zordur. Bu nedenle bu çalışmada LLBP yöntemi kullanılmıştır. LLBP yönteminin farkı, çizgi yönünde karşılaştırma yapmasıdır. Fan ve ark. bağlantı elemanı tespiti için LLBP yöntemini kullandı. Ancak LLBP yöntemi görüntünün tamamına uygulanmakta ve görüntü balast ve ray gibi bileşenleri içerdiğinden bağlantı elemanlarını doğrudan tespit etmek zordur. Bu nedenle, bağlantı elemanı bileşenini aramak için ayrı bir sürgülü pencere filtresi gereklidir. Bu filtrenin boyutunun optimize edilmesi gerekiyor. Bizim yöntemimizde LLBP sadece traversin bulunduğu bölgeye uygulanır ve bağlantı elemanlarının tespiti için herhangi bir arama işlemi gerektirmez. Bu yöntemin matematiksel hesabı denklem (3.34)'te verilmiştir.

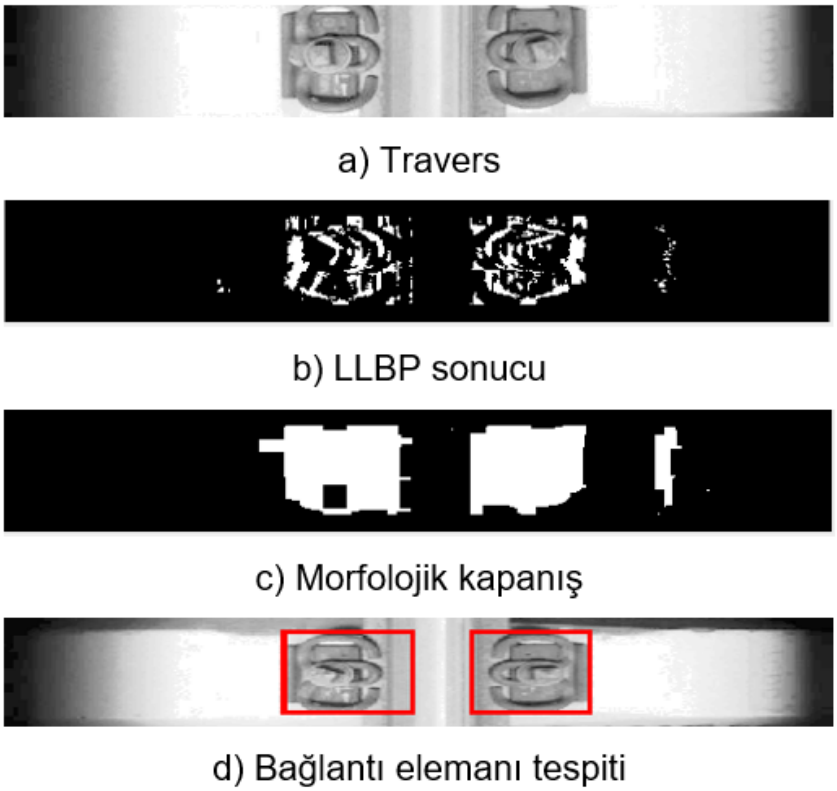
$$LLBP = sign(p_{c-n} - p_c) x sign(p_{c+n} - p_c) \tag{3.34}$$

Denklem (3.34), $p_c$ve $p_{c\pm n}$, merkezi pikseli ve iki komşusunu gösterir. LLBP, komşuluk boyutunu temsil eden yalnızca bir parametreye sahiptir, n. Bu parametre çok küçük seçilirse, gürültülü verilerden dolayı bağlantı elemanının konumunu belirlemek zorlaşır. N'nin değeri ne kadar büyük olursa, bağlantı elemanı o kadar net görünür. Uygun bir değerin seçilmesi, görüntüdeki tutturucunun boyutuna bağlıdır.




Şekil 3. 63. LLBP yapısı ve farklı n değerleri için uygulaması

Şekil 3.63, LLBP yöntemini ve farklı n değerleri için uygulamasını göstermektedir. N değeri küçük olduğunda çok gürültülü bir görüntü elde edilir ve gürültülü piksel değerleri bağlantı elemanının tespit edilmesini zorlaştırır. Öte yandan, yüksek bir n değeri seçildiğinde gürültü azalır ve bağlantı elemanı tespiti kolaylaşır.




Şekil 3. 64. Sırasıyla a'dan d'ye bağlantı elemanı algılama adımları

Travers üzerindeki bağlantı elemanı tespit edildikten sonra görüntü konturu çizilerek kenar noktaları bulunur. Şekil 3.64, travers algılamasından sonra bağlantı elemanlarının LLBP yöntemiyle nasıl algılandığını gösterir. Bölütleme tabanlı teknikler kullanıldığında, gürültülü bölgeler nesne olarak kolayca algılanabilir. Bu, travers taraflarındaki gölgeli alanlarda sıklıkla meydana gelir. Bu çalışmada n'nin en uygun değeri deneme yanılma yoluyla 10 olarak belirlenmiştir.

### 3.8.3. Hafif Evrişimli Sinir Ağı

Evrişimli sinir ağları (CNN'ler), farklı katmanlardan oluşan çok aşamalı çerçevelerdir. Bu çerçevede, her aşama, özellik haritaları olarak bilinen çok kanallı matrislerde ifade edilen tensörlerden oluşur. Görüntü sınıflandırma uygulamalarında, bir CNN'nin giriş katmanı genellikle 2 boyutlu, üç kanallı bir görüntüdür. Bir CNN, üç katmanlı türlerden oluşur. Bu katmanlar genellikle evrişim, havuzlama ve tam bağlantılı katmanlardır. Evrişim katmanı, bir CNN mimarisinin en temel öğesidir ve özellik çıkarma ve doğrusal olmayan işleme için kullanılır. Evrişimden kaynaklanan negatif değerler, ağırlıklı toplamın elde edilen görüntüsüne doğrusal olmayan bir işlem uygulanarak elimine edilir. Doğrusal olmayan operatörler, sigmoid, hiperbolik tanjant (tanh) ve doğrultulmuş doğrusal birimler (ReLU) gibi farklı şekillerde uygulanabilir. Bu durumda, ReLU işleminden sonra bir havuz katmanı uygulanır ve görüntü

boyutunu aşağı örneklemek için kullanılır. Bu amaç için en yaygın olarak kullanılan maksimum havuzlama algoritması, özellik eşlemeli bir görüntünün her bir alt bölgenin maksimum değeri alınarak alt örneklenmesidir.

Tam bağlantılı katman, ileri beslemeli sinir ağlarına benzer ve bir CNN'nin uçlarına doğru birkaç tam bağlantılı katman yerleştirilir. Tam bağlantılı bir katman, önceki katmanlardan bilgi toplayarak yüksek düzeyde bir soyutlama sağlar. Burada çok fazla parametre olduğu için aşırı uyum sorununa yol açar. Bu nedenle bazı bağlantıları kesmek için bırakma uygulanır. Tam bağlantılı katmandan alınan çıktılardan sınıf tahmin puanları üretmek için softmax işlemi uygulanır. Bu işlem denklem (3.35)'te verilmiştir.

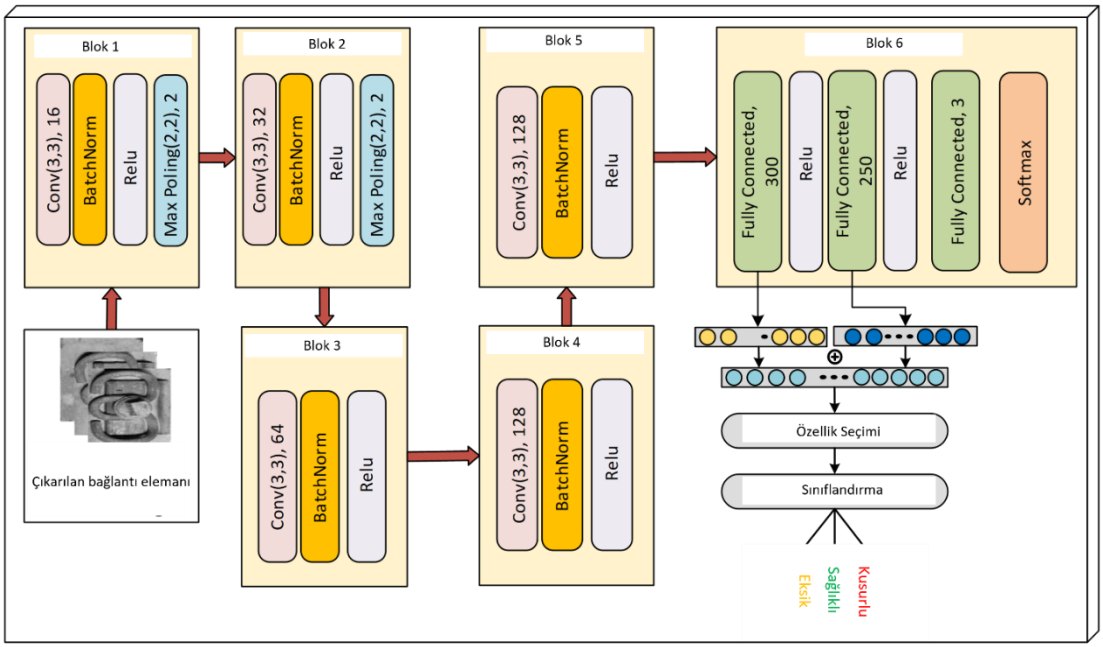
$$\sigma_{softmax}\left(z^{(i)}\right) = \frac{e^{z^{(i)}}}{\sum_{k=1}^{c} e^{z_k^{(i)}}} \tag{3.35}$$

Denklem (3.35)'de her $z^{(i)}$ girişi için $\sigma_{softmax}$ hesaplanır. CNN'nin eğitim süresini azaltmak için bir toplu normalleştirme katmanı kullanılır. Bu katman, ağ başlatma hassasiyetini yönetmek için de gereklidir. Normalizasyon süreci, denklem (3.36)'da tanımlandığı gibi takip edilir.

$$x_{normalized} = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}} \tag{3.36}$$

Mini-batch ortalaması ve varyansı sırasıyla μ ve σ ile gösterilir. $\varepsilon$ değeri, sıfıra bölmeyi önlemek için seçilen küçük bir eşik değeridir.

Bu aşamada, ROI'si çıkarılabilen detaylı bağlantı elemanları sınıfının tanınması amaçlanmaktadır. Yüksek sınıflandırma doğruluğu ve azaltılmış hesaplama süresi elde etmek için derin bir sinir ağı modeli önerilmiştir. Küçük boyutlu veri kümelerini eğitmek için genellikle önceden eğitilmiş ağlara bir aktarım öğrenme algoritması uygulanır, ancak bu ağların birçok katmanı ve karmaşık yapıları vardır. Bu nedenle, bu çalışmada daha küçük boyutlu bir ağ modeli önerilmiştir. Giriş katmanından alınan görüntülere evrişim, toplu normalleştirme, ReLU ve maksimum havuzlama işlemleri uygulanır. Bu işlem beş kez tekrarlandıktan sonra ağın sonunda 3 tam bağlantılı katman kullanılır. Tam bağlantılı iki katmandan elde edilen öznitelikler birleştirilir ve girdi olarak makine öğrenmesi tabanlı sınıflandırıcılara verilir. Önerilen yöntemin blok diyagramı Şekil 3.65'te verilmiştir.




Şekil 3. 65. LCNN mimarisi ve sınıflandırması

Şekil 3.65'te gösterilen evrişimli sinir ağı 26 katmana sahiptir. Öznitelik sayısı ilk tam bağlı katmanda 300, sonraki tam bağlı katmanda 250 olarak belirlenmiştir. Bu özellik numaraları deneylerle bulunmuştur. Son tam bağlı katmandaki öznitelik sayısı, sınıf sayısını gösteren 3'tür. İlk iki katmanda elde edilen özellikler birleştirilir. Daha sonra sınıflama modeline verilerek bağlantı elemanı durumu belirlenir. Sınıflandırma amacıyla farklı modeller test edilir ve en iyi sınıflandırıcı seçilir. Önerilen CNN tabanlı hata tespit sisteminin performansı, bir ölçüm dizisinden alınan bağlantı elemanı görüntüleri kullanılarak değerlendirilir. Sınıflandırıcının performansı, aşağıdaki dört standart ölçü kullanılarak hesaplanır.

$$Accuracy(Acc) = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.37}$$

$$Recall(R) = \frac{TP}{TP+FN} \tag{3.38}$$

$$Precision(P) = \frac{TP}{TP+FP} \tag{3.39}$$

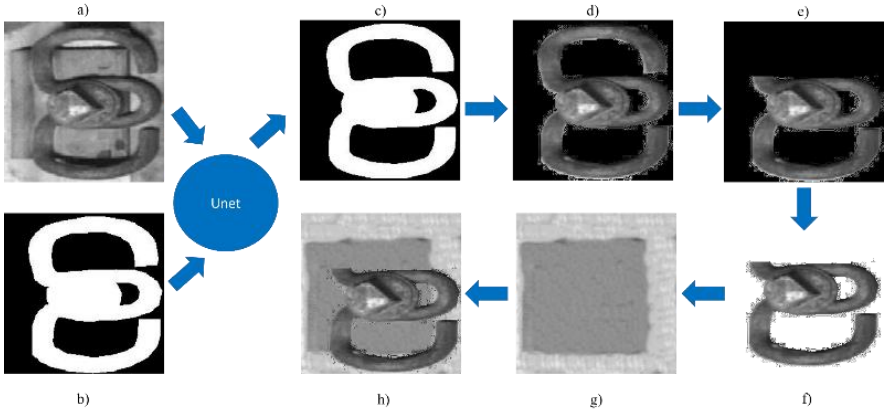
$$F1 = 2 * \frac{PxR}{P+R} \tag{3.40}$$

TP'nin doğru pozitif, TN'nin gerçek negatif, FP yanlış pozitif ve FN'nin yanlış negatif olduğu durumlarda. Sınıflandırıcının performansını belirlemek için tek bir metrik yeterli değildir.

Doğruluk, tüm sınıflardaki girdi verilerinin performansını ölçmek için kullanılır. Diğer metrikler her sınıfa özeldir ve ilgili sınıfın sınıflandırma algoritmasında geri çağırma oranını hesaplar.

### 3.9. Derin Öğrenme Yöntemleri ile Demiryolu Bağlantı Elemanlarının Sınıflandırılması

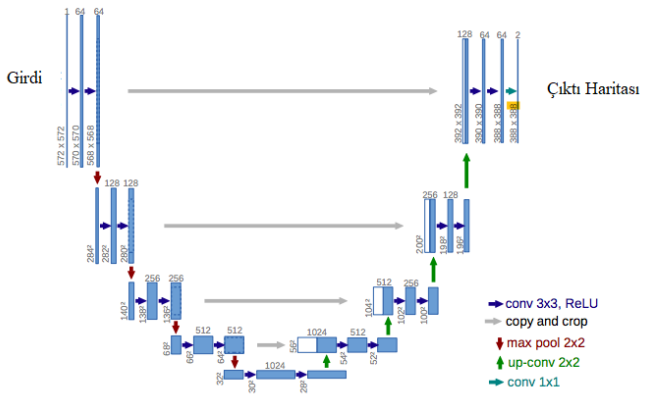
Bu çalışmada, dengesiz bir veri seti üzerinde ray bağlantı elemanlarının ne tür bir kusur içerdiklerini tespit etmek için yeni bir yöntem önerilmiştir. Önerilen yöntem iki aşamadan oluşmaktadır. Birinci aşamada, kusurlu veri oluşturma yöntemi ile arızalı ve eksik bağlantı elemanı örnekleri üretilmiştir. İkinci kısımda ise önerilen CNN, VGG-16 ve ResNet50 modelleri ile kusurlu bağlantı elemanlarını sınıflandırmak amaçlanmıştır. Önerilen kusurlu veri üretim diyagramı Şekil 3.66'da gösterilmektedir. Dış mekân sahnelerinde çekilen görüntüler, zayıf aydınlatma koşulları nedeniyle büyük ölçüde bozuk olabilmektedir. Bu görüntüler, bilgisayarlı görme algoritmalarının genel performansını etkileyen yüksek gürültü seviyelerine sahip olabilirler. Düşük aydınlatma koşullarında bilgisayarlı görme algoritmalarını sağlam hale getirmek için, bir görüntünün görünürlüğünü iyileştirmek gerekmektedir. Bu sebepten dolayı düşük ışıkta görüntü geliştirme yöntemi kullanılmıştır. Düşük ışıklı görüntüleri geliştirmek için pus giderme teknikleri kullanılmıştır. Bu sayede veri setindeki düşük aydınlatma koşullarında alınan sağlıklı demiryolu bağlantı elamanları sınıflandırma için uygun hale getirilmiştir. Demiryolu veri setlerinde, kusurlu bağlantı elemanlarının sayısı sağlıklı bağlantı elemanlarına göre oldukça azdır ve kusurlu veriye erişmek oldukça zordur. Bu nedenle sağlıklı ve kusurlu bağlantı elemanlarının sayısını dengelemek için Unet tabanlı bir model önerilmiştir. GrabCut algoritması, Unet modelinin ürettiği görüntüyü ve orijinal sağlıklı bağlantı elemanına ait görüntüyü kullanarak, görüntüdeki bağlantı elemanının ön plana çıkarılmasını sağlamıştır. GrabCut, bölütlemeyi iyileştirmek ve görüntüdeki nesneyi ön plana çıkarmak için yinelemeli olarak grafik kesimleri için uygulanmaktadır. Bu sayede sağlıklı ve kusurlu bağlantı elamanları sayısı eşitlenmiş ve daha verimli bir sınıflandırma modeli oluşturulmuştur.




Şekil 3. 66. Kusurlu bağlantı elamanı üretimi a) Sağlıklı bağlantı elemanı b) Sağlıklı referans veri c) Unet bölütleme sonucu d) GrabCut ile ön plana çıkarılmış bağlantı elemanı e) Yeni üretilen kusurlu bağlantı elamanı f) Şeffaflaştırılmış kusurlu bağlantı elamanı

Unet, otomatik bir görüntü bölütleme yöntemidir. Unet, Ronneberger ve arkadaşları tarafından biyomedikal resim bölütleme için tasarlanmıştır. Unet, farklı mimari ve evrişimli sinir ağı katmanlarından oluşan piksel tabanlı görüntü bölütleme yöntemidir. Klasik modellere göre daha başarılı sonuçlar vermektedir. Az sayıda görüntü içeren veri setlerinde de başarılı sonuçlar vermiştir. Unet, adını U harfine benzeyen mimarisinden almaktadır ve Şekil 3.67'de görülmektedir. Unet kodlama ve kod çözme kısımlarından oluşmaktadır. Kodlama kısmında görüntündeki içerik yakalanmaktadır. Maksimum havuzlama katmanlarından oluşmaktadır. Kod çözme kısmı ise, devredilen konvolüsyonları kullanarak hassas lokalizasyonu sağlamak için kullanılan simetrik genişleme yoludur (dekoder olarak da adlandırılır).

Unet modeli sayesinde dengelenen veri seti sınıflandırma için uygun hale getirilmiştir. Veri setindeki bağlantı elemanları sağlıklı ve kusurlu (arızalı ve eksik) olarak sınıflandırılmıştır. Sınıflandırma kısmında VGG16, ResNet50 ve tarafımızca oluşturulan CNN modelleri kullanılmıştır.




Şekil 3. 67. Unet mimarisi

Çalışmada kullanılan VGG-16 24 katmandan oluşmaktadır. VGG-16 ImageNet Challenge 2014'te başarılar kazanan bir derin öğrenme modelidir. Küçük filtreler (3x3) VGG-16'da evrişim katmanlarında kullanılmaktadır. VGG16, 13 evrişim katmanı ve 3 tam katmandan oluşmaktadır. 4x4 boyutunda 5 havuz katmanı bulunmaktadır. Son katman softmax katmanından oluşmaktadır. Softmax katmanıyla gelen giriş verileri sınıflandırılmıştır. Relu, aktivasyon fonksiyonu olarak kullanılmıştır.

ResNet mimarisi, 2015 yılında He Kaiming tarafından önerilmiştir. Derin öğrenme mimarilerinde katman sayısı arttığında bir noktaya kadar performans artarken bir noktadan sonra hızlıca düşüş yaşanmaktadır. ResNet mimarisi derinliği artırırken modelin performansını da korumuştur.

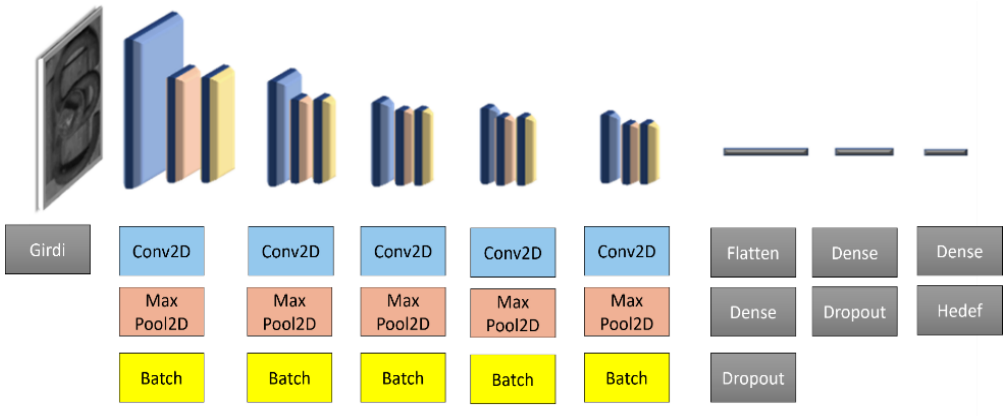
Ayrıca bu modelde hesaplama karmaşıklığında önemli bir faktör olan parametre sayısı diğer ResNet modellerine göre azaltılmıştır. ResNet50 modeli 180 katmandan oluşmaktadır. Aktivasyon fonksiyonu olarak Softmax ve Relu kullanılmıştır.

Evrişimli sinir ağı (CNN), bilgisayarlı görme için kullanılan derin öğrenme ağlarından bir tanesidir. CNN algoritması, hayvanların görsel merkezinden esinlenilerek oluşturulmuştur. CNN'ler, görüntüleri girdi olarak almak için tasarlanmış yapılardır ve bilgisayarlı görmede etkin bir şekilde kullanılmaktadır. CNN, bir veya daha fazla evrişimli katmandan oluşmaktadır. Bunlar; giriş katmanı, konvolüsyonel katmanı (convolution), yığın normalleştirme katmanı (batch normalization), düzleştirme katmanı (flatten), havuzlama katmanı (pooling), tam bağlantılı katman (fully connected, dense) ve düğüm seyreltme katmanıdır (dropout). CNN giriş verilerini aldıktan sonra katman katman işlemler yaparak eğitim sürecini gerçekleştirilir. Sonunda doğru sonuç ile karşılaştırma yapmak için bir hedef çıktısı verir. Üretilen çıktı ile gerçek veri arasında bir hata farkı oluşur. Bu hatayı azaltmak için geriye yayılım algoritması kullanılır. Her bir iterasyonla ağırlıkların güncellenmesi yapılarak hatanın azaltılması sağlanır ve sınıflandırma işleminde başarım sağlanmış olur. VGG-16 ve ResNet50'de olduğu gibi Softmax ve Relu aktivasyon fonksiyonu kullanılmıştır. Tablo 3.6'da görüldüğü gibi CNN modeli ResNet50 ve VGG-16 modeline göre daha az parametre ve katman içermektedir. CNN modelinde geleneksel katman yapısına ek olarak yığın normalleştirme ve düğüm seyreltme katmanları eklenmiştir. Düğüm seyreltme (dropout), modelin aşırı öğrenmesini engellemektedir. Tam bağlı katmandaki düğümler arasındaki bağları koparak ağdaki zayıf bilgilerin unutulmasını sağlamıştır. Böylelikle eğitim performansı artmıştır. Yığın normalleştirme ise konvolüsyonel katmanları arasında kullanılmıştır. Modeli daha düzenli hale getirerek eğitim süresini kısaltmıştır. Eğitim süresi performans kaybı yaşamadan %39 azalmıştır. Şekil 3.68'de önerilen CNN mimarisi görülmektedir.

Tablo 3. 6. Önerilen modellerinin katman ve parametre sayıları

| Ağ | Yıl | Katman Sayısı | Parametre Sayısı |
|---|---|---|---|
| **VGG-16** | 2014 | 24 | 14M |
| **ResNet50** | 2015 | 180 | 24M |
| **Oluşturulan CNN** | - | 21 | 0.6M |

Tablo 3.7'de kullanılan derin öğrenme tabanlı modellerin eğitim parametreleri verilmiştir. Tablo 3.7'de görüldüğü gibi seyreltme değeri önce 0.3, sonra 0.5 olarak seçilmiştir. Modellerin ezberlemesini önlemek için seyreltme değeri arttırılmıştır. Arttırma işleminden sonra model performansında bir düşüş gözlemlenmemiştir.




Şekil 3. 68. CNN mimarisi

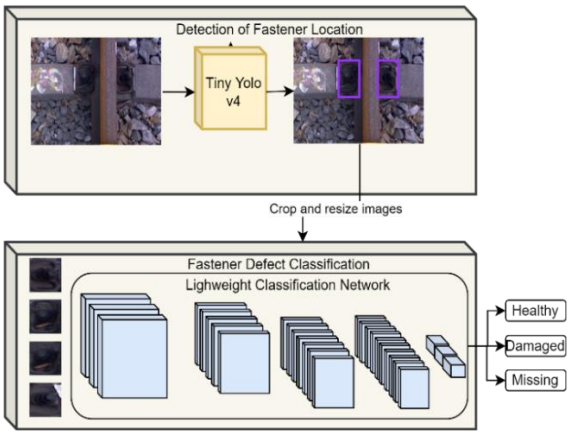
Tablo 3. 7. Eğitim parametreleri

| Ağ | Döngü Sayısı | Mini Batch Boyutu | Öğrenme Oranı | Seyreltme Değeri |
|---|---|---|---|---|
| **VGG-16** | 20 | 8 | 1e-3 | 0.3, 0.5 |
| **ResNet50** | 20 | 8 | 1e-3 | 0.3, 0.5 |
| **Oluşturulan CNN** | 20 | 8 | 1e-3 | 0.3, 0.5 |

## 3.10. Gerçek Zamanlı Bağlantı Elemanı Kusurlarının Tespiti için Derin Öğrenme Tabanlı Hibrit Yaklaşım

Bu çalışmada hibrit bir yaklaşım önerilmiştir. Önerilen hibrit yaklaşım, bağlantı elemanı hata tespitini iki aşamada gerçekleştirir. İlk aşama, bağlantı elemanının konumunun belirlenmesi,

ikinci aşama ise bağlantı elemanı hata tespitidir. Önerilen yaklaşımın blok diyagramı Şekil 3.69'da verilmiştir.

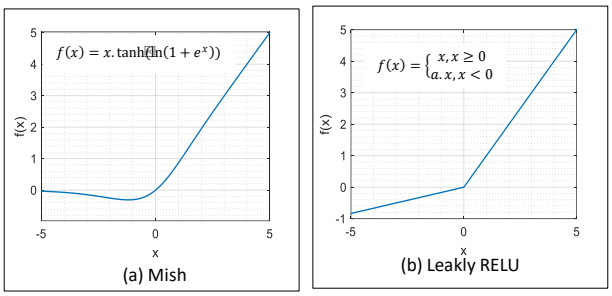



Şekil 3. 69. Bağlantı elemanının kusur tespiti için önerilen hibrit yöntem

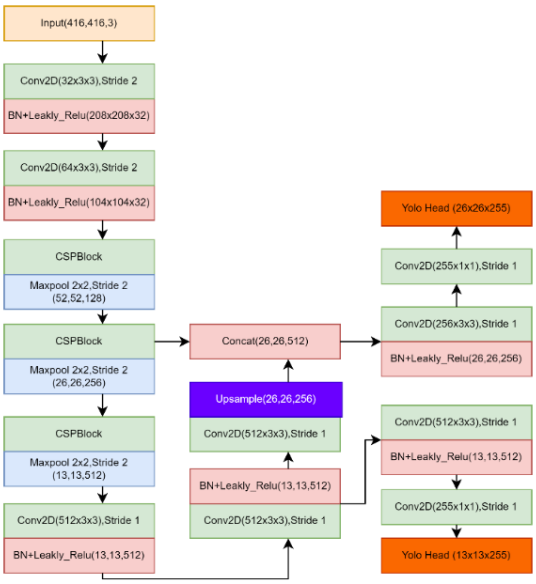
Şekil 3.69'da önerilen yöntemin ilk adımında bağlantı elemanlarının konumları belirlenir. Demiryolu görüntüsündeki ray, travers ve balast gibi bileşenlerin karmaşık arka planı nedeniyle bağlantı elemanının tespiti zordur. Bu nedenle Tiny YOLOv4, bağlantı elemanının gerçek zamanlı ve yüksek doğrulukta tespiti için kullanılacaktır. Bağlantı elemanı durum tanıma için düşük ağırlıklı bir evrişimsel sinir ağı önerilmiştir. Inverted rezidü blokları ve dikkat modülü sayesinde bu model hem boyut olarak daha küçük hem de daha hızlı tanıma gerçekleştirebilmektedir.

### 3.10.1. Bağlantı Elemanı Konumunun Tespiti

Bağlantı elemanının konumunu belirlemek için YOLOv4-Tiny modeli kullanılmıştır. Bağlantı elemanı konumlandırma yöntemi, karmaşık arka plan varlığında hem hız hem de doğruluk açısından iyi sonuçlar vermelidir. YOLOv4-Tiny, YOLOv4 modeli üzerine tasarlanmış bir modeldir ve nesne tespiti için daha hızlı çalışır. Bu özelliği sayesinde özellikle gömülü sistemlerde ve mobil cihazlarda gerçek zamanlı olarak kullanılabilir. YOLOv4-Tiny, CSPDarknet53 modülünü bir omurga olarak modifiye etmiştir. Bu modül, Rezidüel ağda ReSBlock yerine CSPBlock kullanır. CSPBlock'un hesaplama maliyetini azaltmak için Darboğaz blokları kullanılmayarak bu maliyet düşürülür ve ağın doğruluğu ve öğrenme yeteneği arttırılır. Standart YOLOv4 ile karşılaştırıldığında YOLOv4-Tiny arasındaki diğer bir fark, CSPDarknet53'teki Mish etkinleştirme işlevi yerine Leaky Relu aktivasyon fonksiyonu kullanmasıdır. Her iki aktivasyon fonksiyonu da Şekil 3.70'te verilmiştir.

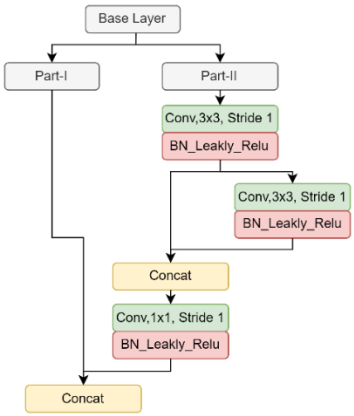



Şekil 3. 70. Bağlantı elemanının kusur tespiti için önerilen hibrit yöntem




Şekil 3. 71. YOLOv4-Tiny mimarisi

Şekil 3.71'de gösterilen mimari, farklı ölçeklerde özellik haritaları elde etmek için bir piramit ağ yapısı kullanır. Bu yapı algılama hızını arttırır ve YOLOv4'te kullanılan uzaysal piramitten farklıdır. Ayrıca tahmin için 26x26 ve 13x13 olmak üzere 2 farklı ölçekte özellik haritaları kullanır. YOLOv4-Tiny modelinde kullanılan CSP bloğu Şekil 3.72'de verilmiştir.

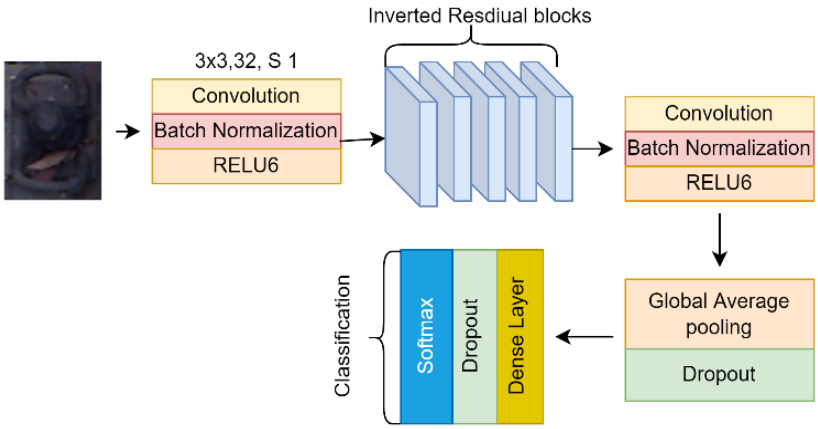



Şekil 3. 72. YOLOv4-Tiny'de kullanılan CSP bloğu

Şekil 3.72'deki CSP bloğu, taban katmanını iki parçaya böler. İkinci parçaya belirli evrişim işlemleri uygulandıktan sonra birinci parça ile birleştirilir. Yapının tabanında artık bloklar bulunmaktadır. Bu bloklar algoritmanın hızını arttırır.

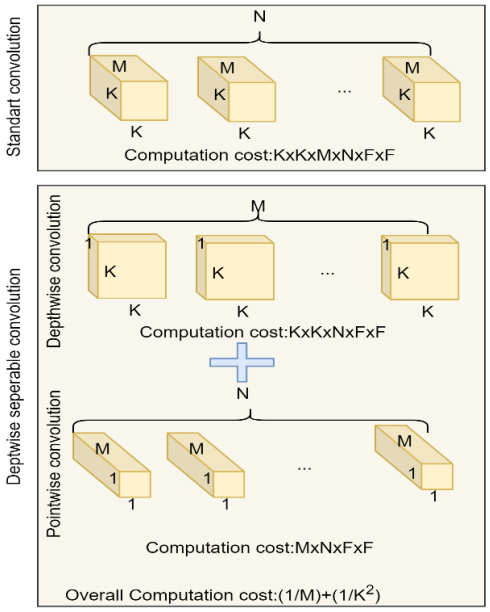
### 3.10.2. Bağlantı Elemanının Durumunu Tanıma

YOLOv4-Tiny modeli ile bağlantı elemanı tespit edildikten sonra, bağlantı elemanındaki kusur tipinin tespiti için hafif bir ağ modeli önerilmiştir. Gerçek zamanlı olarak çalışacak yüksek sınıflandırma doğruluğuna sahip bir ağ oluşturmak için ters çevrilmiş artık blokları kullanan düşük ağırlıklı bir ağ modeli önerilmiştir. Önerilen ağ modeli Şekil 3.73'te verilmiştir.



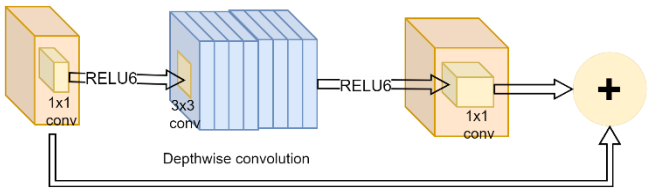

Şekil 3. 73. Önerilen düşük ağırlıklı CNN modeli

Şekil 3.73'te önerilen yöntem, hafif bir ağ modelinden oluşmaktadır. Önerilen model, derinliğe dayalı evrişim katmanlarını kullanarak artık blokları tersine çevirmiştir. Derinliksel evrişim katmanları ilk olarak Xception modelinde tanıltımış ve parametre sayısını azaltmak için hafif ağlarda kullanılmıştır [79]. Derinlemesine ayrılabilir evrişim işlemi, her kanal için 3x3 tek kanallı evrişim işlemini, ardından 1x1 M kanallı noktasal evrişim işlemini gerçekleştirir. Yapı Şekil 3.74'te görülmektedir.




Şekil 3. 74. Standart ve derinlik açısından ayrılabilir konvolüsyonlar

Şekil 3.74'te, FxF boyutlu bir görüntü üzerinde derinlik bazında ayrılabilir evrişim kullanıldığında, hesaplama maliyeti standart bir evrişim katmanı kullanmaktan 10 kat daha küçüktür. Birçok evrişimli sinir ağında, her evrişim işleminden sonra RELU aktivasyonu uygulanarak ağın gücü artırılır. Bu işlem karmaşık ağlarda iyi çalışsa da daha düşük boyutlu ağlarda veri kaybına neden olur. Bu nedenle evrişim katmanından sonra verilerin doğrudan diğer katmana aktarılmasını sağlayan lineer darboğaz işlemi, kaçırılan özelliklerin RELU ile kullanılmasına olanak tanır. İlk olarak MobileNet tarafından önerilen hafif ağ modeli, hız ve az sayıda parametre açısından avantajlar sağlasa da doğruluğu düşüktür. Daha sonraları geliştirilen 2. versiyonundan (MobileNetv2) sonra ters çevrilmiş artık bloklar kullanılmıştır [80]. Bu bloklar üç evrişim işleminden oluşur. Bu katmanlar, öznitelik boyutunu artıran darboğaz bloğu, düşük ağırlıklı öznitelik çıkarımı sağlayan derinliğe dayalı evrişim katmanı ve noktasal evrişim katmanından oluşur. Bu yapı Şekil 3.75'te gösterilmektedir.




Şekil 3. 75. Ters artık blok

Şekil 3.75'te verilen model, geleneksel artık bloklardan daha yüksek doğruluk sonuçları sağlar. Geleneksel artık bloklar, boyut küçültme işleminden sonra özellikleri çıkardığı için birçok özellik kaybolur. Tersine çevrilmiş artık bloklar, öznitelik çıkarmadan önce boyutu arttırır ve derinlikli

evrişim katmanları nedeniyle düşük sayıda parametreye sahiptir. Ayrıca, global havuzlama katmanından sonra modele yoğun bir katman dahil edilmiştir. Önerilen modelin yapısı ve parametreleri Tablo 3.8'de verilmiştir.

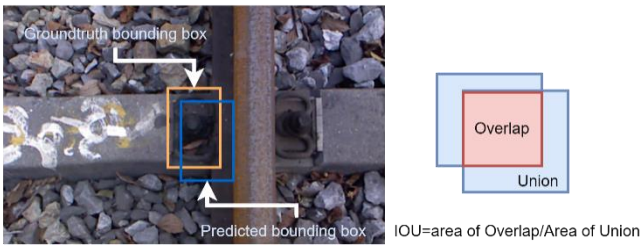
Tablo 3. 8. Önerilen CNN'nin yapısı ve parametreleri

| **Filtre sayısı** | **Katman** | **t** | **n** | **s** |
|---|---|---|---|---|
| 32 | convolution2d 3x3 | - | 1 | 1 |
| 32 | ınverted residual block | 1 | 1 | 1 |
| 8 | ınverted residual block | 6 | 1 | 2 |
| 16 | ınverted residual block | 6 | 2 | 1 |
| 32 | ınverted residual block | 6 | 3 | 2 |
| 64 | ınverted residual block | 6 | 2 | 1 |
| 256 | global average pooling | - | - | - |
| 512 | dense | - | - | - |

Tablo 3.8'deki t değeri, giriş tensörüne uygulanan kanal genişletme faktörüdür. Örneğin t değeri üç alınırsa ve giriş tensörü 16 kanaldan oluşuyorsa çıkış 3x16=48 olur. n değeri, blokların kaç kez tekrarlanacağını gösterir. S değeri, evrişim işlemindeki adım parametresidir.

### 3.10.3. Performans Metrikleri

Bu çalışmada hem nesne tespiti hem de kusur tanıma, bir YOLOv4-Tiny ve hafif bir evrişimli sinir ağı ile yapılmıştır; işlemler iki farklı metrik üzerinde gerçekleştirilmiştir. Nesne algılama için en önemli metriklerden biri, etiketli sınırlayıcı kutu ile öngörülen sınırlayıcı kutunun kesişimlerinin birleşime oranı olarak temsil edilen Birleşim Üzerinden Kesişme'dir (IoU). Bu yapı Şekil 3.76'da verilmiştir.




Şekil 3. 76. IoU hesaplaması

Nesne tespitinde, IOU dışında kesinlik, geri çağırma, mAP ve kesinlik-hatırlama eğrisinden elde edilen F1 puanlarına ihtiyaç vardır. Hassasiyet ve geri çağırma, nesne tespitinde iki önemli parametredir ve bu iki parametre ile oluşturulan eğri, diğer metrikleri hesaplamak için kullanılır. Kesinlik, bir modelin yalnızca ilgilenilen nesneleri tahmin etme yeteneğidir. Geri çağırma, bir modelin tüm durumları bulma yeteneğidir. İki metrik için denklemler aşağıda verilmiştir.

$$P = \frac{TP}{TP+FP} \tag{3.41}$$

$$R = \frac{TP}{TP+FN} \tag{3.42}$$

$$F_1 = 2 \times \frac{P \times R}{P+R} \tag{3.43}$$

Denklem (3.41)-(3.43)'de, TP gerçek pozitif değeri temsil eder ve eğitilmiş modelin doğru pozitifi tahmin ettiği anlamına gelir. FP değeri yanlış pozitiftir ve eğitilen model, pozitif sınıfı yanlış bir şekilde tahmin eder. FN değeri, bir örneğin gerçek değeri pozitif olduğunda negatif olarak tahmin edilmesidir. Nesne algılama yöntemlerinin doğruluğu genellikle her sınıf için Ortalama Ortalama Hassasiyet (mAP) ile ölçülür. Ortalama kesinlik, P-R eğrisinin altındaki alandır. Bu iki parametre Denklem (44)-(45)'te verilmiştir. Hafif sınıflandırma ağındaki doğruluk oranı, bağlantı elemanı kusur sınıflandırması için de kullanılır

$$AP = \int_0^1 p(r)dr \tag{3.44}$$

$$mAP = \frac{1}{N}\sum AP_i \tag{3.45}$$

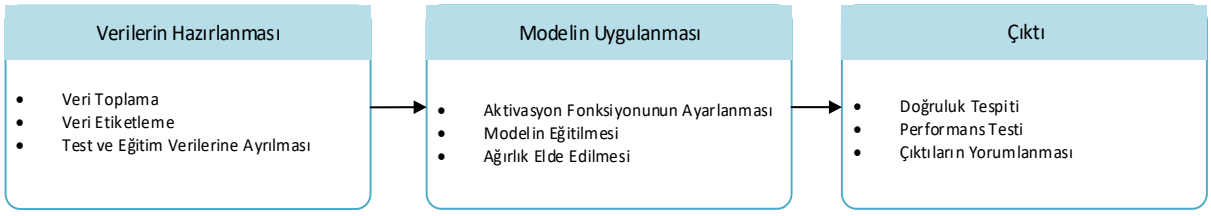
## 3.11. Demiryolu Bağlantı Elemanlarında Bulunan Kusurların YOLOv4 ve Bulanık Mantık Kullanarak Tespiti

Önerilen yöntemde, bağlantı elemanlarının tespiti ve kusur durumunun belirlenebilmesi için derin öğrenme ve bulanık mantık yapısı kullanılarak yeni bir algoritma uygulanmıştır. Bağlantı elemanlarının tespiti için YOLOv4 algoritması kullanılmıştır. İlk olarak, bağlantı elemanı görüntüsü 6 bölgeye ayrılarak her bölge etiketlenmiştir. Bu işlemin amacı, bağlantı elemanının kusur durumunu ayrıntılı inceleyebilmektir, çünkü bağlantı elemanları titreşim ve sıcaklık değişiminin uzun vadeli etkileri ile zarar görebilir. Hasarlı bağlantı elemanı; kısmen aşınıp etkinliğini yitirmiş olabilir, kısmen kırılmış veya tamamen kırılmış olabilir. Şekil 3.77'te bazı kusurlu bağlantı elemanı görüntüleri verilmiştir. Görüleceği üzere bağlantı elemanları farklı bölgelerden kırılmış veya eğilmiş olabilir. Bu nedenle bağlantı elemanının tek parça yerine 6 parçaya bölünerek incelenmesi daha uygun olacaktır. Tespit edilen 6 parçanın güven değeri, 6 girişli tek çıkışlı bulanık mantık yapısında kullanılarak bağlantı elemanının sağlık durumu hakkında çıkış değeri elde edilmiştir.

Şekil 3. 77. Kusurlu bağlantı elemanı örnekleri

### 3.11.1. Bağlantı Elemanlarının Tespiti İçin YOLOv4 Algoritmasının Uygulanması

Model uygulanmadan önce verilerin hazırlanması gerekmektedir. Veri hazırlama işleminden sonra yapılan işlemlerin akış diyagramı Şekil 3.78'deki gibidir.




Şekil 3. 78. Yapılan işlemlerin akış diyagramı

Eğitim işlemi için standart hale getirilmiş ve etiketlenmiş 2100 adet görüntü kullanılmıştır. YOLO algoritmasında eğitim işlemi için kullanılacak görüntülerin kısıtlaması yoktur. Veri setinde az görüntü bulunması aşırı öğrenmeye neden olabilir. Veri seti büyük olsa dahi çeşitlilik yok ise yine aşırı öğrenme durumu ortaya çıkabilir. Bu çalışmada, kullanılan bağlantı elemanı görüntüleri ray üzerinden toplanan görüntülerden kırpılarak elde edilmiştir. Verilerdeki çeşitliliği sağlamak için farklı aydınlatma koşullarında görüntüler kullanılmıştır. Veri seti etiketleme işlemi

için 6 bölge belirlenmiştir. Bölgelerin sınırları seçilirken kırılma veya eğilmenin olabileceği bölgeler dikkate alınmıştır. Bölgelerin sınırları Şekil 3.79.b'de gösterilmiştir. Görüntülerin 6 parçaya bölünmesinin amacı bağlantı noktalarını sınıflandırarak YOLO ağının bağlantı plakasını bir bütün halinde değil parçalar halinde öğrenmesini sağlamaktır. Bağlantı elemanı görüntülerinin parçalara ayrılması bölgesel hataların tespit oranını artırmıştır. Ayrıca bağlantı plakasının bölgesel hatalarının tespiti için bulanık mantık kullanılmasının önünü açmıştır.

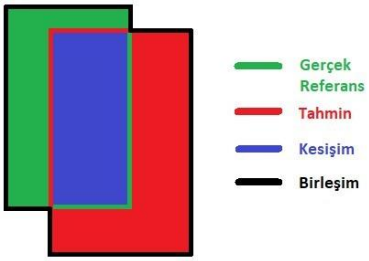

(a) (b)

Şekil 3. 79. (a) Orijinal görüntü (b) Etiketli görüntü

YOLOv4, Darknet framework üzerinde geliştirilmiş bir nesne tespit algoritmasıdır. YOLOv4 ağı, hızlı çalışması nedeniyle birçok uygulamada kullanılır. Derin evrişimli sinir ağlarını kullanarak nesneleri tespit eder ve kutucuklar içerisinde gösterir. Veriler eğitildikten sonra, YOLOv4 ağına bir test görüntüsü verildiğinde çıkış olarak görüntü üzerinde bulunan nesnelerin hangi sınıfa ait olduğunu gösterir. Güven skoru, tahmin edilen nesnenin gerçek nesneye ne kadar benzediğini göstermektedir. Nesnenin güven skoru 0 ve 1 arasında oluşturulmaktadır. Güven skoru denklem (3.46)'deki gibi hesaplanmaktadır.

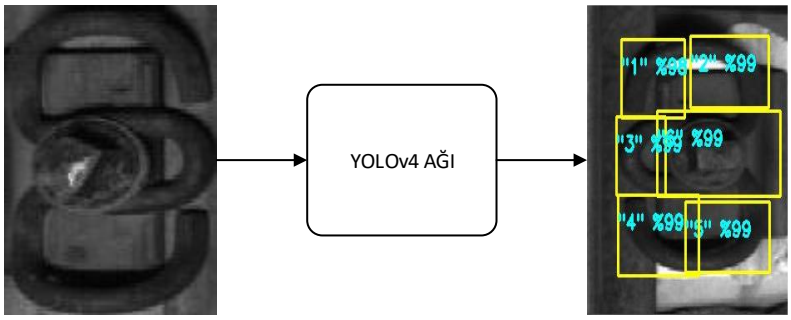
$$Güven\ Skoru = Pr(Nesne) * IoU \tag{3.46}$$

$$IoU = \frac{Kesişim(Gerçek-Referans \cap Tahmin)}{Birleşim(Gerçek-Referans \cup Tahmin)} \tag{3.47}$$



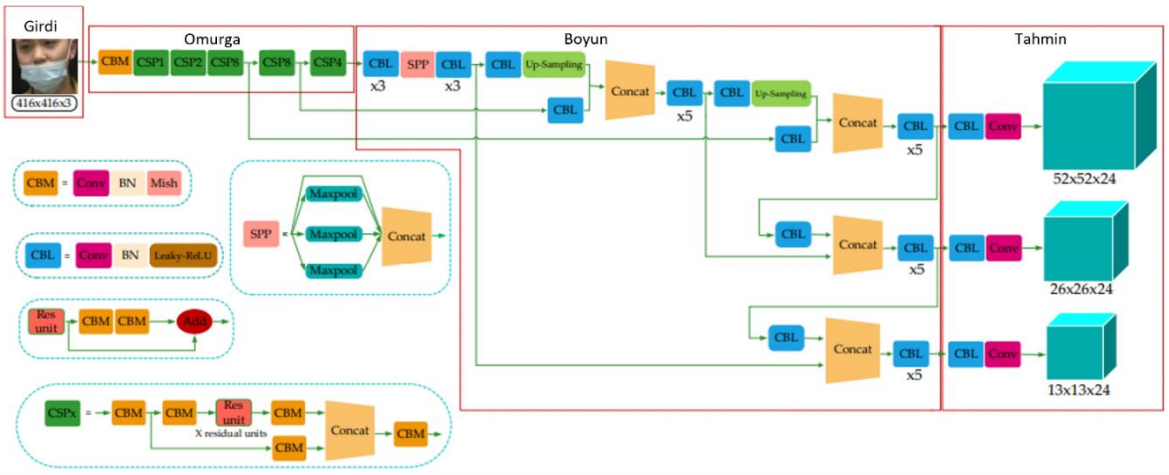

Şekil 3. 80. (a) Orijinal görüntü (b) Etiketli görüntü

Şekil 3.80'deki gösterildiği gibi gerçek referans ve tahmin kutucuklarının kesişimleri IoU (Intersection over Union) olarak ifade edilmiştir. IoU değeri denklem (3.47)'de gösterildiği gibi hesaplanır.




Şekil 3. 81. YOLOv4 ağı giriş çıkış örneği

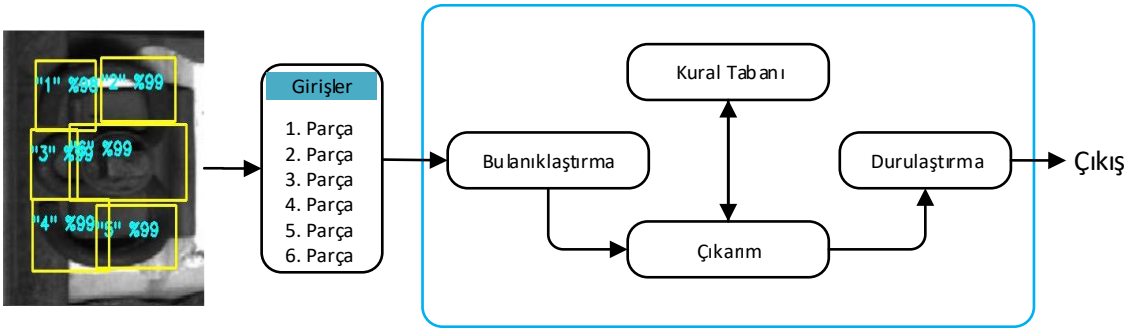
YOLOv4'te SAM, SPP, PANet kullanılır. Kafa bölümü, Dense Prediction katmanıdır. YOLOv3 ile aynı yapı kullanılmıştır. Görevi ise nesnenin koordinatlarını, güven skorunu ve etiketini içeren bir vektör oluşturmaktır. YOLOv4 mimarisi Şekil 3.82'de gösterilmiştir. YOLO, tek aşamalı nesne tespiti (One-Stage Detector) yapmaktadır. Sparse Prediction katmanı; Faster R-CNN, R-CNN gibi algoritmalarda kullanılmaktadır ve bu algoritmalar iki aşamalı olarak nesne tespitini yapar. Kullanılacak veri setini zenginleştirme ve optimizasyon işlemleri için Bag of freebies ve Bag of specials paketleri kullanılmıştır. Bag of freebies, veri zenginleştirme işlemi için kullanılır ve ağın başarımını arttırır. Fakat ağın hızını etkilemez.




Şekil 3. 82. YOLOv4 mimarisi

### 3.11.2. Kusur Tespiti İçin Tasarlanan Bulanık Mantık Yapısı

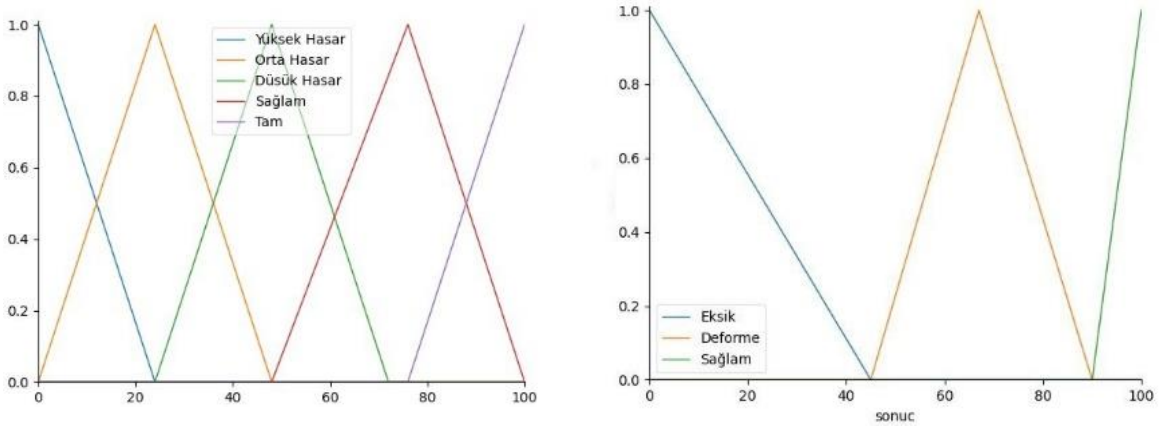
Test görüntüsüne YOLOv4 algoritmasının uygulanması ile 6 farklı güven değeri oluşturulmuştur. Oluşan 6 farklı güven değeri dikkate alınarak bağlantı elemanının durumu hakkında yorum yapmak zordur. Demiryolunda binlerce bağlantı elemanı olduğu düşünülürse hepsinin incelenmesi uzun zaman alacaktır. Bu nedenle, oluşan güven değerlerini kullanarak otomatik denetim sistemi oluşturmak şarttır. Bu tür karmaşık durumların olduğu durumlarda, insan beynine benzer tahminler yapmak için bulanık mantık kullanılabilir. Bu nedenle, güven değerleri kullanılarak kusur tespiti yapmak için bulanık mantık kullanılacaktır. Bulanık mantık terimi, Lotfi Zadeh tarafından 1965 tarihli bulanık küme teorisi önerisiyle tanıtılmıştır. Bulanık Mantık, insanın düşünme yapısına benzeyen bir akıl yürütme yöntemidir. Bulanık mantık yaklaşımında, dijital değerler "0" ve "1" arasındaki tüm ara olasılıkları içeren karar verme şeklini taklit eder. Bağlantı elemanlarının kusur tespiti için düşünülürse, çıktı durumu klasik kümede "hasarlı" ve "sağlam" şeklinde kesin sonuç üretilebilir. Ancak, bulanık kümede "eksik", "orta hasarlı", "az hasarlı", "sağlam" gibi sonuçlar üretilebilir. Bulanık mantık; bulanıklaştırma birimi, çıkarım birimi, kural tabanı ve durulaştırma birimi olmak üzere dört ayrı birimden oluşur. Bulanıklaştırma biriminde, giriş bilgileri dilsel niteleyicilere dönüştürülür. Giriş bilgilerinin ait olduğu üyelik dereceleri tespit edilir. Bu nedenle, bulanık mantık kullanılarak görüntüdeki bağlantı elemanının durumunun değerlendirilmesi için ilk olarak giriş görüntüsü giriş fonksiyonlarına göre işlenir ve her kümenin üyelik dereceleri belirlenir. Daha sonra bulanık çıkarım tasarlanır. Çıkarım işlemi kurallara göre yapılır. Kurallar, "eğer-değilse" koşullarına göre tasarlanmıştır. Son olarak, durulaştırma işlemi uygulanır. Durulaştırma işlemi sonucunda, bulanık değerler insanlar tarafından anlaşılabilir hale getirilir. Kullanılan bulanık sistemin blok diyagramı Şekil 3.83'teki gibidir.




Şekil 3. 83. Bulanık mantık yapısı

YOLOv4 algoritması sonucunda tespit edilen parçaların güven değerlerine göre kusur durumları hakkında çıkış üretilmesi amaçlanmıştır. Bu amaç doğrultusunda giriş ve çıkış değerlerinin isimleri, sayıları ve limitleri belirlenmiştir. Giriş için belirlenen 6 farklı giriş değeri için üyelik fonksiyonları Şekil 3.84.a' da gösterilmiştir. Şekil 3.84.b' ise çıkış için üyelik fonksiyonlarını göstermektedir. Şekil 3.84.b'de görüldüğü gibi %0 ile %45 aralığında sonuç

alınırsa “eksik”, %45 ile %90 aralığında sonuç alınırsa “deforme”, %90 üzerinde sonuç alınırsa “sağlam” olarak nitelendirilecektir. Oluşturulan kurallar ise Tablo 3.9’da verilmiştir. Tabloda YH; yüksek hasar, OH; orta hasar, DH; düşük hasar, S; sağlam, T; tam, E; eksik ve D; deforme olarak kodlanmıştır




(a) (b)

Şekil 3. 84. Giriş ve çıkış için oluşturulan üyelik fonksiyonları (a) Girişler için üyelik fonksiyonları (b) Çıkış için üyelik fonksiyonları

Tablo 3. 9. Kurallar

| **Girişler** | | | | | | **Çıkış** |
|---|---|---|---|---|---|---|
| 1. Parça | 2. Parça | 3. Parça | 4. Parça | 5. Parça | 6. Parça | Sonuç |
| YH | YH | YH | YH | YH | YH | E |
| OH | OH | OH | OH | OH | OH | D |
| DH | DH | DH | DH | DH | DH | D |
| S | S | S | S | S | S | S |
| T | T | T | T | T | T | S |

## 3.12. Demiryolu Genleşme Aralıklarının Görüntü İşleme Teknikleri ile Tespiti ve Ölçümü

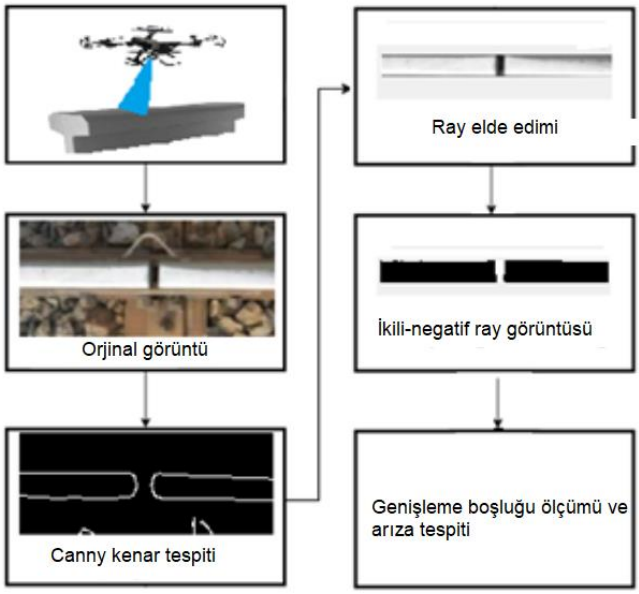
Raylar monte edilirken bölgenin hava sıcaklığı, rayların o bölgede ulaşabileceği maksimum sıcaklık ve ray demirinin uzama katsayısı dikkate alınarak aralarında belirli bir boşluk bırakılır. Bu boşluklar, rayların yürümesi nedeniyle ray uzunluklarında meydana gelen değişiklikler veya raylar üzerindeki genişlemeler sonucunda rayların şeklinde bir değişiklik olmaması için bırakılır. Bu nedenle genleşme boşlukları olarak adlandırılan bu boşlukların montajdan sonra da kontrol edilmesi gerekir. Manuel kontroller uzun zaman alır ve personel emeği gerektirir.

Genleşme boşluklarında belirlenen miktara göre azalma olursa daraltılmış genleşme boşluğu, artarsa genişlemiş genleşme boşluğu olarak adlandırılır. Sıcaklığın etkisiyle genişleyen raylarda bu aralıklarda daralma olursa raylarda bükülme meydana gelir. Genişleme durumunda ray başlarında kırılmalar görülür. Bu durumların her ikisi de trenin raydan çıkarak kazalara yol açmasına neden olabilir. Bu çalışmada önerilen yöntemde, temassız görüntü işleme teknikleri ile raylar ve genleşme boşlukları belirlenmiştir.

Genişleme boşluğu;

- e = (L * t * a*)/100 formülü ile hesaplanır
- e : mm cinsinden boşluk miktarı
- L : m cinsinden ray uzunluğu
- t : Santigrat cinsinden sıcaklık değişimi
- a : ray demirinin genleşme katsayısı.

Her bölgenin kendine has hava sıcaklığı değişimlerine göre raylar arasında genleşme boşlukları bırakılır. İyi bakım yapılmaması, yüksek eğim, fren mıknatısı ve sıcak havalarda rayın yüksek ısısı gibi nedenlerle genleşme boşlukları bozulur. Bu bozulan genleşme boşlukları, rayların bükülmesine ve rayların kırılmasına neden olarak trenin raydan çıkmasına neden olabilir. Bu nedenle raylar döşendikten sonra kontroller yapılmalıdır. Bu çalışmada önerilen yöntemde, demiryolu hattının drone ile izlenmesi, hat üzerindeki ray ve genleşme boşluklarının belirlenmesi ve tespit edilen genişleme boşluklarının piksel sayılarının hesaplanması adımları gerçekleştirilmiştir. Şekil 3.85'te blok diyagramı verilen yöntemin adımları aşağıdaki şekilde gerçekleştirilmiştir.




Şekil 3. 85. Bilgisayarlı görme blok şeması

Demiryolu görüntüleri, hat boyunca sabit bir yükseklikte seyahat eden bir IHA üzerine monte edilmiş bir kamera ile elde edilmiştir. MATLAB'da rayların belirlenmesi için elde edilen demiryolu görüntülerine keskin kenar algılama yöntemi uygulanmıştır.

- Tespit edilen raylar görüntülerden kesilmiştir.
- Çıkarılan ray görüntüsü üzerindeki genişleme boşluklarını daha net ölçmek için öncelikle ikili formata dönüştürülmüştür. Sonra negatif alındı.
- Böylece son aşamada elde edilen görüntülerdeki siyah alanlar rayları, beyaz alan ise genişleme boşluklarını göstermektedir.
- İkili formatta olan bu görüntüler grafiksel olarak analiz edilmiş ve genişleme boşlukları ölçülmüştür.
- Elde edilen çıktılar deneysel sonuçlarda verilmiştir.

### 3.13. Görüntü İşleme ile Demiryolu Hat Açıklığının Belirlenmesi

Çalışmada, görüntü işleme yöntemleri ile demiryolu hat açıklığı kontrolünün yapılmasına olanak sağlayan bir yöntem geliştirilmiştir. Yöntemin adımları aşağıdaki gibi tanımlanmıştır.

- Demiryolu, hat boyunca sabit bir yükseklikte seyreden bir drone tarafından takip edildi.
- Video kaydındaki titreşimler, drone kamera ile kaydedilen hat görüntülerinden daha sağlıklı ölçümler yapabilmek için çerçeve oluşturulmadan önce kaldırıldı.
- Titreşimleri kaldırılan kayıttan çerçeveler oluşturuldu.
- Oluşturulan çerçeveler MATLAB'da ön işleme tabi tutulmuştur.
- Ön işleme aşamasında, demiryolu görüntülerinden ray dışı gürültüyü gidermek için gauss filtrelemesi uygulanmıştır.
- Daha sonra rayları tanımlamak için keskin kenar algılama algoritması uygulandı.
- Rayların belirlenmesinden sonra raylar arasındaki piksel değerleri ölçüldü, kalibre edildi ve mm'ye çevrildi.
- Birçok ülkede standart olarak kullanılan 1435 mm ray açıklığı ile görüntülerden ölçülen ray açıklığı değerleri karşılaştırılmıştır.

- 1435 mm'ye eşit olanlar hasarsız ve standart ölçü, 1435 mm'den küçük olanlar büzülme hatası, fazla olanlar ise genişleme hatası olarak kaydedildi.

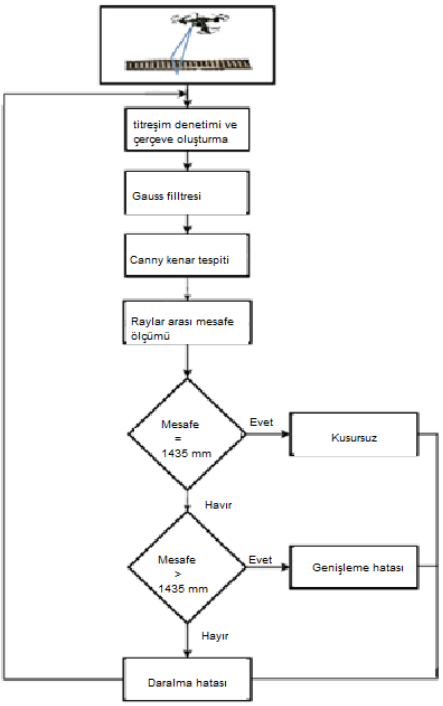
- Güneşli ve normal gün ışığında drone kullanılarak elde edilen ray görüntüleri için işlemler ayrı ayrı tekrarlanmıştır.
- Elde edilen sonuçlar deneysel sonuçlarda verilmiştir.

Türkiye Cumhuriyeti Devlet Demiryolları'nın şehir içi raylı sistem bölümleri drone ile izlenerek görsel veriler elde edildi. Demiryolu hattı boyunca 5 metre sabit yükseklikten hareket eden drone üzerine 1280x720 piksel çözünürlüğe sahip kamera yerleştirildi. Drone tarafından kaydedilen demiryolu videosundaki titreşim sesleri, nokta öznitelik eşleştirme algoritması kullanılarak Matlab video stabilizasyonu kullanılarak giderilmiştir. MATLAB Videoreader kullanılarak videodan çerçeveler oluşturulmuştur. Kare hızı 30 fps. Piksel kalibrasyonu için 5m yükseklikten ve bilinen 1435 mm çizgi açıklığına sahip bir görüntü kullanılmıştır. Görüntüdeki çizgi açıklığının piksel sayısı belirlendi ve mm'ye dönüştürülmüştür. Güneşli havada kaydedilen demiryolu görüntüsü örneği Şekil 3.86.a'da normal gün ışığında kaydedilen demiryolu görüntüsü örneği ise Şekil 3.86.b'de verilmiştir. Önerilen yöntemin akış şeması Şekil 3.87'de verilmiştir.

a)

b)

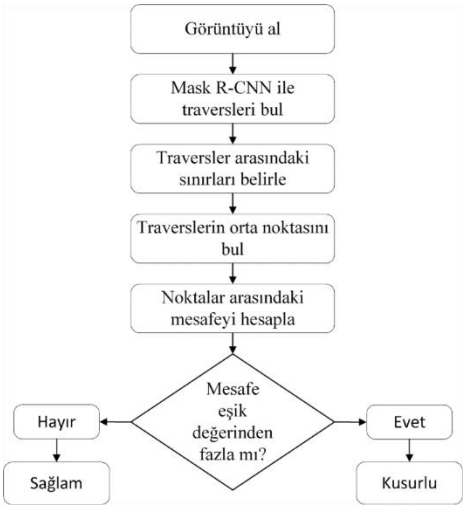
Şekil 3. 86. a) Güneşli havada kaydedilen görüntü b) Normal havada çekilen görüntü




Şekil 3. 87. Önerilen yöntemin akış şeması

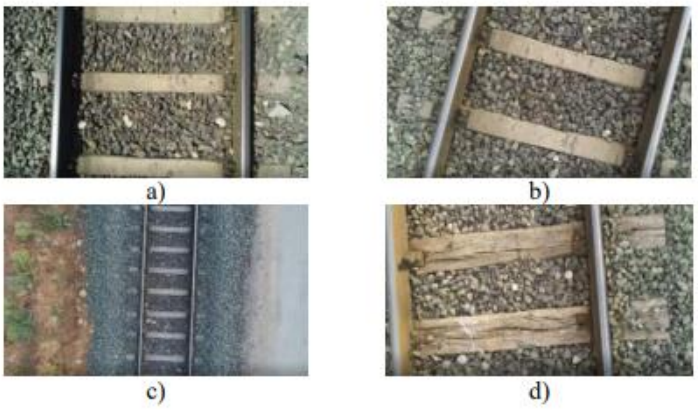
### 3.14. Mask R-CNN Kullanılarak Demiryolu Travers Aralığının Ölçülmesi

Önerilen yöntem, nesne bölütleme algoritması Mask R-CNN'ye dayanmaktadır. Amaç, tam konumlarını belirlemek için Mask RCNN algoritmasını kullanarak demiryolu görüntüsünden traverslerin ikili maskesini elde etmektir. Traverslerin alt ve üst limitleri belirlendikten sonra aradaki piksel sayısı ölçülerek kusurlu ve sağlam olarak sınıflandırılmıştır. Önerilen yöntemin akış şeması Şekil 3.88'de verilmiştir.



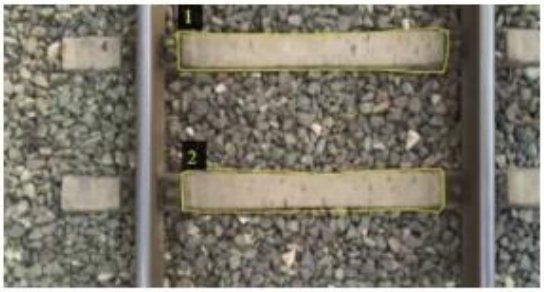

Şekil 3. 88. Önerilen yöntem

Bu çalışmada kullanılan veri seti, aktif olarak kullanılan bir demiryolu sahasından toplanmıştır. Veri seti farklı aydınlatma, yükseklik ve değişken açılardan oluşan senaryolardan alınmıştır. Böylece eğitim için kullanılacak veri setinde çeşitlilik oluşturulmuştur. Uçuşta çekilen videolar Parrot ANAFI 4K tarafından kaydedildi. Parrot ANAFI, uzaktan kontrol edilebilen bir uçaktır. Parrot ANAFI, bir el kumandası (Skycontroller 3) ile birlikte gelen ve bir akıllı telefon uygulaması (FreeFlight 6) aracılığıyla izlenebilen bir tüketici drone'udur. Drone, özel bir TCP bağlantı noktası aracılığıyla bir akıllı telefon uygulamasına akış yapabilen bir 4k HDR video kameraya sahip. Bu kamera tarafından çekilen video çözünürlüğü 4096 × 2160 piksel ve saniyede 24 karedir. Şekil 3.89, demiryolu sahasında farklı senaryolarda çekilen görüntülerin örneklerini göstermektedir.



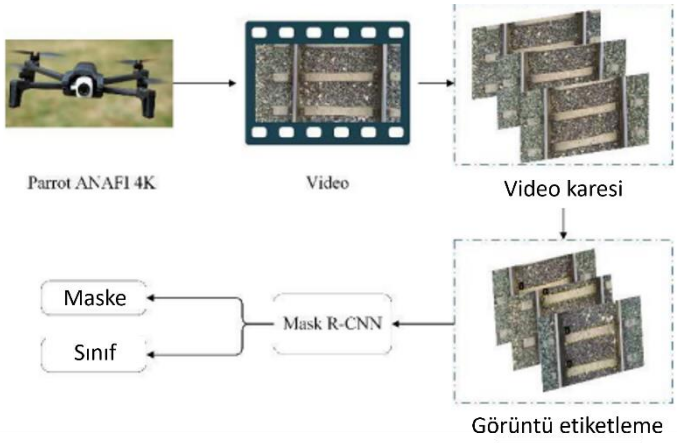

Şekil 3. 89. Farklı senaryolarda toplanan görüntüler a) Güneşli b) Farklı açılardan c) Yüksek uçuştan toplanan görüntü d) Ahşap traversler

Eğitim hızını artırmak için giriş görüntülerinin boyutu azaltıldı. Videolardan elde edilen veri seti 415×220 piksele kırpılmış ve JPEG formatında kaydedilmiştir. Veri seti, eğitim için 800 görüntü ve test için 750 görüntü olmak üzere 1550 görüntüden oluşmaktadır. Eğitim veri seti için görüntü etiketleme aracı olarak VGG Image Annotator (VIA) kullanıldı. Bu yardımcı program yardımıyla etiketlenmiş bir görüntünün ekran görüntüsü Şekil 3.90'da görülebilir. Her travers, dış kenar boyunca noktalarla seçilir ve görüntülerde kapalı bir döngüde bağlanana kadar bir etiket adıyla işaretlenir. Etiketlemeden sonra, temel olgu verileri, veri açıklaması için Mask RCNN için kullanılacak bir tablo biçimi olan json veri biçiminde depolanır.




Şekil 3. 90. VIA ile etiketlenmiş bir resim

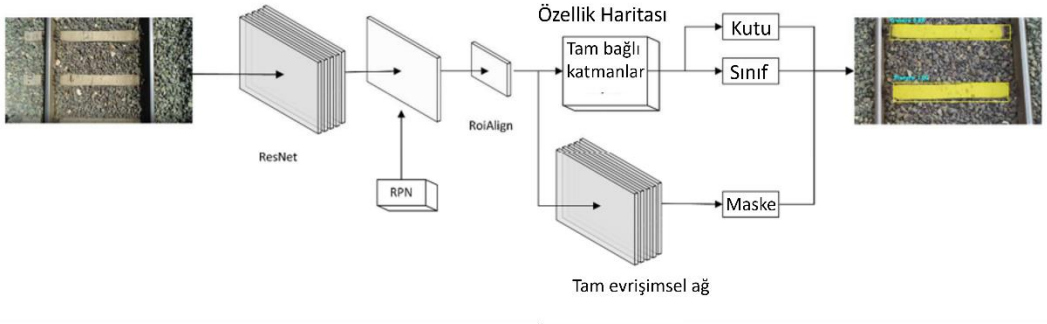
Şekil 3.91, Mask R-CNN için veri setinin nasıl oluşturulduğunu ve önerilen sistem için çalışma blok diyagramını göstermektedir. Toplanan demiryolu videosundan traversleri tespit etmek ve sınıflandırmak için video kareleri oluşturulmuş ve bu kareler üzerinde segmentasyon yapılmıştır. Videodan jpeg formatında oluşturulan ve yeniden boyutlandırılan görüntüler eğitim ve test için kullanılmıştır.




Şekil 3. 91. Önerilen yöntemin yapısı

Bu çalışmada, traverslerin doğru tespiti için Mask R-CNN yapısına dayalı bir algoritma önerilmiştir. Önerilen algoritma, dronun demiryolundan topladığı RGB formatındaki görüntülere uygulanmıştır. Mask RCNN algoritması, traversleri, traversleri sınırlayan kutuları, her maskenin etiketlerini ve her traversler için nesnellik değerini çıktı olarak içeren maskeler üretir. Mask R-CNN çerçevesi, Faster RCNN altyapısı ile tasarlanmış bölge tabanlı bir evrişimli sinir ağıdır. Bir bölge teklif ağı aracılığıyla oluşturulan bölge tekliflerine dayanmaktadır. Mask R-CNN, tespit edilecek nesneyi yüksek doğrulukla algılayabilen ve nesneyi doğru bir şekilde segmente edebilen bir algoritmadır. Daha Hızlı RCNN'den farklı olarak, ROI-Pooling işlemi Mask-RCNN tarafından değiştirilir. Bu işlem, doğru numune segmentasyon maskelerinin oluşturulmasını sağlar. Diğer fark, Mask-RCNN'nin, oluşturulacak örnek segmentasyonları oluşturmak için küçük, tamamen evrişimli bir sinir ağı eklemesidir. Mask R-CNN yapısı Şekil

3.92’de gösterilmiştir. Segmentasyon, instance segmentasyonu olarak yapılır. Örnek bölütlemede sınıftan bağımsız bölütleme yapılır. Amacımız, karmaşık bir demiryolu görüntüsündeki traversleri tespit etmek ve segmentlere ayırmak olduğundan, sonuçlar yalnızca travers sınıfına sahiptir.




Şekil 3. 92. Önerilen Mask RCNN segmentasyon yönteminin yapısı

Mask R-CNN mimarisi iki aşama içerir. Bu aşamalardan ilki bölgesel ihale ağıdır (RPN). İkinci adımda, Roi Align işlemi gerçekleştirilir ve nesneyi çevreleyen sınırlayıcı kutunun boyutu ve konumu bulunur ve her ilgilenilen bölge (ROI) için bir maske oluşturulur. Uygulamada, giriş görüntüsünün travers özellikleri çıkarılır. Bu işlem için ResNet-101 omurgası kullanılmaktadır. Ardından, RPN aracılığıyla özellik haritasında İlgi Bölgesi (RoI) oluşturulur. Ayrıca, Konumları tam olarak korumak ve özellik haritasını sabit boyutta tutmak için İlgi Bölgesi Hizalaması (RoI Hizalaması) kullanılır. Ağın sonunda, sınırlayıcı kutu algılama bölümünde bulunur ve sınıflandırılır. Travers maske, görüntü üzerinde segmentasyon olarak Full Convolution Network (FCN) tarafından oluşturulur. Son olarak maske bilgisi ve sınıf bilgisi ayrıştırılır. Maske ağ kafası, sınıf tahmini ağ kafasından bağımsız olarak maskeyi tahmin eder. Bu, çoklu görev kaybı fonksiyonunun kullanılmasını gerektirir. Mask R-CNN algoritmasında kullanılan kayıp fonksiyonu, sınıflandırma, sınır regresyonu ve segmentasyon kaybı fonksiyonlarından oluşur. Kayıp fonksiyonu denklem (3.48)'de verilmiştir.

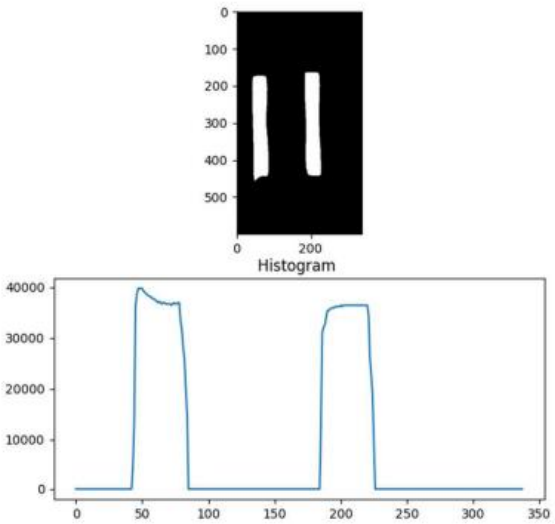
$$L = L_{cls} + L_{reg} + L_{mask} \tag{3.48}$$

Denklem (3.48)'de, Lcls, sınıflandırma sonucunun kayıp fonksiyonu, Lreg, sınıflandırma fonksiyonunun kayıp fonksiyonu ve Lmask, segmentasyon sonucunun kayıp fonksiyonudur.
Bu bölümde, traversler arasındaki mesafeyi ölçmek için kullanılan yaklaşımımızı detaylandırıyoruz. Bu bölümün amacı, giriş görüntüsü üzerinde traverslerin konumunu belirlemek ve traversler arasındaki mesafeyi hesaplamaktır. Bu amaçla kullanılan algoritmanın ilk adımı Mask R-CNN ile traverslerin ikili maskelerini elde etmektir. Ortaya çıkan görüntüde, birler ve sıfırlar sırasıyla beyaz ve siyah alanları temsil eder. İkinci adımda, traverslerin tepe

noktalarını tespit etmek için görüntü 90 derece döndürüldü ve tüm piksellerin toplamı alındı. Ortaya çıkan tepe noktaları, traverslerin konumunu temsil eder. Son aşamada ise pikler arası mesafe hesaplanarak kayıp travers olup olmadığı kontrol edilmiştir. Travers tespit etmek için Mask R-CNN tabanlı ölçüm yöntemiyle başlıyoruz. İlk olarak RGB renk uzayında elde edilen görüntü Mask R-CNN algoritmasına girdi olarak verilir. Daha önce bahsedildiği gibi, maske, kutu ve etiketli çıktı görüntüsü elde edilir. Sonuç, Şekil 3.93.a'da gösterilmektedir. Maske R-CNN sonucunda elde edilen Şekil 3.93.b'deki ikili maske görüntüsü, travers sınırlarının daha iyi tespiti için kullanılmıştır.

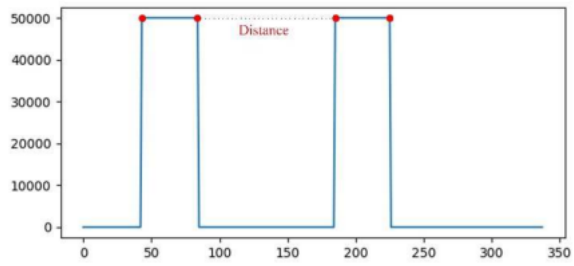

Şekil 3. 93. Mask R-CNN sonucu elde edilen görüntü a) Mask, kutu, etiket bilgisi b) İkili mask

Görüntüdeki bir traversin noktalarını belirlemek için önce görüntünün bir histogramı elde edilir. Histogram, görüntünün X ekseni yönü boyunca en büyük yoğunluğa sahip piksel konumunda bir tepe gösterir. Şekil 3.94, traveslere karşılık gelen maksimum değerlerle birlikte bir görüntü histogramı örneğini göstermektedir.




Şekil 3. 94. X ekseninde piksel yoğunluğu toplamının gösterimi

Görüntüdeki travers konumlarını belirlemek için, dikey yöndeki piksel yoğunluğunun toplamı, Şekil 3.94'te çizildiği gibi x eksenine yansıtılır. İkili maske görüntüsünde, travesler gibi pikseller arasında güçlü bir kontrast vardır. 1 ile ve diğer pikseller 0 ile gösterilir. Başka bir deyişle, traversli ve traverssiz bölge net ve kolay bir şekilde ayırt edilebilir. Bu nedenle, traversler Şekil 3.94'teki histogram grafiğinde belirgin bir şekilde gösterilmiştir. Ancak traverslerin tepe kenar çizgilerini tam olarak elde etmek için, tepe noktası tek bir değer olacak şekilde histogram grafiği yeniden çizilmiştir. Sonuç, Şekil 3.95'te verilmiştir.




Şekil 3. 95. X ekseninde piksel yoğunluğu toplamının gösterimi

Histogram, tepe noktaları sabit tutularak yeniden çizildi. Köşelerde gösterilen kırmızı noktalar, traverslerin başlangıç ve bitiş noktalarını temsil etmektedir. İlk travers kişinin bitiş noktası ile ikinci travers kişinin başlangıç noktası arasındaki piksel farkı hesaplanarak kayıp travers olup olmadığı tespit edilmiştir.

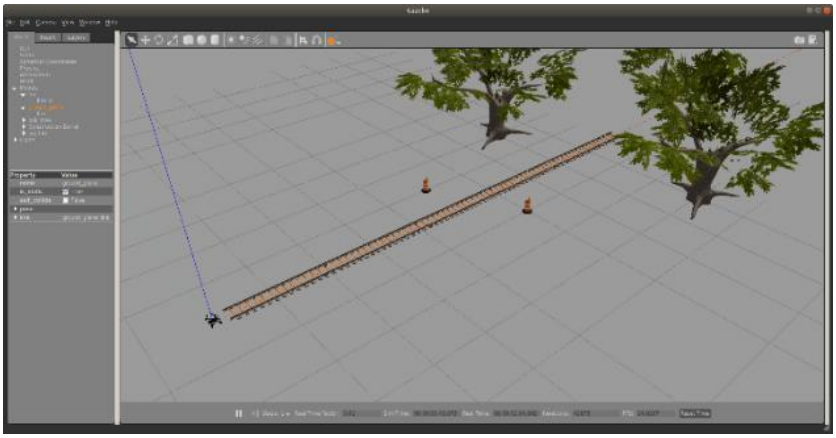
# 4. BULGULAR

## 4.1. Otonom Ray Takibi İçin Deneysel Sonuçlar

Otonom ray takibi için farklı İHA'lar ve farklı demiryolu sahaları kullanılarak testler yapılmıştır. Kontrol denetleyicisi olarak PID denetleyici ve bulanık denetleyici kullanılmıştır. Detaylı deneysel sonuçlar alt başlıklarda verilmiştir.

### 4.1.1. Simülasyon Ortamında İHA ile Otonom Uçuş ve Ray Takibi İçin Deneysel Sonuçlar

Gazebo, 3 boyutlu bir robot simülatörü olup çeşitli sensör modellerini, fizik motorlarını kullanmayı ve 3 boyutlu bir dünya ortamını simüle etmeyi sağlamaktadır [81]. Gazebo'nun önemli özelliklerinden birisi de İHA, ağaç, ev gibi popüler modelleri simülasyon ortamı için sağlamasıdır. Şekil 4.1'de Gazebo simülasyon ortamı görülmektedir.



Şekil 4. 1. Gazebo ortamı

Parrot Anafi4k Gazebo ortamıyla uyumlu çalışan bir İHA'dır. Bir adet ön kamerası mevcuttur. Demiryolu ray görüntüleri bu kamera ile kayıt altına alınmıştır. Yüksek menzilli kamera eğim açısı -90 ile +90 derece arası İHA üzerindeki nesneyi incelemeye izin vermektedir. Gazebo'da bir demiryolu ray modeli oluşturmak için 3D Warehouse'dan Collada formatında (.dae) 3 boyutlu bir model dosyası indirilmiştir [82]. Ardından, Gazebo ile uyumlu SDF dosyası oluşturulmuştur. Gerçek dünyada perspektiften bakıldığında demiryolu rayları ufukta buluşuyor gibi görünmektedir. Bu paralel hatların kesişme noktalarından izlenen demiryolunun ufuk noktası tanımlanabilir. Rayların görüntüden nasıl çıkarıldığına ilişkin prosedürün bir açıklaması aşağıda verilmiştir. Bir görüntüdeki çizgileri tespit edebilmek için, keskinleştirme, kenar algılama, yumuşatma gibi düşük seviyeli ön işleme tekniklerine ihtiyaç duyulmaktadır. Rayların

tespiti için öncelikle Gabor filtresi kullanılmıştır. Gabor filtresi yardımıyla Anafi4k İHA'sından alınan canlı görüntüler üzerinde belli bir yöne uzanan ayrıtlar tespit edilmiştir. Şekil 4.2'de Gabor filtresi çıktısı görülmektedir.

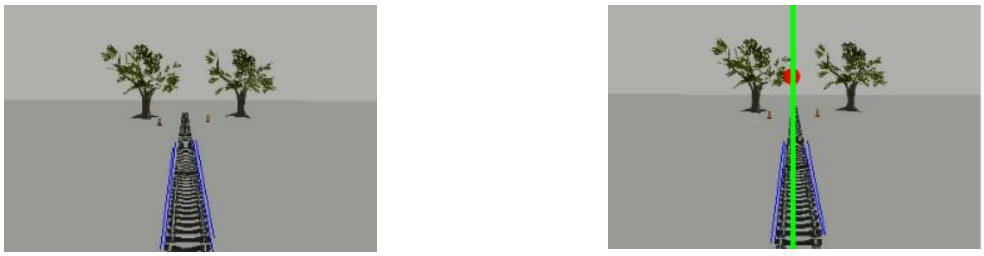
Şekil 4. 2. Gabor filtresi çıktısı

Elde edilen görüntüdeki kenarları tespit edebilmek için Canny kenar çıkarım yöntemi kullanılmıştır. Kullanılan Anafi4k modeli, ön kamerasından tespit edilen ufuk noktasına bağlı olarak demiryolu hattının merkezi boyunca konumunu otonom olarak koruyabilmektedir. Ufuk noktasına dayalı navigasyon, Gazebo ortamında ayarlanmış bir PID kontrolör ile gerçekleştirilmiştir. Ufuk noktası bulunduktan sonra klavyeden "T" tuşu ile takip başlatılmaktadır. Takip işlemini bitirmek için "ESC" tuşu kullanılmalıdır. Gazebo simülasyon ortamında bir İHA başarıyla ufuk noktasına dayalı otonom bir uçuş gerçekleştirmiştir. Gelecek saha testleri için yapılan bu simülasyon rehber niteliği taşımaktadır. Şekil 4.3'te Canny algoritmasının çıktısı görülmektedir.


Şekil 4. 3. (a) Canny algoritması çıktısı (b) Kırpılmış hali

Görüntüde bulunan raylar Hough dönüşümü kullanılarak çıkarılmıştır. Elde edilen Hough dönüşümünün başlangıç ve bitiş noktaları iki ayrı dizide tutularak en küçük kareler yöntemi ile

ufuk noktası tahmini yapılmıştır. Intersect() fonksiyonu, parametre olarak iki ayrı diziyi alarak ufuk noktasının koordinatlarını döndürmektedir. Şekil 4.4'te Hough dönüşümü sonucu ve tahmin edilen ufuk noktası Anafi4k'nın kamerasından görülmektedir.

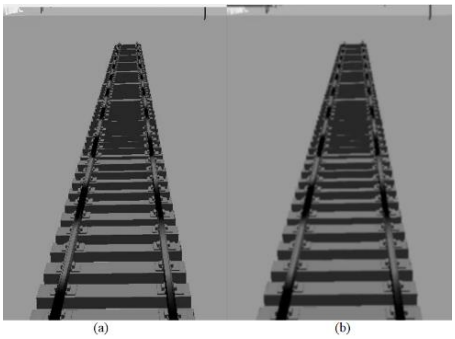


Şekil 4. 4. Hough dönüşümü ve ufuk noktası tahmini

Çalışmada Gazebo ortamında simülasyon ortamı için Anafi4K drone, ray, ağaç, posta kutusu, trafik işareti, bina gibi modeller eklenmiştir. Gazebo deney ortamı için yaratılan dünya Şekil 4.5'te gösterilmektedir.

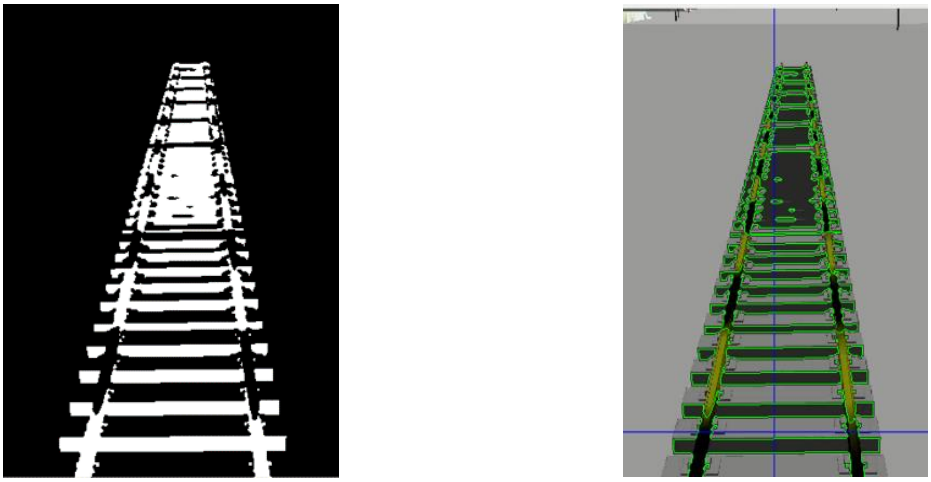

Şekil 4. 5. Gazebo deney ortamı

Ray takibi ile drone modelinin demiryolu hattını otonom olarak takip etmesi amaçlanıyor. Öncelikle Python ortamında drone kamerasından alınan gerçek zamanlı görüntü kırpılır. Buradaki amaç drone'un sadece ray kısmına odaklanmasını sağlamaktır. Daha sonra gerçek zamanlı görüntü gri tonlamaya dönüştürülür ve Gauss bulanıklığı uygulanır. Görüntüdeki bulanıklığı gidermek için Gauss kullanıldı. Bu adımların çıktıları Şekil 4.6'da gösterilmektedir.




Şekil 4. 6. a) Grayscale çıktısı b) Gaussian çıktısı

Görüntü, elde edilen görüntüye bir renk eşiği uygulanarak ikili bir değere dönüştürülür. Eşikli görüntü Şekil 4.7'de gösterilmektedir.



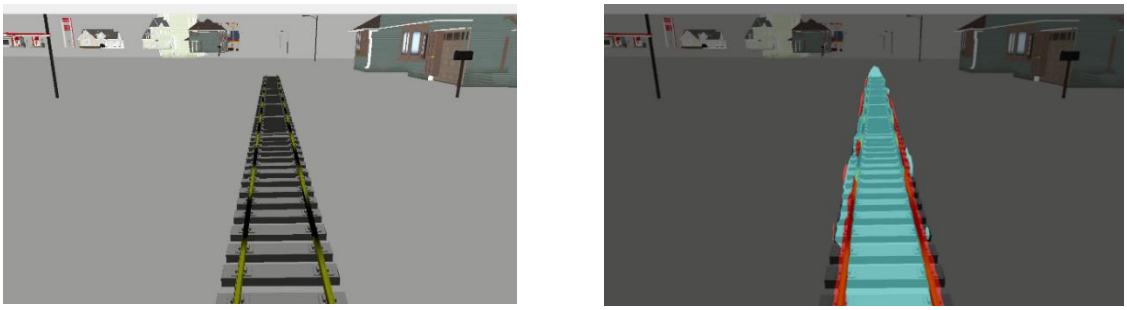
Şekil 4. 7. Eşikli çıktı ve tespit edilen rota

Eşikli görüntüde çizgiler algılanır. Tespit edilen hattın koordinatlarına göre drone'a uçuş talimatı verildi.

Çalışmada yapılan testler sonucunda elde edilen IoU değeri 0.99'dur. Tahmini ray görünümü Şekil 4.7'deki gibidir. BiSeNetV2 modeli, Tablo 4.1'de gösterildiği gibi literatürdeki diğer çalışmalardan daha iyi performans göstermiştir.

Tablo 4. 1. BiSeNetV2'nin bölütleme performansının literatürdeki diğer çalışmalarla karşılaştırılması

| **Model** | **IoU** |
|---|---|
| RailNet [83] | 0.89 |
| RailCNN [84] | 0.94 |
| BiSeNetV2 | 0.99 |



Şekil 4. 8. Tahmin edilen ray çıktısı

Gazebo ortamında simüle edilen Parrot Anafi modelinin kamerası ile demiryolu hattının ortasındaki konumunu otonom olarak koruyabiliyor. Bu süre zarfında aynı kamera ile gerçek zamanlı olarak semantik segmentasyon ile demiryolu hattı tespit edilebilir. Gazebo simülasyon ortamında, bir drone, ray takibine dayalı otonom bir uçuşu başarıyla gerçekleştirdi. Semantik segmentasyon, hata tespiti, malzeme sınıflandırması ve demiryolu hattı tespiti gibi problemlerde de kullanılmıştır. Çalışmada IoU değeri 0.99 olarak elde edilmiştir. BiSeNetV2 mimarisi, çoğu evrişimli model tarafından uygulanabilen genel bir mimaridir. BiSeNetV2 modeli, iyi bir segmentasyon doğruluğu ve çıkarım hızı elde etmiştir. Bu simülasyon, gelecekteki saha testleri için bir rehberdir. Gelecekteki çalışmalarda, Gazebo ortamında demiryolu dışındaki demiryolu bileşenlerinin: balast, bağlantı elemanlarının tespit edilmesi amaçlanmaktadır.

### 4.1.2. HSV'ye Dayalı Ray Takip Algoritması

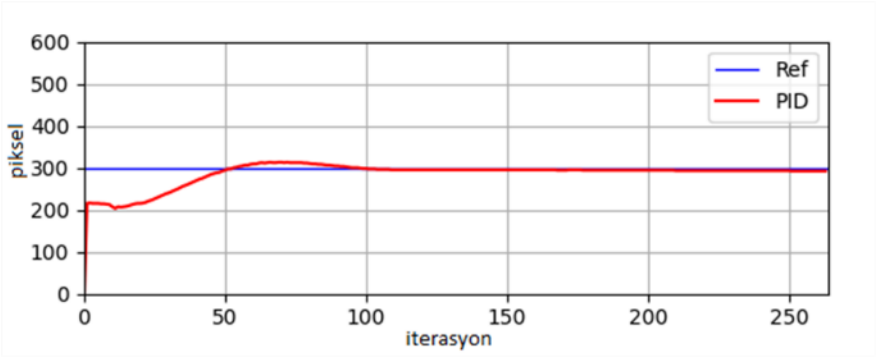
HSV'ye dayalı ray takip algoritması gerçek demiryolu sahasında test edilmeden son kontrolleri videolar üzerinde yapılmıştır. Algoritma, gerçek demiryolu ortamında denenmeden önce demiryolu ortamından alınan videolar üzerinde en iyi eşik değerleri belirlenmiştir. Bu videolar, İHA ile farklı koşullarda manuel kontrolle toplanmıştır. Buradaki amaç, algoritmanın farklı koşullarda performansını doğrulamaktır. Yapılan deneyde, farklı ray koşullarında uygulamanın iyi çalıştığı görülmüştür.

Son aşamada algoritma, aktif olarak kullanılan demiryolu üzerinde test edilmiştir. Şekil 4.9'da demiryolu sahasında yapılan test anlarına ait görüntüler görülmektedir. Takip için PID denetleyici kullanılmıştır. Uygulamada, Kp değeri 0.09, Ki değeri 0.05 ve Kd değeri 3 olarak belirlenmiştir. Şekil 4.9.b'de İHA'nın rayların ortasını yakalama anı gösterilmiştir.

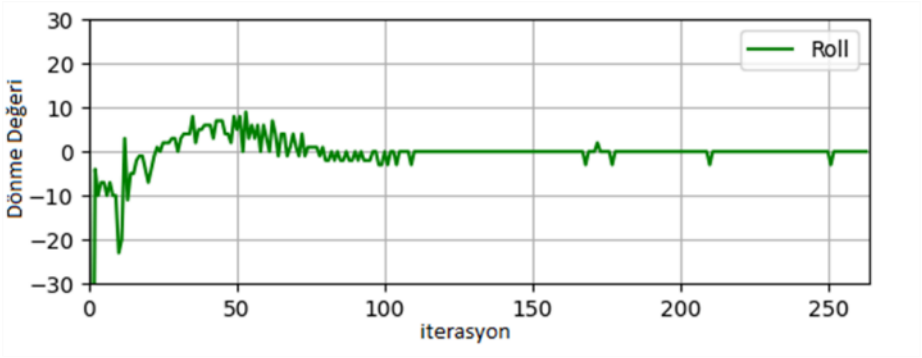
Şekil 4.10.a'da PID denetleyici kullanılarak ray takibi yapılan İHA'nın, rayların ortasını yakalamasını göstermektedir. Şekil 4.10.b'de ise PID denetleyici çıktısının İHA'ya dönüş değeri olarak gönderilmesi sırasında oluşan değerleri göstermektedir.

a) b)

Şekil 4. 9. Ray takibi algoritmasının sahada test edilmesi (b) İHA'nın istenen değeri yakalaması




a)




b)

Şekil 4. 10. a) Demiryolu sahasında ray hattını takip eden İHA'nın orta noktayı yakalaması b) İHA'ya gönderilen dönme değerleri

### 4.1.3. Gabor Filtresi ve Bulanık PID Tabanlı Ray Takip Algoritmasının Deneysel Sonuçları

Önerilen yöntemde ilk olarak alınan görüntü, önişleme adımlarından geçirilmelidir. Görüntü önişleme, kameradan gelen görüntülerle ilgilenen ve yol çizgilerini algılamak için yararlı bilgiler üreten düşük seviyeli görüntü işlemedir. Ardından yukarıda anlatılan HSV'ye dayalı ray tespiti yöntemi ile ray çizgileri tespit edilmiştir. Son olarak, hesaplama süresinden tasarruf etmek için

görüntü daha küçük bir ilgi alanına (ROI) indirgenmiştir. İHA tarafından yakalanan demiryolu görüntülerinde bulunan rayların şekli sabittir ve önemli ölçüde değişmez. Yine de hesaplama maliyetlerinin azaltılması için belirli bölge anlamına gelen ROI (Region of İnterest) belirleme işlemi yapılmıştır.

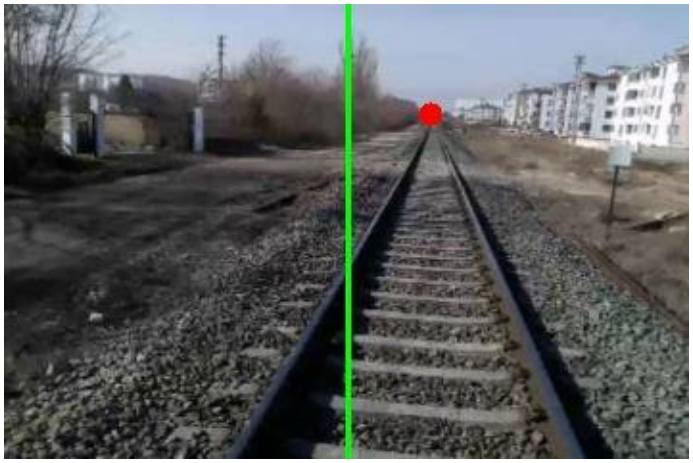
ROI işlemi sonrasında ise ana işlem başlatılmıştır. Bu işlemde, elde edilen kenarlara Hough dönüşümü uygulanmıştır. Hough dönüşümü kullanılarak tespit edilen çizgilerin eğimlerinin ortalaması hesaplanmıştır. Sol ray çizgisi negatif eğime sahiptir. Bu nedenle, görüntüde elde edilen ray çizgilerinden negatif eğimli tüm çizgiler sol şerit noktaları olarak kabul edilir. Sol çizgilerin eğim ortalaması hesaplanarak tek bir sol çizgi oluşturulmuştur. Aynı şekilde sağ şerit çizgisi ise pozitif eğim değerine sahiptir. Pozitif eğimli tüm çizgiler ise sağ şerit çizgileri olarak kabul edilir. Sağ çizgilerin ortalamasından ise tek bir sağ çizgi oluşturulmuştur. Görüntüde bulunan dikey çizgiler ise atlanmıştır. Bu işlemin amacı sadece iki farklı çizginin kesişiminden tek ufuk noktası elde etmektir. Önerilen yöntemin amacı sağ ve sol çizgilerin kesişiminden ufuk noktası elde etmektir. Sonuçlar Şekil 4.11'de gösterilmiştir.

Şekil 4. 11. Sağ ve sol çizgilerin oluşturulması

Şekil 4.11'de yeşil renk ile gösterilen iki çizginin kesişiminden oluşturulan ufuk noktası Şekil 4.12'de gösterilmiştir. Ufuk noktası kırmızı nokta ile işaretlenmiştir.

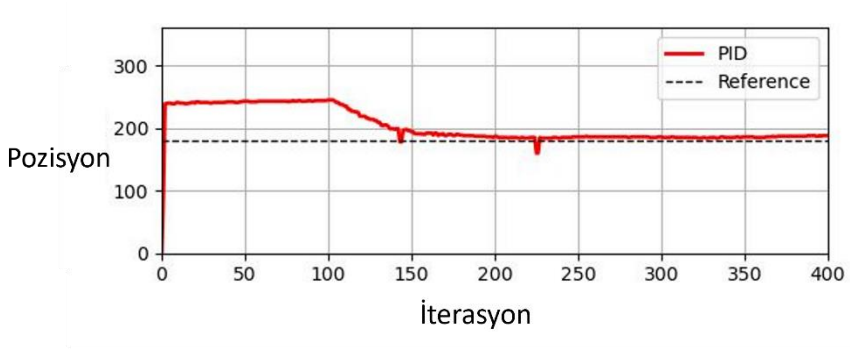
Şekil 4. 12. Ufuk noktasının kırmızı nokta ile işaretlenmesi

PID denetleyici kullanarak takip işlemini gerçekleştirebilmek için ilk olarak, İHA'dan alınan görüntünün genişliği 360 px ve yüksekliği 240 px olarak alınmıştır. Bu değerlerin düşük çözünürlüklü seçilme sebebi görüntü işlemede hızı yükseltmektir. İHA'nın demiryolunu ortalaması için alınan görüntünün x ekseninde tam orta noktası olan 180. pikseli istenen değer olarak seçilmektedir. Böylece İHA, ufuk noktası ile orta noktayı birbirine yaklaştıracak ve yolun ortalanması işlemi gerçekleşecektir. Görüntünün orta noktası Şekil 4.13'te gösterilen yeşil çizgi, x ekseninde bulunan 180. pikselin y ekseni boyunca çizilmesinden oluşmuştur.



Şekil 4. 13. Ufuk noktası ile merkez konumun gösterimi

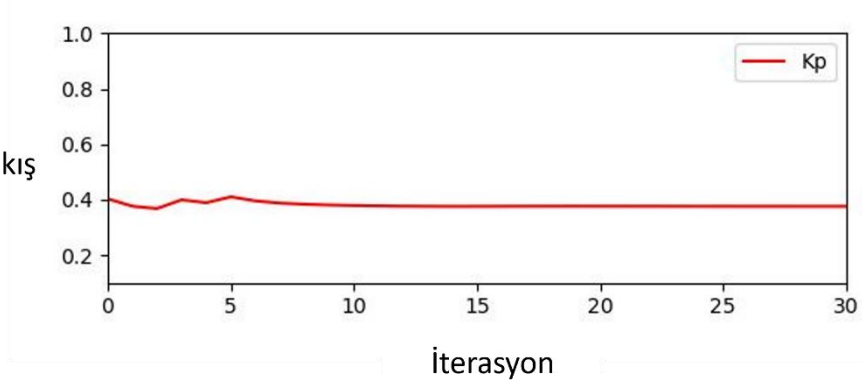
PID kontrol sistemi için Kp değeri 0.4, Ki değeri 0.2 ve Kd değeri 0.4 olarak belirlenmiştir. Bu değerler kullanılarak hatanın sıfıra inme süresinin makul seviyede olduğu gözlenmiştir. Önerilen algoritma gerçek demiryolunda test edilmiştir. Sistemin ufuk noktasını yakalaması Sekil 4.14'te gösterilmiştir.



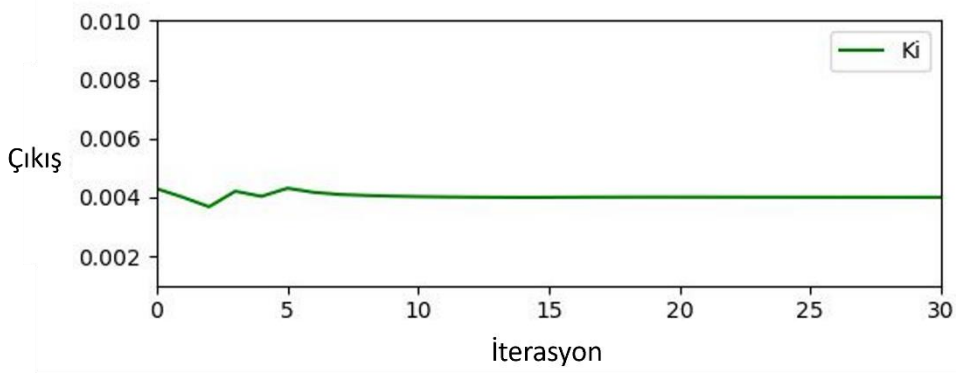

Şekil 4. 14. Referans noktasına göre PID çıkışı

PID kontrolörünün çıkışı, İHA'nın sapma kanalına iletilmiştir. Bu sayede İHA'nın yolun ortasında hareket etmesi sağlandı. Bu süreçte İHA sabit bir hızla ileri yönlü hareket yapmaktadır. Yüksekliği ise sabittir. Testler, PID kontrolörünün dört rotorlu bir İHA'yı kontrol etmede etkili olduğunu göstermektedir.

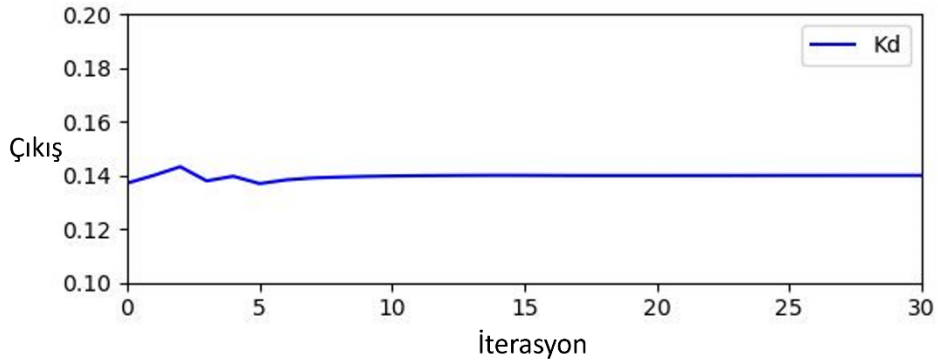
Kullanılan Bulanık-PID yapısındaki PID kazanımları, her iterasyon sonunda hataya ve hatanın türevine göre değişecektir. Bu durumun bir sonucu olarak klasik PID denetleyiciye göre daha dinamik kontrol sağlanacaktır. Kp, Ki ve Kd değerlerinin değişimi Şekil 4.15, Şekil 4.16, Şekil 4.17'de gösterilmiştir.




Şekil 4.15. Kp'nin değişimi

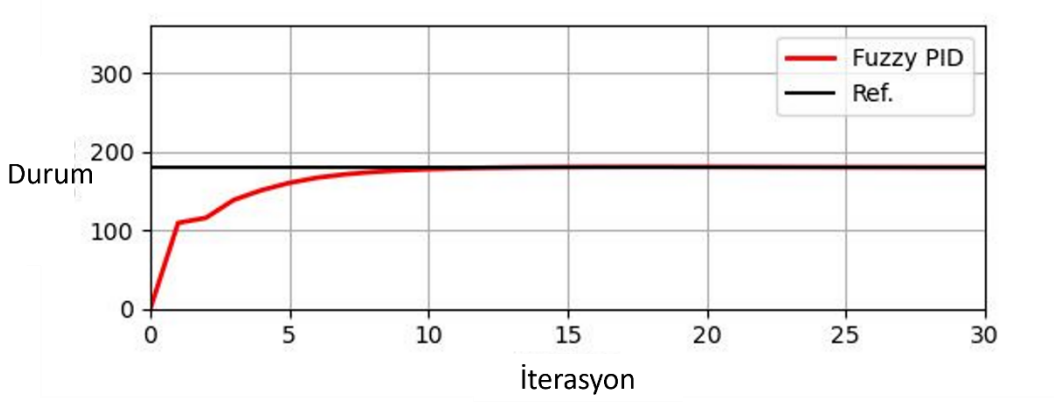



Şekil 4.16. Ki'nin değişimi

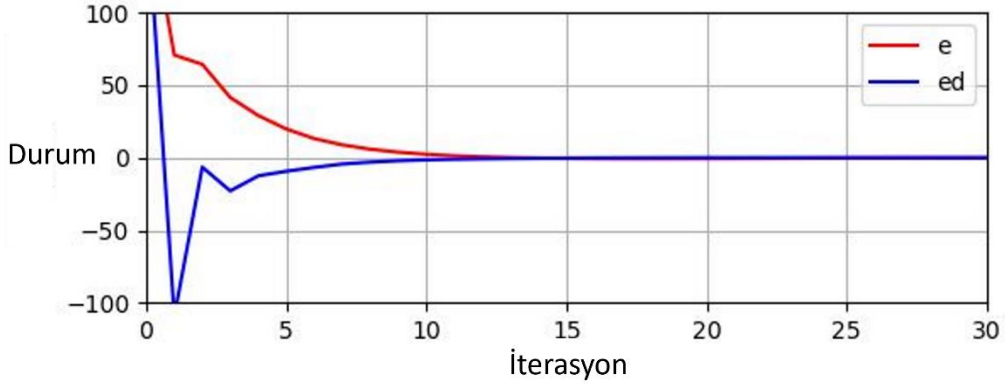



Şekil 4.17. Kd'nin değişimi

Şekil 4.18'de, referans değerini yakalayan Bulanık PID kontrolörü gösterilmektedir. Şekil 4.19, hatanın (e) ve hatanın (ed) türevinin sıfıra yakınsadığını göstermektedir.



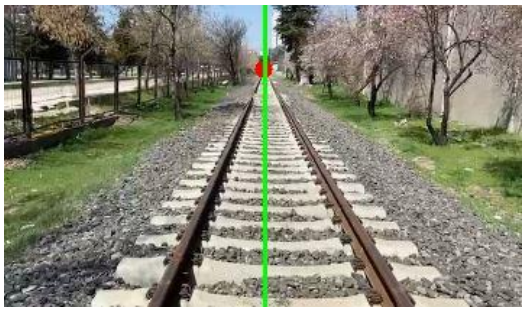

Şekil 4.18. Referans değerinin bulanık PID




Şekil 4.19. Hata(e) ve hatanın türevinin(ed) sıfıra yakınsaması

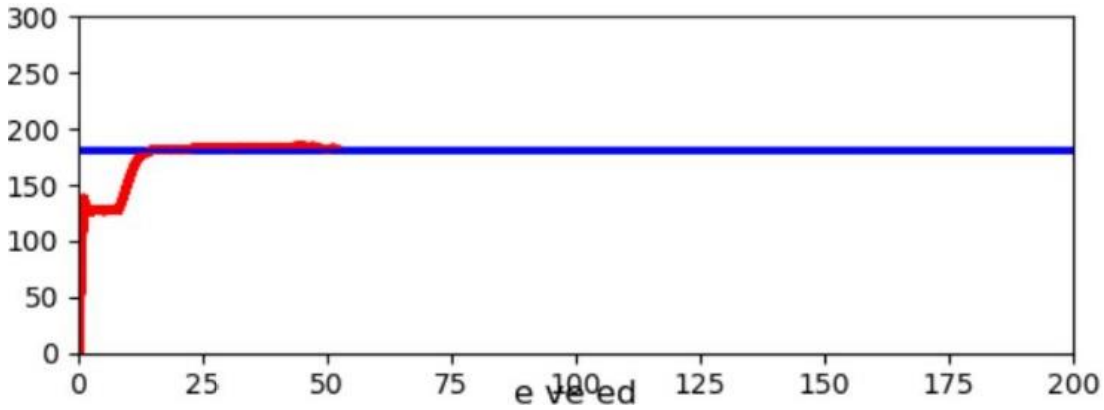
Tasarlanan demiryolu takip algoritması da sahada test edilmiştir. Algoritmayı test etmek için 720p çözünürlüklü ve 30 fps video kaydına sahip DJI Ryze Tello kullanılmıştır. İHA, demiryolu üzerinde konumlandırılmıştır. İHA'dan alınan görüntülerden ufuk noktasının bulunmasının ardından takip sürecine geçilmiştir.

İHA'dan alınan görüntü Şekil 4.20'de gösterilmektedir. Burada kırmızı işaret ufuk noktasını temsil etmektedir. Görüntünün orta noktası olan 180. pikselden dikey eksende yeşil çizgi çizildi.

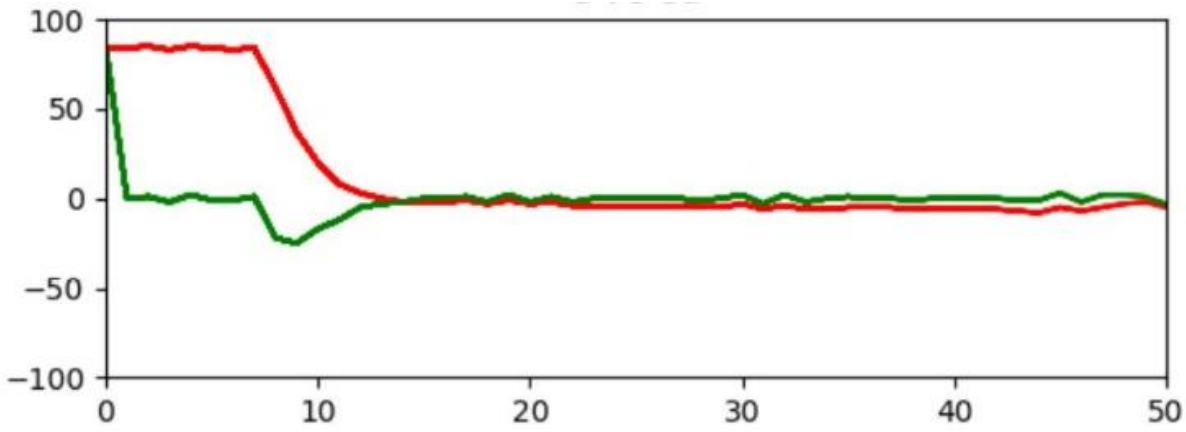


Şekil 4. 20. Ufuk noktasının bulanık-PID ile yakalanması

İzleme komutu verildiğinde, PID kontrol parametreleri, mevcut hata durumuna ve hatanın türevine göre bulanık sistem tarafından ayarlanır. Bu işlem, çekilen her kare için devam etmektedir. Böylece klasik PID'ye göre daha dinamik bir sistem tasarlanmış ve bu sistem hedefi klasik PID'ye göre daha sorunsuz yakalamıştır. Bunun nedeni, bulanık sistem sayesinde PID kontrol parametrelerinin hedefe yaklaştıkça azalmasıdır. Bu işlem sonucunda hedefe ne kadar yakınsa sapma kanalına o kadar düşük değer atanır ve yumuşak bir dönüş sağlanır. Şekil 4.21, İHA tarafından sahada elde edilen ufuk noktasının yakalanmasını göstermektedir. Şekil 4.22, hatanın (e) ve hatanın türevinin (ed) sıfıra yakınsamasını gösterir.




Şekil 4. 21. Ufuk noktasının İHA tarafından yakalanması

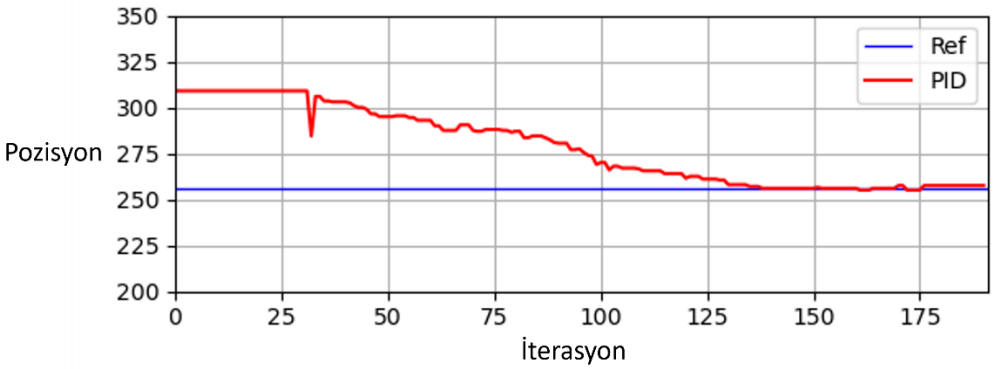



Şekil 4. 22. Deney esnasında hatanın (e) ve hatanın türevinin (ed) sıfıra yakınsaması

Bulanık PID sistemini tercih etmemizin nedeni klasik PID'den daha verimli çalışmasıdır. Tasarlanan sistemin sahada test edildiği ve verimli çalıştığı, İHA'nın demiryolunu sabit bir hızda otonom olarak takip edebildiği görülmüştür.

### 4.1.4. YOLACT Tabanlı Ray Takibi

İHA'dan alınan görüntünün genişliği 480 px ve yüksekliği 240 px olarak alınmıştır. İHA'nın demiryolunu ortalaması için alınan görüntünün x ekseninde tam orta noktası olan 240. pikseli istenen değer olarak seçilmektedir. Böylece İHA, ufuk noktası ile orta noktayı birbirine yaklaştıracak ve yolun ortalanması işlemi gerçekleşecektir. Önerilen algoritma test edilmiştir. Sistemin rayların orta noktasını yakalaması Şekil 4.23'te gösterilmiştir.




Şekil 4.23. PID denetleyici sonucu

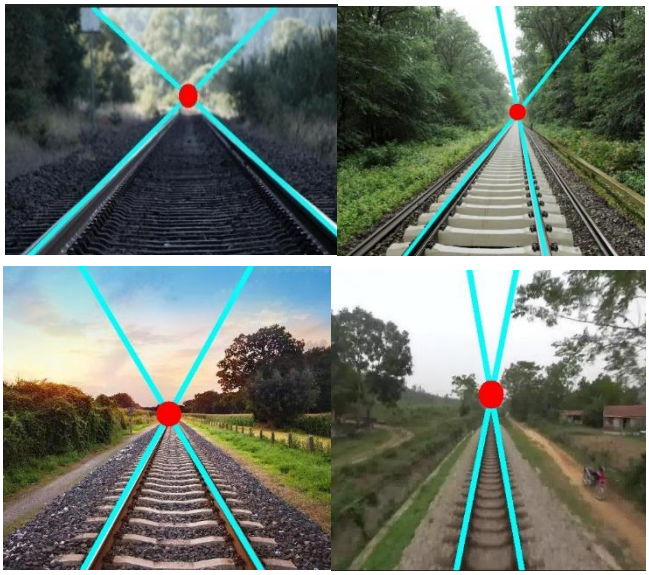
Şekil 4.23'te görüldüğü gibi ray takibi, rayların orta noktasının alınan görüntünün merkezine yakınsaması ile sağlanmıştır. İki nokta bir arada tutulurken ileri yönlü hız ve irtifa sabittir. Böylece İHA, demiryolunu otonom olarak gezinmektedir.

### 4.1.5. Derin Hough Dönüşümü Tabanlı Otonom Ray Takibi ve Kusur Tespiti

İHA'nın otonom uçuş yapabilmesi ve farklı ışıklandırma koşullarında sitemin doğru çalışabilmesi için İHA ile alınan görüntülerin yanı sıra internet ortamından da 1000 adet görüntü toplanmıştır. Elde edilen görüntüler daha çok ufuk noktası tespiti için uygun görüntülerden seçilmiştir. Şekil 4.24'te kullanılan görüntülerden bazıları verilmiştir.

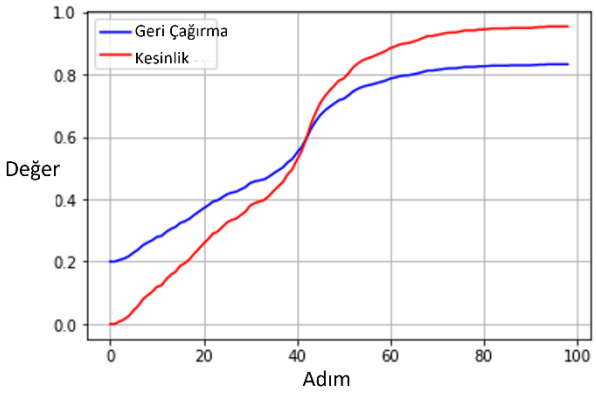
Şekil 4. 24. Ufuk noktası tespiti için kullanılan bazı görüntüler

Veri seti örneklerinden bazıları internet ortamında alınırken son görüntü kendi veri setimizden oluşturduğumuz verilerden oluşmaktadır. Derin Hough dönüşümü ile sonucu bulunan ray görüntüleri ve çizilen ufuk noktası Şekil 4.25'te verilmiştir.



Şekil 4. 25. Farklı görüntüler için derin hough dönüşümü sonuçları ve ufuk noktası tespiti

Şekil 4.25'te gösterildiği gibi derin Hough dönüşümü ile hem ray hatları farklı ışıklandırma koşullarında belirlenebilmekte hem de iki ray bileşenin kesişim noktası doğru bir şekilde elde edilmektedir. Derin Hough dönüşümünün eğitim seti üzerindeki kesinlik ve geri çağırma değerleri Şekil 4.26'da verilmiştir.




Şekil 4.26. Derin Hough dönüşümü için kesinlik ve geri çağırma değerleri

Şekil 4.27'de verilen derin Hough dönüşümü kesinlik ve geri çağırma değerlerine göre yüksek başarım elde edildiği görülmektedir. Derin Hough dönüşümünün en önemli avantajı ray ile ilgili çizgilerin tam bir şekilde tespit edilebilmesi ve Hough dönüşümünde olduğu gibi çoklu kenarlar bulunmamaktadır. İHA'nın ön kamerasından alınan görüntüler ile İHA konumu PID ile tam ray ortasına konumlandırıldığı için alt kamera ile tam ray ortasından ray görüntüleri alınmaktadır. Şekil 4.27'de alt İHA kamerasından alınan ray görüntüsü görülmektedir.

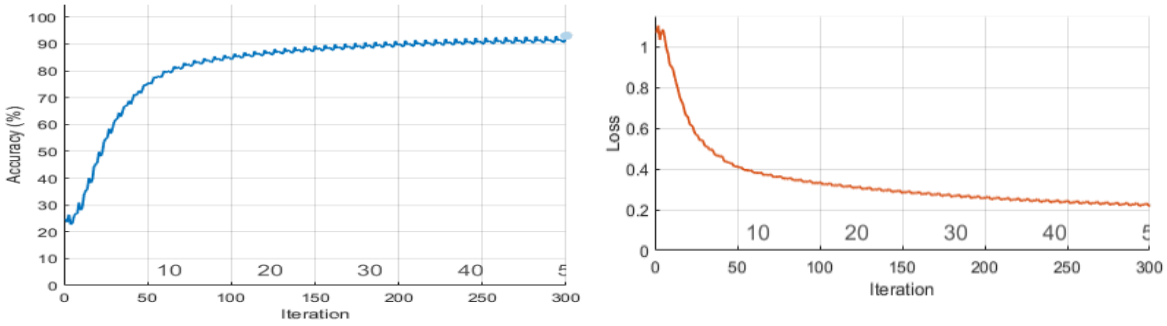
Şekil 4. 27. İHA alt kameradan ray görüntüsü

Şekil 4.27'de verilen ray görüntüsünde İHA rayın tam ortasından görüntü aldığı için sol ve sağ raylar iki ayrı görüntü olarak derin öğrenme tabanlı bölütleme algoritmasına verilmektedir. Tablo 4.2'de kullanılan derin öğrenme tabanlı bölütleme algoritmasının eğitim parametreleri verilmiştir.

Tablo 4. 2. Eğitim parametreleri

| Parametre | Değer |
|---|---|
| Maksimum adım sayısı | 50 |
| Mini Batch boyutu | 16 |
| Öğrenme oranı | 1e-3 |
| Doğrulama frekansı | 15 |

Tablo 4.2'de verilen eğitim parametrelerine göre derin öğrenme tabanlı bölütleme algoritması çalıştırılmıştır. Şekil 4.28'de eğitim süreci ve kayıp grafiği gösterilmiştir. Şekil 4.28'de verilen eğitim sürecine göre doğruluk oranı %93.05 olarak elde edilmiştir. Doğruluk oranı elde edilen görüntülerden alınan piksel sınıflandırma doğruluğudur. Diğer performans ölçütleri Tablo 4.3'te verilmiştir.




(a) Eğitim grafiği (b) Kayıp grafiği

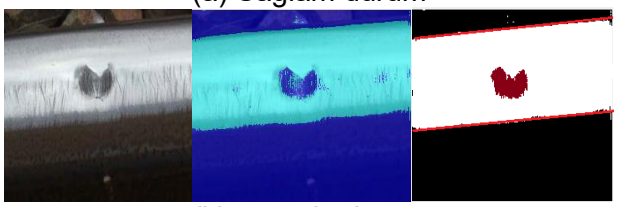
Şekil 4. 28. Eğitim ve kayıp grafiği

Tablo 4. 3. Bölütleme performans ölçütleri

| Ölçüt | Değer (%) |
|---|---|
| Global accuracy | 91.46 |
| Mean accuracy | 91.82 |
| Mean IoU | 78.18 |
| Weigheted IoU | 85.44 |
| F1 Score | 47.67 |

Tablo 4.3'te verilen sonuçlar bölütleme sonucunda ray bileşenin yüksek doğrulukta bölütlendiğini göstermektedir. Ray bölütleme işleminden sonra kusur belirleme algoritması çalışmaktadır. Şekil 4.29'da bölütleme sonucunda kusur tespit algoritmasının sonuçları verilmiştir.

(a) Sağlam durum



(b) Kusurlu durum

Şekil 4. 29. Bölütlenmiş ray görüntüsünden kusur tespiti

Şekil 4.29.a'da sağlam durumda herhangi bir kusur belirlenmemiş olup rayın sınırları belirlendikten sonra kusur tespit algoritması herhangi bir sonuç bulmamıştır. Şekil 4.29.b'de ise ray yüzeyindeki kusur bölütleme işleminden sonra rayın kenarları arasındaki piksellerin yoğunluğuna göre belirlenmektedir. Burada eşik değer her bir sütun için ilgili alanda kusurlu piksel sayısının kusursuz piksele oranı 0.2'den büyük ise o alan kusur olarak işaretlenmektedir. Bu çalışmada otonom bir İHA'nın kontrolü ve İHA'dan alınan görüntülerde kusurların tespit edilmesi için derin öğrenme tabanlı teknikler geliştirilmiştir. İHA'nın ön kamerasından alınan görüntülerden derin Hough dönüşümü tabanlı ray tespiti ve takibi ile görüntüler toplanmıştır. Elde edilen görüntülere bölütleme işlemi uygulanarak sağ ve sol ray belirlenmiştir. Elde edilen ray görüntülerinden kusurlu bölgeler tespit edilmektedir. Önerilen yöntem hem bölütleme açısından doğru sonuçlar vermektedir. Hem de kusur tespiti için yeni bir yöntem önerilmiştir. Önerilen yöntem bölütleme hatalarını kusurdan ayırt etmektedir. Çalışmada temelde iki katkı sunulmaktadır. Birinci katkı otonom uçuş için derin Hough dönüşümü tabanlı bir yöntem önerilmektedir. Önerilen yöntem ile farklı ışıklandırma koşullarında raylar doğru bir şekilde tespit edilmektedir. Elde edilen görüntülerden kusur tespiti için yeni bir yaklaşım sunulmaktadır.

### 4.1.6. Hafif Hat Segmentasyon Yaklaşımı ile Otonom İHA Tabanlı Ray Takip ve Travers Muayenesi

Bu çalışmada otonom bir İHA ile ray takibi ve veri toplama için Parrot Anafi İHA kullanılmıştır. Bu İHA Python programlama dili ile programlanabilmekte ve Gazebo simülasyon ortamında hem gerçek İHA ile hem de Olympe üzerinden otonom uçuşlar yapılabilmektedir. Şekil 4.30'da İHA'nın gerçek bölgede test edilmesi ve alınan bir görüntü verilmektedir.

(a) İHA'nın demiryolu bölgesinde test edilmesi

(b) Örnek görüntü

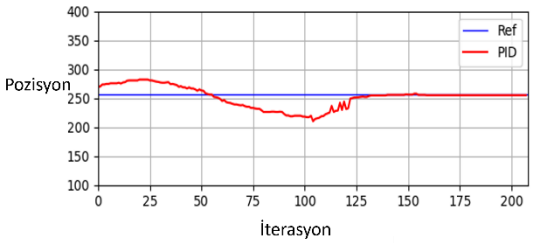
Şekil 4. 30. Otonom İHA saha testleri

Şekil 4.30'daki saha testleri sırasında uçuş bilgilerine erişim sağlanabilmektedir. Ayrıca GPS ile elde edilen görüntülerin konum bilgileri de kayıt altına alınmaktadır. Veriler sahada 4 km'lik bir alanda toplanmıştır. Uçuşlar genellikle 5 metreden daha alçak irtifalarda yapılıyordu. Bu test sırasında belirli bir yükseklikten İHA uçuşu yaptıktan sonra eve dönüş komutu verilerek kalkış noktasına geri dönülmüştür. Bu nedenle ilk dakikanın ortasında İHA 30 metre yüksekliğe çıkarak evine dönmüştür.

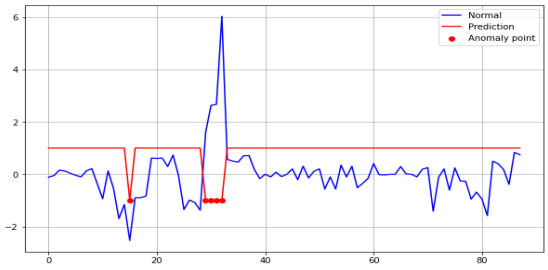
Şekil 4. 31. Tespit edilen ray ve traversler

Bu çalışmada büyük bir hafif hat bölütleme yöntemi kullanılmıştır. Çalışmada, hafif hat bölütleme modeli kullanıldığında rayların tespit edilmediği görülmektedir. Büyük model kullanıldığında ray konumlarının birden fazla çizgi ile gösterildiği görülmektedir. Takip için sağ ve sol raylar tek çizgi ile gösterilmelidir. Bu nedenle, x eksenine ilişkin görüntünün merkezinin sağındaki çizgilerin ortalaması, sağ rayı ve soldakilerin sol rayı göstermek için ortalaması alınır. Aynı durum birbirine yakın traversler için de elde edilmiş ve traversin üst ve alt kenarları gösterilmiştir. Şekil 4.31, sol ve sağ ray bileşenlerini ve travers bileşenlerini göstermektedir. Şekil 4.31'deki sağ ve sol raylar için elde edilen çizgilerin x ekseni boyunca orta noktası bulunarak İHA'nın konumu ile iki rayın merkezi arasındaki fark en aza indirilecektir. Bu amaçla İHA'nın PID sistemi ile rayı takip etmesi sağlanacaktır. PID kontrol sistemi sonucu Şekil 4.32'de verilmiştir. Şekil 4.32'de görüntünün merkezi ile sol ve sağ rayın orta noktası arasındaki mesafe PID kontrolü kullanılarak en aza indirilmeye çalışılmıştır. Traversler arasındaki mesafe bir

zaman serisi olarak kaydedilir ve mesafelerdeki değişim bir anomali olarak temsil edilir. Bu amaçla izolasyon orman anomali tespit yöntemi kullanılmıştır. Şekil 4.33'te, 45 kare için elde edilen zaman serilerinde meydana gelen bir anomali için tespit sonucu gösterilmektedir.




Şekil 4.32. İHA'nın PID kontrol sistemi ile konum kontrolü




Şekil 4. 33. Traverslerde anomali tespiti

İzolasyon orman yöntemi sağlıklı veriler için 1 değerini verirken, anomali verisi durumunda -1 değerini vermektedir. Literatürde travers tespiti ile ilgili çalışmalar yapılmış olmasına rağmen, traverslerde değişim ve traversleri sayma çalışmaları oldukça sınırlıdır. Tablo 4.4'te önerilen yöntemin travers tespit oranı için literatüre göre karşılaştırma sonuçları verilmiştir.

Tablo 4. 4. Travers için tespit oranı karşılaştırma sonuçları

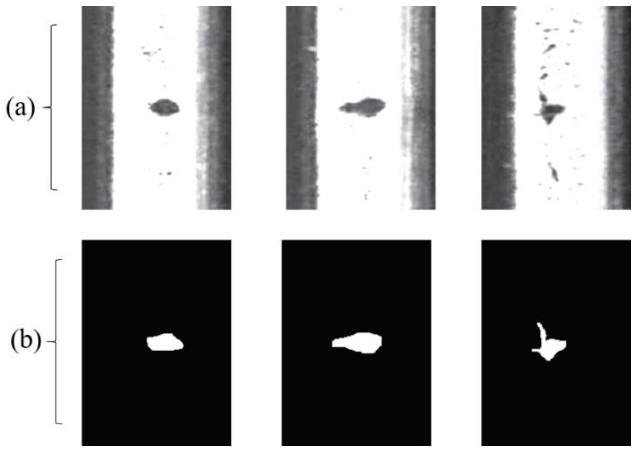
| **Referans** | **Metot** | **Tespit Oranı** | **Tespit Zamanı (saniye)** |
|---|---|---|---|
| [85] | YoloV4 | 92.0 | 12 |
| [86] | Kenar algılama ve klasik görüntü işleme | 59.0 | 0.52 |
| Bizim | Hafif hat segmenti algılama | 97.77 | 0.25 |

Tablo 4.4'te verilen elde edilen sonuçlar incelendiğinde önerilen yöntemin hem travers tespit oranı hem de tespit süresi açısından daha iyi olduğu görülmektedir. [85]'de İHA görüntülerinden traverslerin tespiti YoloV4 ile gerçekleştirilmiştir. Yöntemleri gerçek zamanlı

çalışma için uygun değildir. [86]'da trenin altından alınan görüntülerde travers pozisyonu belirlemek için bir çalışma yapılmıştır.

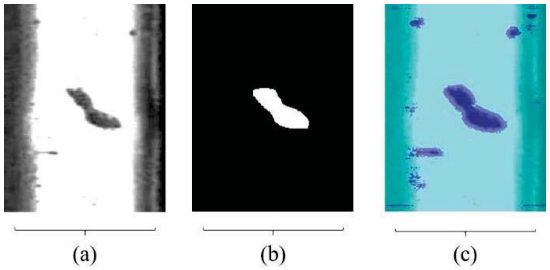
## 4.2. Semantik Segmentasyon Kullanarak Demiryollarındaki Kusurlarının Tespiti İçin Deneysel Sonuçlar

Veri setindeki ham görüntüler DATEM'den alınmıştır. Ham görüntüler tarafımızca bölütlenmeden önce görüntü ön işleme ve iyileştirme sürecinden geçirilmiştir. Kullanılan veri seti, 603 ray kusuru içeren görüntülerden oluşmaktadır. Bu görüntülerin 453'ü eğitim için, 150'si ise test için kullanmıştır. Eğitim ve test veri setlerinde orijinal gri tonlamalı görüntülerin yanı sıra referans görüntüleri de mevcuttur. Kullanılan veri setindeki kusurları içeren ray görüntüleri Şekil 4.34'te gösterilmiştir.




Şekil 4. 34. Veri setindeki örnekler; (a) Orijinal kusurlu ray görüntüleri, (b) Referans görüntüler

Unet eğitim sürecinde 24 katman kullanılmıştır. 192×256 boyutundaki girdi görüntüler döngü boyutu 100 ayarlanarak eğriltilmiştir. DCNN modeli 5 katmana sahiptir ve Unet modeline benzer aşamalardan geçmiştir. Önerilen modeller, AMD FX-8350 4.00 GHZ ve 32 GB belleğe sahip bir bilgisayarda gerçekleştirilmiştir. Unet modelinin eğitim ve test aşamaları Spyder ile Python 3.6 üzerinde yapılmıştır. DCNN modeli ise Matlab R2018b kullanılarak gerçekleştirilmiştir. Şekil 4.35'te DCNN modelinin tahmin edilen görüntüleri, orijinal gri tonlamalı görüntü ve referans görüntü ile karşılaştırılmıştır. Sonuçlar kabul edilebilir seviyededir. Şekil 4.35, raydaki kusurların tespit edildiğini göstermektedir.



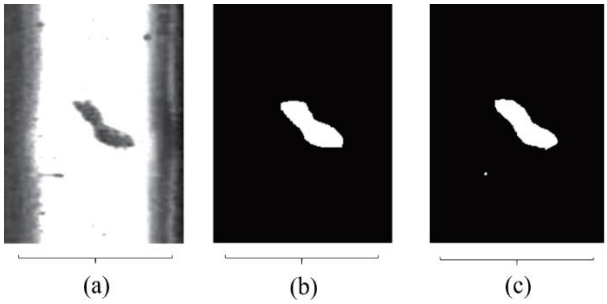

Şekil 4. 35. DCNN modelinin sonucu (a) Orijinal gri tonlamalı kusur, (b) Referans kusur (c) DCNN modelinin tahmini

Tablo 4.5, DCNN modelinin performans metriklerini göstermektedir. DCNN tabanlı kusur tespit deneyinde doğruluk oranı %96,23, IoU %64,29 ve F1 skoru %36,11 olarak bulunmuştur.

Tablo 4. 5. DCNN modelinin performans metrikleri

| Doğruluk | IoU (Jaccard) | F1 Score (Dice) |
|---|---|---|
| 0.9623 | 0.6429 | 0.3611 |

Şekil 4.36'da orijinal gri tonlamalı görüntü ile, referans görüntüler ve Unet modelinin tahmin edilen görüntüsü karşılaştırılmıştır. Şekil 4.36'da demiryolu üzerindeki kusurlar DCNN modeline göre daha iyi tespit edildiği görülmektedir. Şekil 4.36, ray üzerinde tespit edilen kusurları göstermektedir.



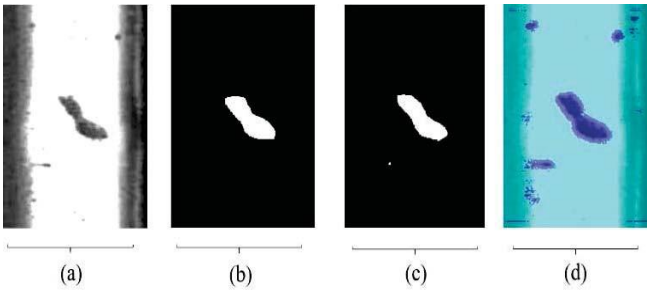

Şekil 4. 36. Unet modelinin sonucu (a) Orijinal gri tonlamalı kusur (b) Referans kusur (c) Unet modelinin tahmini

Tablo 4.6, Unet modelinin performans metriklerini göstermektedir. Unet tabanlı kusur tespit deneyinde doğruluk oranı %99,99, IoU %74,93 ve F1 skoru %42,83 olarak bulunmuştur.

Tablo 4. 6. Mayr Ark modelinde kullanılan bileşenler

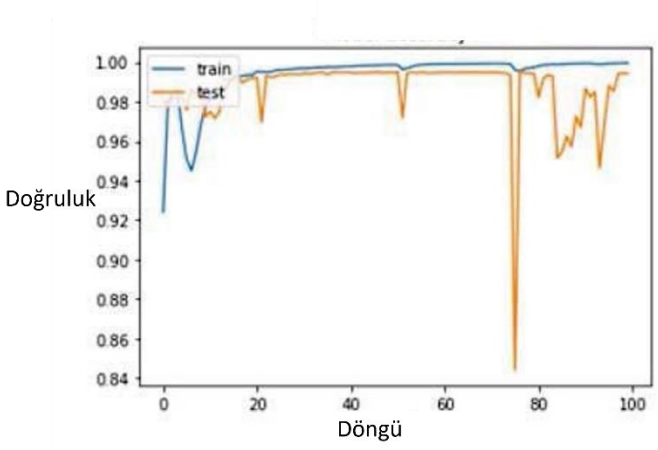
| Doğruluk Oranı | IoU | F1 Skoru |
|---|---|---|
| 0.9999 | 0.7493 | 0.4283 |

Önerilen modeller aynı veri seti üzerinde test edilmiştir. Önerilen modellerin, deneylerde ray üzerindeki kusurları tespit edebildiği ispatlanmıştır. Unet modeli, %99,99 doğruluk oranıyla DCNN modelinden daha iyi bir doğruluk oranı yakalamıştır. İki model arasındaki katman sayısı farkı bu sonuca neden olmuş olabilir. Şekil 4.37'de göre, her iki model tarafından yapılan tahminler, referans görüntüsü ile karşılaştırıldığında, Unet modeli DCNN modeline göre daha iyi bir bölütleme sonucu elde etmiştir.

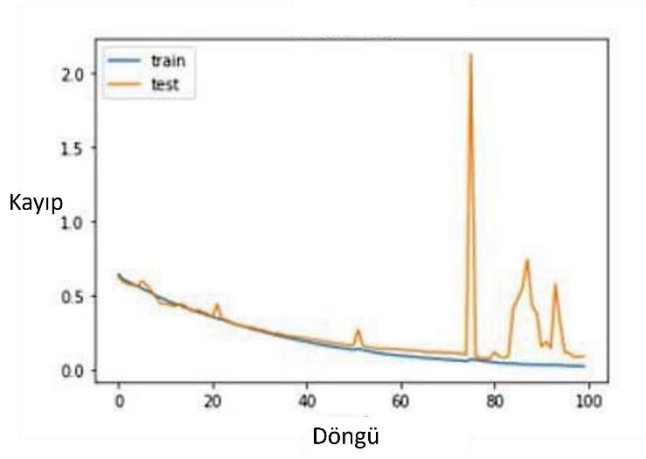



Şekil 4. 37. Önerilen modellerin sonuçlarının karşılaştırılması of (a) Orijinal gri tonlamalı kusur, (b) Referans kusur, (c) Unet modelinin tahmini (99.99% doğruluk oranı) ve (d) DCNN modelinin tahmini (96.23% doğruluk oranı)

Eğitim doğruluğu ve kayıp grafikleri Şekil 4.38 ve 4.39'da verilmiştir. Şekil 4.38 ve 4.39'de olduğu gibi, düşük kayıplı yüksek doğruluk, düşük hata anlamına gelmektedir. [87]. Doğruluk ve kayıp eğrilerini gösteren Unet eğitim grafikleri, kullanılan sınırlı demiryolu eğitim verisine rağmen modelin gereğinden fazla ezber yapmadığını göstermektedir. Şekil 4.38 ve 4.39, Unet modelinin %99,99 doğruluk oranını doğrulamaktadır.




Şekil 4. 38. Unet doğruluk grafiği




Şekil 4. 39. Unet kayıp grafiği

Önerilen modellerin performansı literatürde önerilen yaklaşımlarla karşılaştırılmıştır. Karşılaştırma sonuçları Tablo 4.7'de verilmiştir. Tablo 4.7'de görüldüğü gibi Unet modeli, düşük örneklem büyüklüğüne rağmen literatürdeki diğer çalışmalardan daha iyi sonuçlar vermiştir. Unet modeli %99,99 doğruluk oranına sahiptir.

Tablo 4. 7. Önerilen modellerin literatürdeki modellerle karşılaştırılması

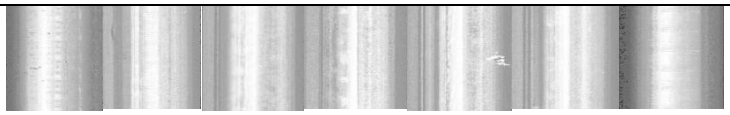
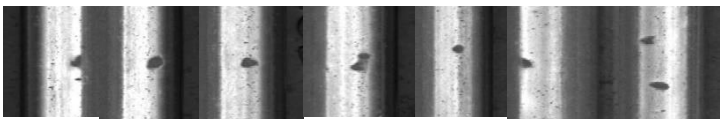
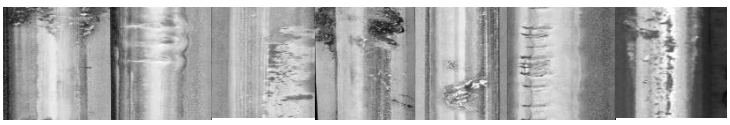
| Referans | Görüntü Sayısı | Model | Döngü Sayısı | Doğruluk Oranı |
|---|---|---|---|---|
| [88] | 22408 | DCNN | 40 | 92% |
| [89] | 17412 | CNN | - | 91.19% |
| [90] | 4101 | Unet | 50 | 80.53% |
| [90] | 4101 | Deeplabv3+ | 50 | 70.51% |
| [90] | 4101 | MCnet | 50 | 85.28% |
| Önerilen | 603 | Unet | 100 | 99.99% |

Önerilen modellerde toplam 603 ray görüntüsü incelendi. Unet modelinin doğruluk oranı, DCNN'ye göre daha iyidir. Unet neredeyse mükemmel bir bölütleme sonucu elde edildi ve Tablo 4.7'te görülmektedir. Önerilen model, ray kusurlarını geleneksel yöntemlerden daha verimli hale getirerek demiryolu güvenliğinin sağlanmasına katkıda bulunmuştur. Ayrıca Unet ve DCNN modellerinin raydaki kusurları tespit etmede başarılı olduğu çalışmada kanıtlanmıştır. Bu makalede demiryollarındaki aksaklıkların tespit edilmesinin demiryolu güvenliği açısından önemi vurgulanmıştır. Demiryolunun kullanımı her geçen gün artacağından kusur sorunu da her geçen gün artacaktır. Gelecekteki çalışmalarda, anlamsal bölütlemenin yanına örnek bölütlemede eklenerek kusur türleri belirlenebilir.

### 4.3. Sürdürülebilir Taşımacılıkta Demiryolu Hatları için Derin Özelliklere Dayalı Kusur Sınıflandırması İçin Deneysel Sonuçlar

Önerilen yöntemin performansını değerlendirmek için Demiryolları Araştırma ve Teknoloji Merkezi (DATEM) tarafından ölçüm treni ile elde edilen görüntüler kullanılmıştır. Ölçüm treninin Ankara-Konya ve Ankara-Eskişehir hatlarında yaptığı seyahatlerden görüntüler elde edildi. Görüntü elde etmek için 2096 piksel çözünürlüğe sahip bir DalsaSpyder3 çizgi tarama kamerası kullanıldı. Kameranın hat hızı saniyede 68.000 satırdır ve saatte 200 km hızla seyahat ederken 1 mm x 0.15 mm çözünürlük sağlamıştır. Elde edilen görüntülerden sağlıklı ve farklı yüzey kusurları kullanılarak bir veri seti oluşturulmuştur. Veri setinden bazı örnekler Tablo 4.8'de verilmiştir.

Tablo 4. 8. Demiryolu yüzey hatası veri seti

| Etiket | Örnek | Örnek Sayısı |
|---|---|---|
| Sağlıklı (Healthy) |  | 492 |
| Hafif Ezilme (Squat) |  | 608 |
| Eklem Yeri (Joint) |  | 408 |
| Şiddetli Ezilme (SSquat) |  | 330 |

Önerilen yöntemin doğruluğunu ve etkinliğini ölçmek için bazı performans ölçütleri kullanılmıştır. Kesinlik, geri çağırma, F1 ve doğruluk oranı gibi metrikler dört temel ölçüm kullanılarak elde edildi: Gerçek Pozitif (TP), Gerçek Negatif (TN), Yanlış Pozitif (FP) ve Yanlış Negatif (FN). TP değeri, tam veri seti içinde doğru sınıflandırılmış örneklerin sayısını gösterir. FP değeri, gerçek sınıfı dışında bir sınıf tarafından tahmin edilen örnek sayısıdır. TN değeri, sınıfı negatif olan ve negatif olarak tahmin edilen örnek sayısıdır. FN, gerçek sınıfı negatif olan ve pozitif olarak tahmin edilen örneklerdir. Kesinlik değeri (P), doğru tahmin edilen numunelerin toplam numune sayısına oranıdır. Ne kadar doğru tahmin edildiğinin bir ölçüsüdür ve 0 ile 1

arasında bir değer alır ve mümkün olduğu kadar yüksek olması gerekir. Geri çağırma değeri (R), pozitif örneklerin ne kadar başarılı tahmin edildiğini temsil eder ve test edilen verilerin doğruluğunun bir ölçüsüdür. F1, kesinlik ve geri çağırma metriklerinin harmonik ortalamasıdır. Doğruluk (A), P, R ve F1 için performans metrikleri aşağıdaki gibi tanımlanır.

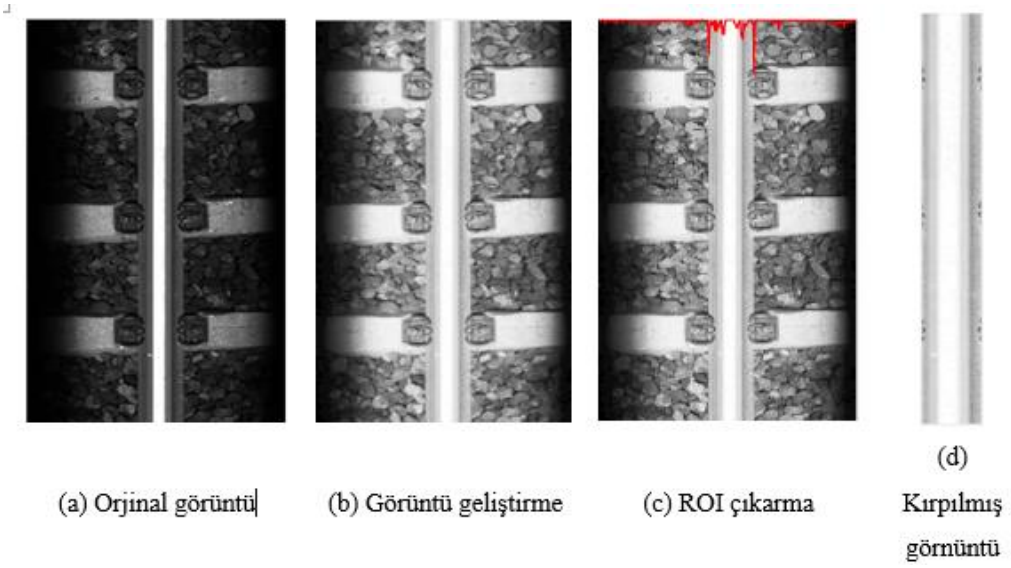
$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$P = \frac{TP}{TP + FP} \tag{4.2}$$

$$R = \frac{TP}{TP + FN} \tag{4.3}$$

$$F1 = 2x\frac{PXR}{P + R} \tag{4.4}$$

Görüntü işleme ve derin öğrenme araç kutuları kullanılarak MATLAB ortamında görüntü işleme ve hata algılama algoritmaları uygulanmıştır. Programları çalıştırmak için bir iş istasyonu bilgisayarı (CPU: Intel i7, GPU: GTX1080, 8GB, RAM: 16GB) kullanılmıştır. Ray yüzey kusurlarını tespit etmek için önce görüntüden ilgilenilen bölgenin kırpılması ve ardından veri setinin oluşturulması gerekmektedir. Önerilen yöntem, önce bir kontrast iyileştirme yöntemi uygulayarak yüksek kontrastlı bir görüntü elde eder. Daha sonra bu görüntüden ilgilenilen bölgeyi çıkarmak için histogram tabanlı teknik uygulanmıştır. Şekil 4.40'ta, ray görüntüsünden hat çıkarımı adım adım gösterilmektedir.




Şekil 4. 40. Parça konumu çıkarma

Şekil 4.40.a'da gösterilen orijinal pist görüntüsünde, aydınlatma nedeniyle bazı kısımlar daha parlak, diğer kısımlar ise daha koyu görünmektedir. Şekil 4. 40.b, bu sorunların görüntü iyileştirmeden sonra çözüldüğünü göstermektedir. ROI çıkarma adımında, Şekil 4. 40.c'de elde edilen sinyalden iki maksimum değer bulunur. Bu seçilen bölgeye göre kırpma rayı kısmı Şekil 4. 40.d'de gösterilmektedir. Önerilen ray çıkarma yöntemi literatürdeki yöntemlerle karşılaştırılmıştır. Bu amaçla, karşılaştırma için veri setinden rastgele 200 görüntü seçilmiştir. Algoritma, klasördeki görüntüleri otomatik olarak okur, ray bölgesini çıkarır ve kırpılan görüntüleri farklı bir klasöre yazar. Yöntemin tespit doğruluğu, doğru ve yanlış tespit edilen izlerin sayısı olarak hesaplanmıştır. Bu amaca yönelik karşılaştırma sonuçları Tablo 4.9'da verilmiştir.

Tablo 4. 9. Önerilen ROI çıkarma yönteminin diğer yöntemlerle karşılaştırılması

| **Referans** | **Ön işleme** | **# doğru algılanan parça** | **# yanlış algılanan parça** | **Doğruluk oranı (%)** |
|---|---|---|---|---|
| [91] | Histogram eşitleme | 170 | 30 | 85.00 |
| [92] | Michelson benzeri kontrast geliştirme | 180 | 25 | 87.50 |
| [93] | Medyan filtresi ve segmentasyon | 185 | 15 | 92.50 |

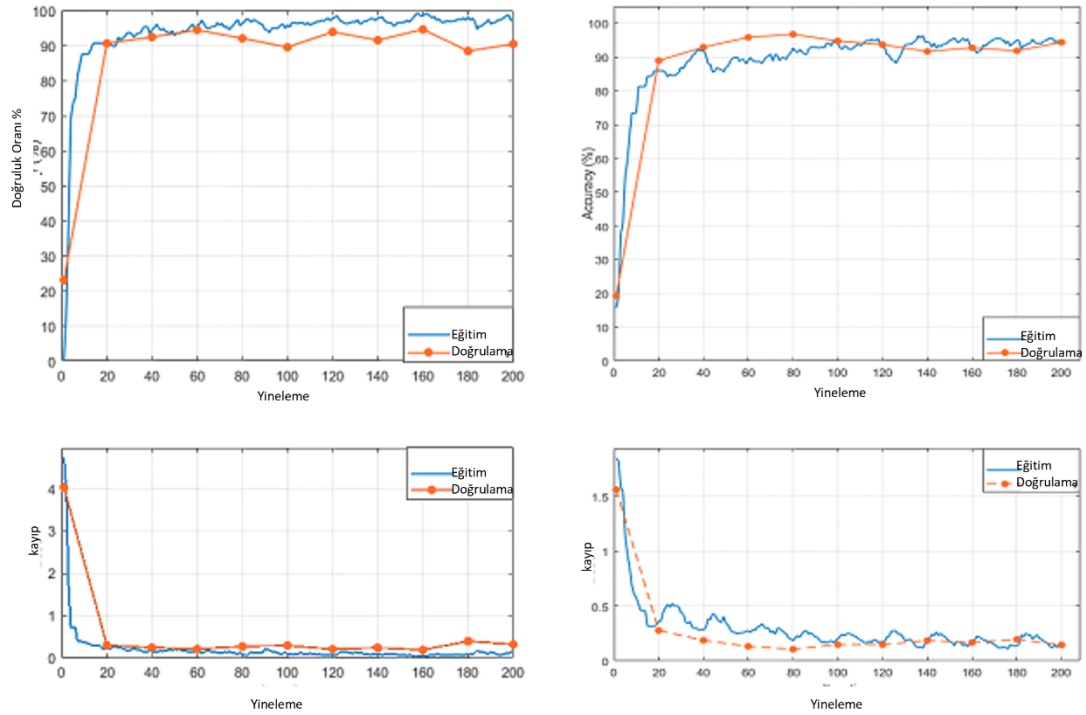
| | | | | |
|---|---|---|---|---|
| Önerilen yöntem | Yüksek kontrastlı görüntü | 195 | 5 | 97.50 |

Tablo 4.9'da verilen sonuçlara göre önerilen yöntemin diğer yöntemlere göre daha iyi sonuçlar verdiği görülmektedir. [91] ve [92]'de sadece rayın başlangıç noktası belirlenmiş ve rayın genişliği algoritmaya parametre olarak verilmelidir. [93]'de, ön işleme adımında sadece 3x3 boyutlu medyan filtreleri kullanıldığından aydınlatma ile ilgili problemler çözülememiştir. Orijinal görüntüden iz bölgesi elde edildikten sonra veri seti oluşturulmalı ve model sınıflandırma için eğitilmelidir. İlk oluşturulan veri seti, transfer öğrenme kullanılarak SqueezeNet ve MobileNetV2 gibi önceden eğitilmiş derin öğrenme modelleri ile eğitilmiştir. Derin öğrenme modellerinin eğitim parametreleri Tablo 4.10'da verilmiştir.

Tablo 4. 10. Squeezenet ve Mobilenetv2'nin eğitim parametreleri

| **Parametre** | **Değer** |
|---|---|
| Input image size | 224x224 |
| Optimization | sgdm |
| Momentum | 0.9 |
| Learning rate | 1e-4 |
| Mini Batch size | 64 |
| Max epoch | 10 |

Tablo 4.10'da verilen eğitim parametreleri belirlendikten sonra, oluşturulan derin öğrenme modelleri verilen veri seti ile eğitilmiştir. Eğitim aşamasında, veri setinin %70'i eğitimde, %30'u doğrulama için kullanılmıştır. Oluşturulan derin öğrenme modeli, 200 yineleme adımında çalıştı. Şekil 4.41, her bir derin öğrenme modeli için eğitim ve doğrulama başarılarının ilerlemesini göstermektedir.




(a) SqueezeNet'in Doğruluk ve Kayıp Grafiği (b) MobileNetv2'nin Doğruluk ve Kayıp Grafiği

Şekil 4. 41. Derin Öğrenme modellerinin eğitim ilerlemesi

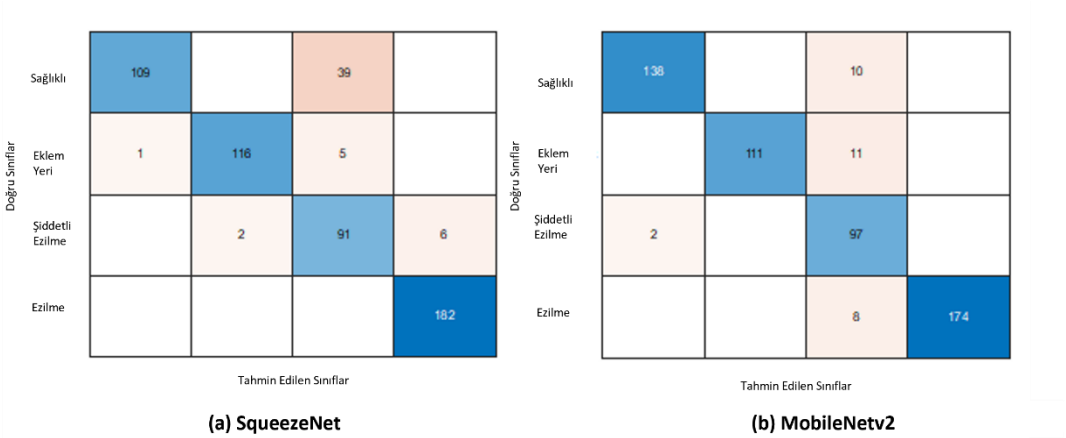
Şekil 4.41.a'da görüldüğü gibi SqueezeNet derin öğrenme modelinin eğitim performansı yüksek olmasına rağmen doğrulama performansı düşüktür. Şekil 4.41.b'de eğitim ve doğrulama performansları birbirine daha yakın olarak elde edilmiştir. İki derin öğrenme modelinin performans ve hesaplama süreleri Tablo 4.11'de verilmiştir.

Tablo 4. 11. Squeezenet ve Mobilenetv2'nin eğitim sonuçları

| **Sınıflandırma Doğruluğu** | | | **Hesaplama Süresi** | |
|---|---|---|---|---|
| **Model** | **Eğitim** | **Doğrulama** | **Eğitim** | **Test** |
| SqueezeNet | 97.44 | 90.38 | 93 s | 21 ms |
| MobileNetv2 | 96.04 | 94.37 | 67 s | 15 ms |

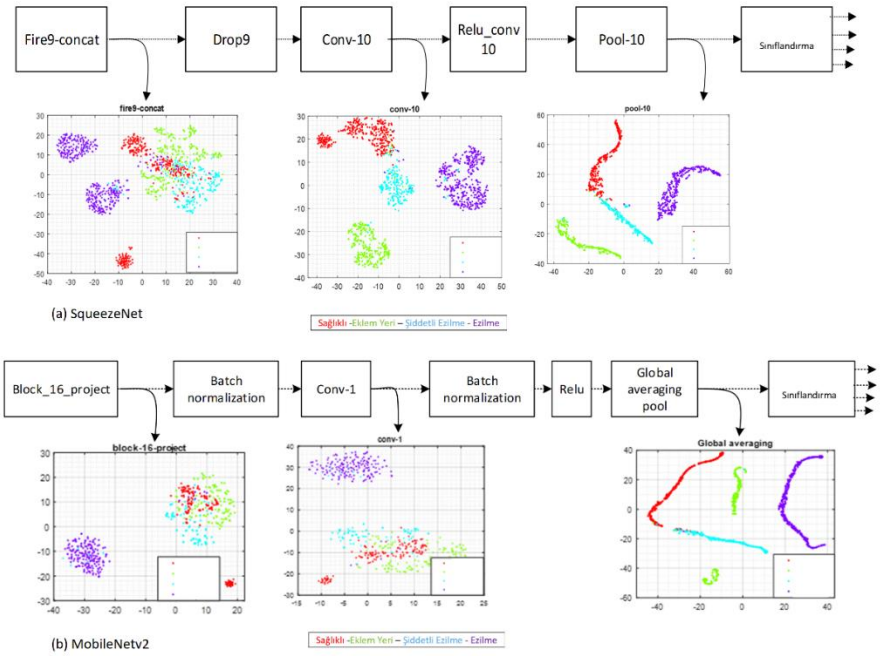
Tablo 4.11'de görüldüğü gibi iki modelin eğitim doğrulukları arasındaki fark düşük olmasına rağmen doğrulama veri setinde MobileNetV2'nin doğruluk oranı daha yüksektir. Ayrıca MobileNetV2 işlemi hesaplama süresi açısından daha kısa sürede tamamlanmıştır. Şekil 4.42,

her kategorinin sınıflandırma oranını açıkça göstermek için iki modelin karışıklık matrislerini temsil etmektedir.



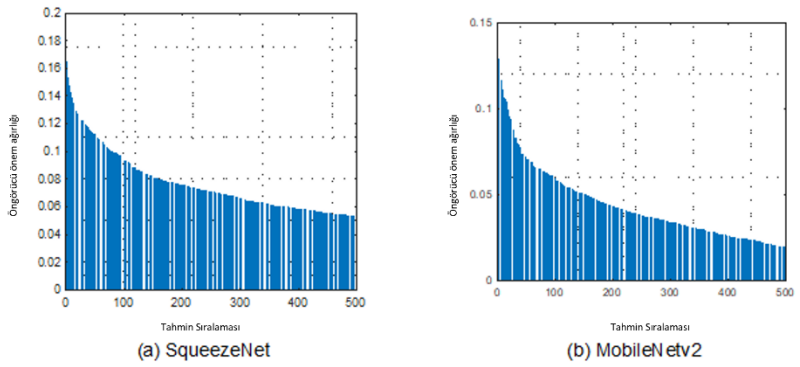

Şekil 4. 42. Doğrulama Veri kümesi için iki derin öğrenme modelinin karışıklık matrisi

Şekil 4.42'de, doğrulama süreci için MobileNetV2 tabanlı yöntemin SqueezeNet'ten daha iyi sınıflandırma doğruluğu elde ettiği karışıklık matrisinden görülebilir. SqueezeNet modelinde SSquat ve sağlıklı (healthy) vakaların karıştırıldığı görülmektedir. Bu çalışmada önerilen model, her bir derin sinir ağından öznitelikler çıkarır ve öznitelik indirgeme işleminden sonra bu öznitelikleri DVM'ler ile sınıflandırır. Şekil 4.43, farklı katmanlarda elde edilen her modelin özelliklerini göstermektedir.



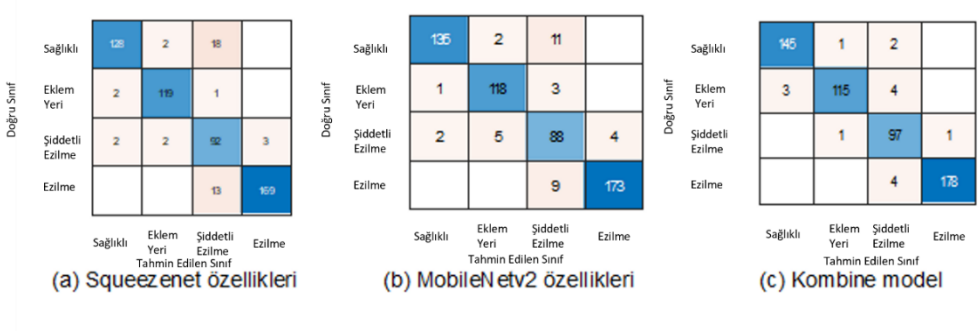

Şekil 4. 43. Farklı katmanlar için her bir derin öğrenme modelinin çıkarılan özelliklerinin görselleştirilmesi

SqueezeNet'in pool10 katmanından ve MobileNetV2'nin global ortalama havuzundan elde edilen özellikler, özellik ağırlıklarını hesaplayan ReliefF algoritmasına verilir. Şekil 4.44'te her bir derin öğrenme modeli için elde edilen 500 öznitelik için ReliefF algoritması ile özniteliklerin önem sırası verilmiştir.




Şekil 4. 44. Her bir derin öğrenme modelinden elde edilen özelliklerin önem sırası

Şekil 4.44'te her bir derin öğrenme modeli için yüksek önem taşıyan ilk 500 özelliğin ağırlıkları verilmiştir. Özellik çıkarımı sırasında SqueezeNet ve MobileNetV2'de sırasıyla 1.000 ve 1.287 öznitelik elde edilmiştir. Deneyler sonucunda 500 öznitelik seçiminin yeterli olduğu görülmüştür. Seçilen öznitelik sayısı azaldıkça sınıflandırma performansının düştüğü ancak 500 özniteliğin üzerindeki değerlerde performansın değişmediği görülmüştür. ReliefF algoritmasının en yakın komşu sayısı 10 olarak seçilmiştir. Şekil 4.45.a ve 4.45.b'de sırasıyla DVM'ler ile SqueezeNet ve MobileNetV2 derin öğrenme modellerinden elde edilen özniteliklerin sınıflandırılmasında elde edilen karışıklık matrisleri verilmiştir. Şekil 4.45.c'de, iki modelin öznitelikleri kaynaştırılarak ReliefF algoritması ile öznitelik seçimi yapılmaktadır. Daha sonra bu öznitelikler DVM'ler ile sınıflandırılarak elde edilen karışıklık matrisi doğrulama veri seti için verilmiştir.




Şekil 4. 45. Özellik çıkarma ve DVM'ler kullanılarak her modelin karışıklık matrisleri

Şekil 4.45’teki karışıklık matrisleri incelendiğinde, öznitelik birleştirmenin tek bir model kullanmaktan daha iyi bir performansa sahip olduğu gösterilmektedir. Hem SqueezeNet hem de MobileNetV2'de SSquat sınıfının sağlıklı (healthy) sınıf örnekleriyle karıştırıldığı ve sınıflandırma performansının düşük olduğu görülmektedir. Önerilen yöntemin performansı, doğruluk, kesinlik, hatırlama ve F1 metriklerine göre diğer derin öğrenme modelleri ile karşılaştırılmıştır. Karşılaştırma sonuçları Tablo 4.12'de verilmiştir.

Tablo 4. 12. Farklı derin öğrenme modellerinin performans karşılaştırma sonuçları

| **Model** | **Sınıf** | **Katman** | **Özellikler** | **P** | **R** | **F1** | **A (%)** |
|---|---|---|---|---|---|---|---|
| SqueezeNet | Sağlıklı | Pool10 | 1000 | 0.96 | 0.86 | 0.91 | 92.45 |
| | Eklem Yeri | | | 0.96 | 0.97 | 0.97 | |
| | Şiddetli Ezilme | | | 0.74 | 0.92 | 0.82 | |
| | Ezilme | | | 0.98 | 0.92 | 0.95 | |
| MobileNetv2 | Sağlıklı | Global average pool | 1287 | 0.97 | 0.91 | 0.94 | 93.28 |
| | Eklem Yeri | | | 0.94 | 0.96 | 0.95 | |
| | Şiddetli Ezilme | | | 0.79 | 0.88 | 0.83 | |
| | Ezilme | | | 0.97 | 0.95 | 0.96 | |
| GoogleNet | Sağlıklı | Pool5 drop | 1024 | 0.92 | 0.97 | 0.95 | 94.90 |
| | Eklem Yeri | | | 0.97 | 0.95 | 0.96 | |
| | Şiddetli Ezilme | | | 0.91 | 0.82 | 0.86 | |
| | Ezilme | | | 0.97 | 0.98 | 0.98 | |
| VGG19 | Sağlıklı | Drop7 | 4096 | 0.93 | 0.96 | 0.95 | 95.30 |
| | Eklem Yeri | | | 0.96 | 0.95 | 0.96 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Şiddetli Ezilme | | | 0.90 | 0.86 | 0.88 | |
| | Ezilme | | | 0.97 | 0.98 | 0.98 | |
| Resnet50 | Sağlıklı | Avg-pool | 2048 | 0.94 | 0.97 | 0.96 | 95.40 |
| | Eklem Yeri | | | 0.95 | 0.97 | 0.96 | |
| | Şiddetli Ezilme | | | 0.94 | 0.83 | 0.88 | |
| | Ezilme | | | 0.96 | 0.98 | 0.97 | |
| Önerilen Model | Sağlıklı | Birleştirilmiş iki model | 1000 | 0.97 | 0.97 | 0.97 | 97.10 |
| | Joint | | | 0.98 | 0.94 | 0.96 | |
| | Şiddetli Ezilme | | | 0.90 | 0.97 | 0.94 | |
| | Ezilme | | | 0.99 | 0.97 | 0.98 | |

Tablo 4.12'de önerilen yaklaşımın farklı derin öğrenme modelleri ile karşılaştırma sonuçları dört metriğe göre verilmiştir. Ayrıca öznitelik çıkarma katmanı ve ilgili katmandaki öznitelik sayısı verilmiştir. Özellikle SSquats (şiddetli ezilme) sınıfında diğer yöntemlerin performans ölçütlerinin düşük olduğu gözlemlenmiştir. Yöntemimizin tüm performans metrikleri %90'ın üzerindeydi. Ayrıca test aşamasında daha az özellik kullanılmış ve diğer derin öğrenme modellerine göre daha iyi performans elde edilmiştir. Ray yüzey hatalarının belirlenmesine yönelik literatürde var olan yaklaşımlar genellikle hata olup olmadığının belirlenmesine dayanmaktadır. Bu nedenle literatürdeki yöntemler genellikle ray yüzeyini sağlıklı ve kusurlu olmak üzere iki sınıfa ayırmaktadır. Sadece bir çalışmada, lazer tabanlı üç boyutlu bir yaklaşımla kusur tipleri belirlendi [94]. Son olarak, sunulan derin öğrenme tabanlı bütünleşmiş yöntem ile literatürde mevcut yöntemler arasındaki hata sınıflandırma performanslarını karşılaştırarak önerilen yöntemin literatüre katkısını belirlemeyi amaçladık. Karşılaştırma sonuçları Tablo 4.13'te verilmiştir.

Tablo 4. 13. Performans sonuçlarının diğer yöntemlerle karşılaştırılması

| **Referans** | **Kullanılan Metod** | **Sınıf Sayısı** | **Doğruluk Oranı (%)** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| [95] | Gauss karışım modelleri ve eğrilik filtresi tabanlı bölütleme | 2 | 95.64 |
| [96] | Vurgulanan maksimum entropi tabanlı segmentasyon | 2 | 90.07 |
| [96] | Kabadan inceye model ve otsu tabanlı eşikleme | 2 | 88.53 |
| [97] | Kenar algılama ve kontur tabanlı bölge büyütme | 2 | 92.03 |
| [98] | Görüntü işleme ve derin öğrenme tabanlı çok modelli öğrenme | 2 | 94.70 |
| [94] | Geometrik özellikler ve sinir ağları tabanlı sınıflandırıcı | 6 | 93.30 |
| Bizim metodumuz | İki derin öğrenme yönteminin ve SVM'nin birleşik özellikleri | 4 | 97.10 |

Tablo 4.13'teki sonuçlar, iki derin öğrenme modeliyle elde edilen öznitelikleri birleştirerek, yöntemimizin diğer ray hatası algılama yöntemlerinden daha iyi sınıflandırma doğruluğu sağladığını kanıtlamaktadır. [95]'deki yöntem önce görüntüyü gri eşitleme ile iyileştirir ve ilgilenilen bölgeyi çıkarır. Daha sonra gürültü bir eğrilik filtresi ile azaltılır ve görüntü Gauss karışım modeli ile bölütlenir. [96]'da kontrast iyileştirmesinden sonra ray yüzey görüntülerine entropi tabanlı bir yöntem uygulanarak kusurlar belirlenir. [96]'da, görüntü iyileştirildikten sonra, görüntü satır satır işlenerek olası kusur noktaları belirlenir. [97]'de, aydınlatma ile ilgili sorunları azaltmak ve ray hatalarını yüksek doğrulukla tespit etmek için parçalı kenar çıkarma tabanlı bir yöntem önerilmiştir. Ancak, o çalışmada önerilen yöntemin farklı türdeki kusurları tespit etmede yetersiz kaldığı belirtilmiştir. [98]'de önerilen çalışma, farklı ray yüzeylerinde test edilmiş, görüntü işleme ve derin öğrenmeye dayalı çoklu hata tespit yöntemi önermektedir. Öncelikle bölütleme ile hata alanı belirlenmiş ve derin öğrenme ile hata tespiti yapılmıştır. Bununla birlikte, önerilen yöntemin hem karmaşık olması hem de kusur tipini belirlememesi gibi dezavantajları vardır. Literatürdeki çalışmaların çoğu sadece ray yüzeyindeki hata tespitine odaklanmıştır. Bu nedenle, kusurun türünü belirlemek için hemen hemen hiçbir çalışma yapılmamıştır. Önerilen çalışmalar genellikle segmentasyona dayalı kusurlar ile ray

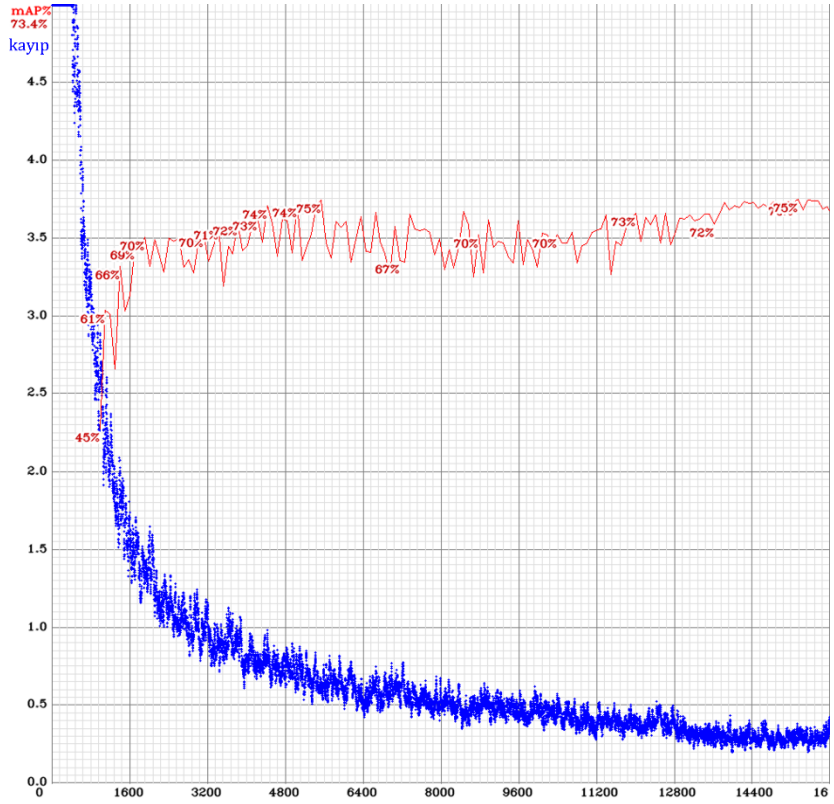
yüzeyi arasında ayrım yapmıştır. Geleneksel bölütleme yöntemleri, aydınlatma problemlerinden ve ray yüzeyindeki kir ve toz gibi gürültüden oldukça etkilenmektedir, bu nedenle bu problemlerin üstesinden gelmek için çeşitli yöntemler önerilmiştir. Ancak, bu yöntemlerin kusur tespit performansı düşük kalmıştır. [94]'de önerilen yaklaşım, lazer kamera kullanarak 3 boyutlu bir görüntü elde etmekte, nokta bulutu oluşturmakta ve bulut noktasından elde edilen geometrik özellikleri yapay sinir ağı ile sınıflandırmaktadır. Ancak 3 boyutlu görüntünün elde edilmesi ve gerçek zamanlı olarak işlenmesi çok zaman almaktadır. Bu nedenle deneyler laboratuvar ortamında gerçekleştirilmiştir. Literatürdeki diğer yöntemlerden farklı olarak önerilen yöntemimiz, ray alanını belirledikten sonra bölütleme yerine derin sinir ağları aracılığıyla çıkarılan özniteliklere göre kusur türlerini belirlemektedir. Böylece ray yüzeyi daha iyi modellendiğinden daha yüksek performans elde edilmiştir.

Bu çalışma, ray yüzeyi kusurlarının tespiti ve sınıflandırılması ile ilgilidir. Önerilen yöntem, görüntü işleme, çoklu derin öğrenme modellerinden özellik çıkarma, özellik birleştirme ve DVM'lerle sınıflandırmadan oluşmaktadır. Önerilen görüntü işleme tabanlı yaklaşım, orijinal görüntüden yüksek kontrastlı bir görüntü elde eder ve ray yüzeyinin bölgesi kırpılır. Önerilen kontrast iyileştirme yaklaşımı, aydınlatma nedeniyle ray yüzeyinde oluşan sorunları ortadan kaldırmakta ve ROI çıkarma yönteminin performansını artırmaktadır. Demiryolu hattı elde edildikten sonra, yüzey kusurlarını sınıflandırmak için özellik çıkarma, çoklu derin öğrenme ağlarından özellik birleştirme, özellik seçimi ve DVM'ler ile sınıflandırma uygulanmıştır. Öznitelik çıkarımı için SqueezeNet ve MobileNetV2 modelleri, öznitelik seçimi için ReliefF algoritması kullanılmıştır. Önerilen yaklaşım, literatürde mevcut bildirilen yöntemlerden daha iyi olan %97,10'luk bir sınıflandırma başarısı elde etmiştir. Yazarların bildiği kadarıyla, bu yöntem, ray yüzey kusurlarını tespit etmek için derin öğrenme modellerini kullanan ilk yöntemdi. Çalışmada önerilen yöntem, Türkiye Cumhuriyeti devlet demiryollarının Ankara-Konya ve Ankara-Eskişehir hatlarında toplanan görüntülerle sağlanmıştır. Elde edilen görüntüler pratik uygulamalar için bazı alternatifler sunmaktadır. Bu yöntemler, bu projedeki çalışmalarımızın bir parçası olarak yakın gelecekte saha testlerinde kullanılacaktır.

## 4.4. YOLO ile Ray Üzerindeki Kusurların Tespiti

### 4.4.1. YOLOv4 Sonuçları

Çalışmada eğitimini gerçekleştirdiğimiz YOLOv4 modeli 0.7338 mAP.5 değeri yakalamıştır. Ayrıca 0.73 kesinlik, 0.69 geri çağırma oranı elde etmiştir. YOLOv4 modelinin kayıp grafiği Şekil 4.46’da görülmektedir.



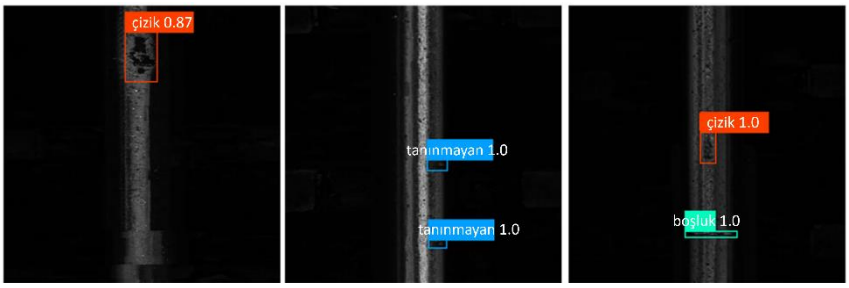

Şekil 4. 46. YOLOv4 modelinin kayıp grafiği

YOLOv4 modelin eğitim sırasında sınıf bazında elde ettiği mAP.5 oranları Tablo 4.14'te görülmektedir.

Tablo 4. 14. YOLOv4 modelinin eğitim sonucunda sınıf bazında elde ettiği mAP değerleri

| **Kusur Çeşidi** | **mAP.5 Değeri** |
|---|---|
| crush | 0.93 |
| dent | 0.65 |
| scratch | 0.77 |
| slant | 0.68 |
| damage | 0.38 |
| dirt | 0.60 |
| gap | 1.00 |
| unknown | 0.83 |

Eğitim sonucunda elde edilen ağırlık ile test edilen ray kusuru içeren görüntüleri görsel ortama aktardığımızda Şekil 9'deki görüntüler elde edilmiştir. Şekil 4.47.a'da scratch kusurunun mAP değeri 0.87, Şekil 4.47.b'deki unknown kusurlarının 1.00 ve Şekil 4.47.c'deki scratch ve gap kusurlarının da 1.00 olarak elde edilmiştir.

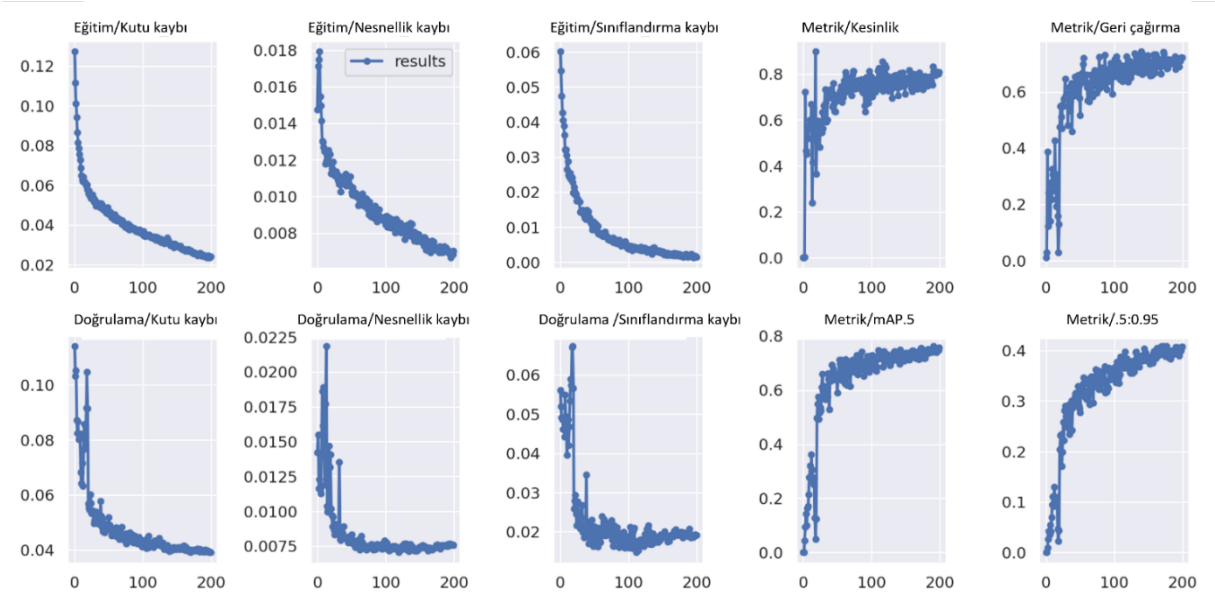



(a) (b) (c)

Şekil 4. 47. YOLOv4 modelinin test sonuçları

### 4.4.2. YOLOv5 Sonuçları

Çalışmada YOLOv5 modelinin alt mimarileri nesne tespiti için kullanılmıştır. Bunlar; YOLOv5s, YOLOv5n, YOLOv5m, YOLOv5l'dir. Şekil 4.48'de YOLOv5s alt mimarisine ait eğitim ve doğrulama grafikleri görülmektedir.




Şekil 4. 48. YOLOv5s mimarisinin eğitim ve doğrulama sonuçları

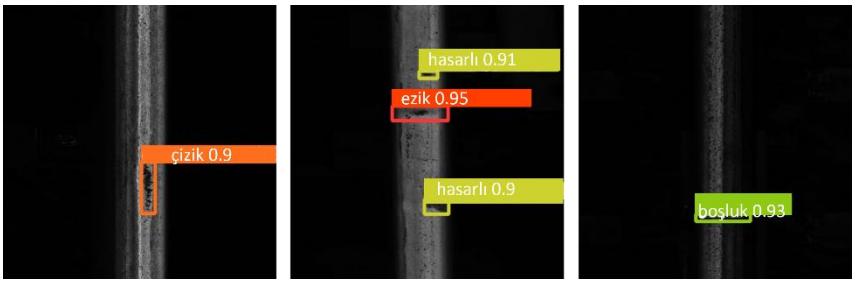
YOLOv5s alt mimarisi eğitimi sonucunda 0.7555 mAP.5 değeri yakalamıştır. Ayrıca 0.744 kesinlik, 0.742 geri çağırma oranı elde etmiştir. Tablo 4.15'te eğitim sonucu sınıf bazında elde edilen mAP.5 değerleri görülmektedir.

Tablo 4. 15. YOLOv5s alt mimarisini eğitim sonucunda sınıf bazında elde ettiği mAP.5 değerleri

| **Kusur Çeşidi** | **mAP.5 Değeri** |
|---|---|
| crush | 0.912 |

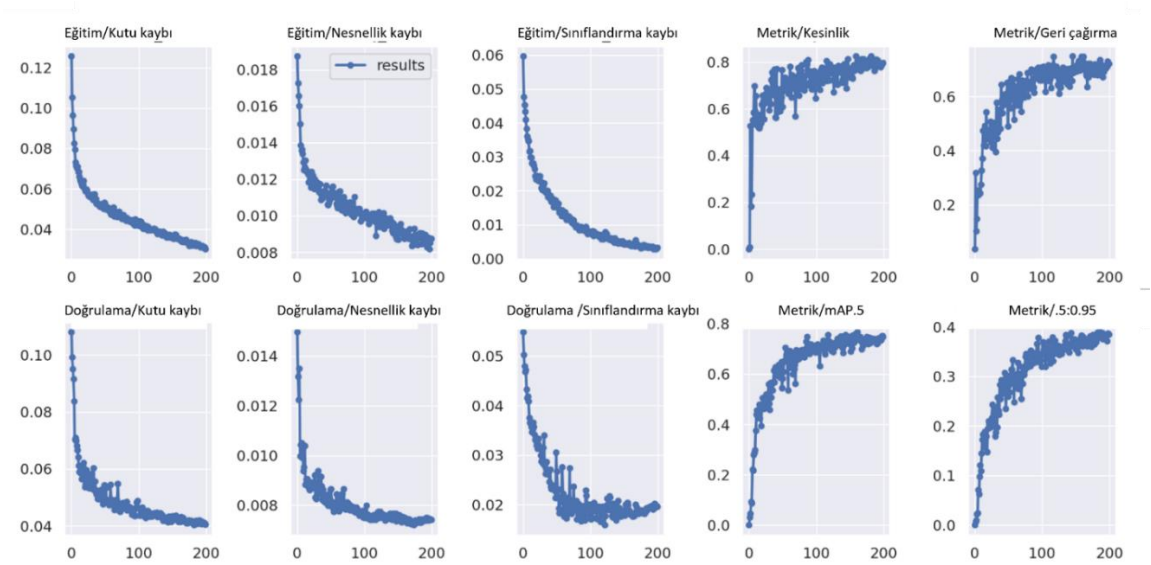
| | |
|---|---|
| dent | 0.688 |
| scratch | 0.832 |
| slant | 0.728 |
| damage | 0.349 |
| dirt | 0.712 |
| gap | 0.995 |
| unknown | 0.821 |

Eğitim sonucunda elde edilen ağırlık ile test edilen ray kusuru içeren görüntüleri görsel ortama aktardığımızda Şekil 4.49'daki görüntüler elde edilmiştir.



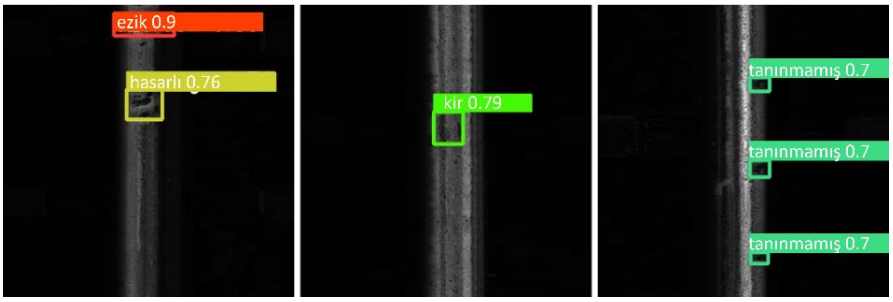

Şekil 4. 49. YOLOv5s alt mimarisinin test sonuçları

Şekil 4.50'de YOLOv5n alt mimarisine ait eğitim ve doğrulama grafikleri görülmektedir.




Şekil 4. 50. YOLOv5n alt mimarisinin eğitim ve doğrulama sonuçları

YOLOv5n alt mimarisinin eğitimi sonucunda elde edilen ağırlık ile test edilen ray kusuru içeren görüntüleri görsel ortama aktardığımızda Şekil 4.51'teki görüntüler elde edilmiştir.



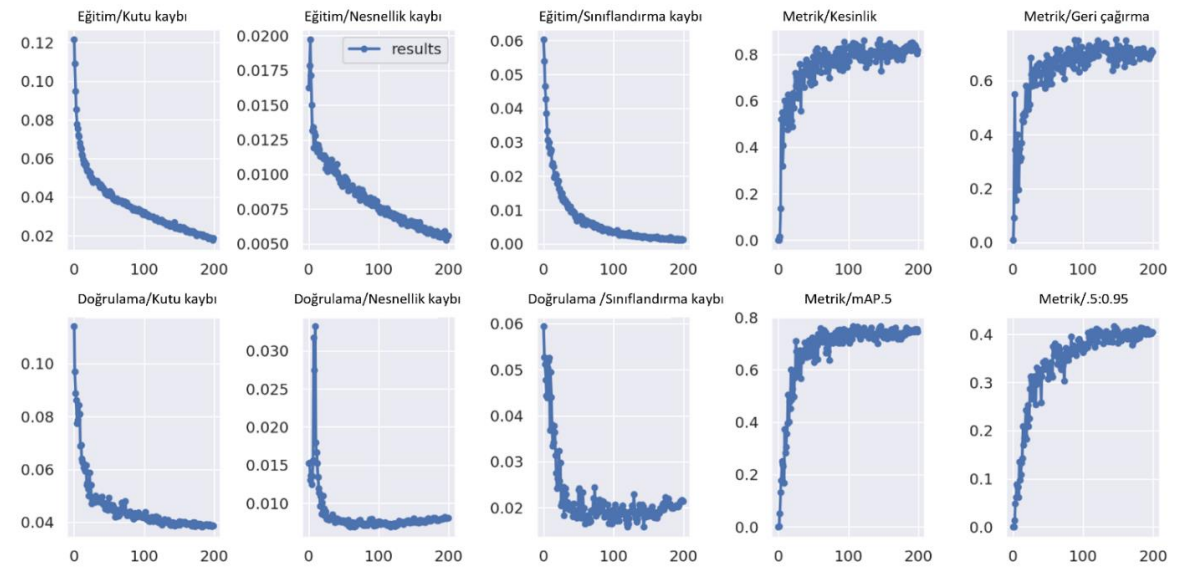

Şekil 4. 51. YOLOv5n alt mimarisinin test sonuçları

YOLOv5n alt mimarisi eğitimi sonucunda 0.755 mAP.5 değeri yakalamıştır. Ayrıca 0.794 kesinlik, 0.704 geri çağırma oranı elde etmiştir. Tablo 4.16'da eğitim sonucu sınıf bazında elde edilen mAP.5 değerleri görülmektedir.

Tablo 4. 16. YOLOv5n alt mimarisini eğitim sonucunda sınıf bazında elde ettiği mAP.5 değerleri

| **Kusur Çeşidi** | **mAP.5 Değeri** |
|---|---|
| crush | 0.869 |
| dent | 0.797 |
| scratch | 0.786 |
| slant | 0.842 |
| damage | 0.253 |
| dirt | 0.718 |
| gap | 0.977 |
| unknown | 0.799 |

Şekil 4.52'de YOLOv5m alt mimarisine ait eğitim ve doğrulama grafikleri görülmektedir.




Şekil 4. 52. YOLOv5m mimarisinin eğitim ve doğrulama sonuçları

YOLOv5m alt mimarisi eğitimi sonucunda 0.752 mAP.5 değeri yakalamıştır. Ayrıca 0.812 kesinlik, 0.68 geri çağırma oranı elde etmiştir. Tablo 4.17'de eğitim sonucu sınıf bazında elde edilen mAP.5 değerleri görülmektedir.

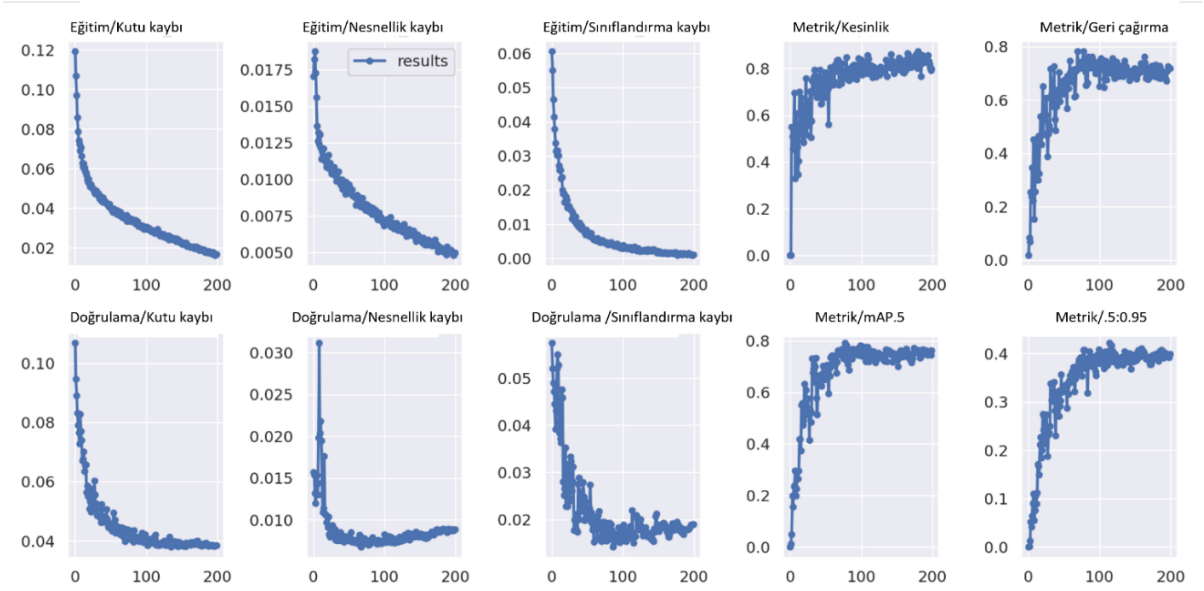
Tablo 4. 17. YOLOv5m alt mimarisinin eğitim sonucunda sınıf bazında elde ettiği mAP.5 değerleri

| **Kusur Çeşidi** | **mAP.5 Değeri** |
|---|---|
| crush | 0.969 |
| dent | 0.722 |
| scratch | 0.829 |
| slant | 0.681 |
| damage | 0.325 |
| dirt | 0.712 |
| gap | 0.99 |
| unknown | 0.786 |

YOLOv5m alt mimarisinin eğitimi sonucunda elde edilen ağırlık ile test edilen ray kusuru içeren görüntüleri görsel ortama aktardığımızda Şekil 4.53'teki görüntüler elde edilmiştir.


Şekil 4. 53. YOLOv5m alt mimarisinin test sonuçları

Şekil 4.54'te YOLOv5l alt mimarisine ait eğitim ve doğrulama grafikleri görülmektedir.




Şekil 4. 54. YOLOv5l mimarisinin eğitim ve doğrulama sonuçları

YOLOv5l alt mimarisi eğitimi sonucunda 0.759 mAP.5 değeri yakalamıştır. Ayrıca 0.813 kesinlik, 0.723 geri çağırma oranı elde etmiştir. Tablo 4.18'de eğitim sonucu sınıf bazında elde edilen mAP.5 değerleri görülmektedir.

Tablo 4. 18. YOLOv5m alt mimarisini eğitim sonucunda sınıf bazında elde ettiği mAP.5 değerleri

| **Kusur Çeşidi** | **mAP.5 Değeri** |
|---|---|
| crush | 0.922 |
| dent | 0.794 |

| | |
|---|---|
| scratch | 0.8 |
| slant | 0.73 |
| damage | 0.327 |
| dirt | 0.719 |
| gap | 0.979 |
| unknown | 0.805 |

YOLOv5l alt mimarisinin eğitimi sonucunda elde edilen ağırlık ile test edilen ray kusuru içeren görüntüleri görsel ortama aktardığımızda Şekil 4.55'teki görüntüler elde edilmiştir.


Şekil 4. 55. YOLOv5l alt mimarisinin test sonuçları

Tablo 4.19'da YOLOv5 alt mimarilerinin doğrulama performansları ve topluluk modelleme ile edilen doğruluk performansları görülmektedir. Topluluk modelleme eğitim sonucu elde edilen ağırlıkların doğrulma performanslarının mAP.5 değerlerini yukarıya çektiği görülmektedir.

Tablo 4. 19. YOLOv5 modelleri ile topluluk modelleme yönteminin doğrulama performansları

| **Model** | **Kesinlik** | **Geri Çağırma** | **mAP. 5** |
|---|---|---|---|
| YOLOv5n | 0.794 | 0.704 | 0.756 |
| YOLOv5s | 0.744 | 0.742 | 0.753 |
| YOLOv5m | 0.812 | 0.68 | 0.752 |
| YOLOv5l | 0.875 | 0.716 | 0.783 |
| YOLOv5n+ YOLOv5s+ YOLOv5m+ YOLOv5l | 0.843 | 0.763 | 0.809 |
| YOLOv5s+ YOLOv5m+ YOLOv5l | 0.849 | 0.751 | 0.811 |

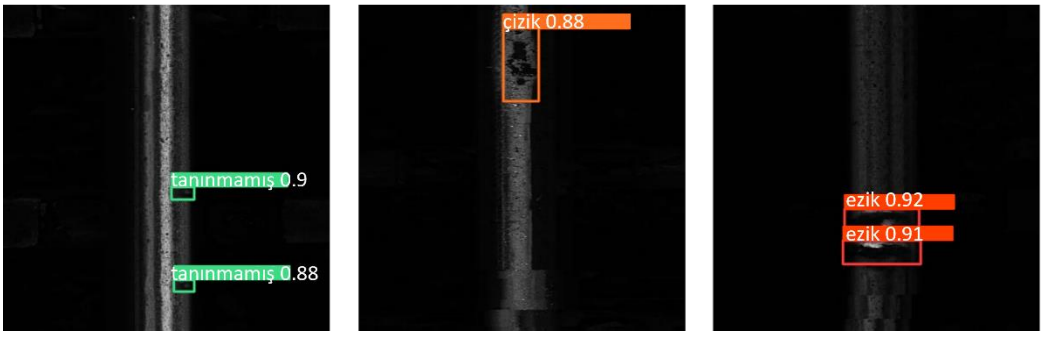
### 4.4.3. YOLOv6 Sonuçları

Çalışmada YOLOv6 modelinin alt mimarileri olan YOLOv6s, YOLOv6n ve YOLOv6t veri setini eğitmek için kullanılmıştır. Kullanılan alt YOLOv6 alt mimarilerinin eğitim sonuçları Tablo 4.20'de görülmektedir. Tablo 4.20'de görüldüğü gibi artan girdi boyutu ray üzerindeki

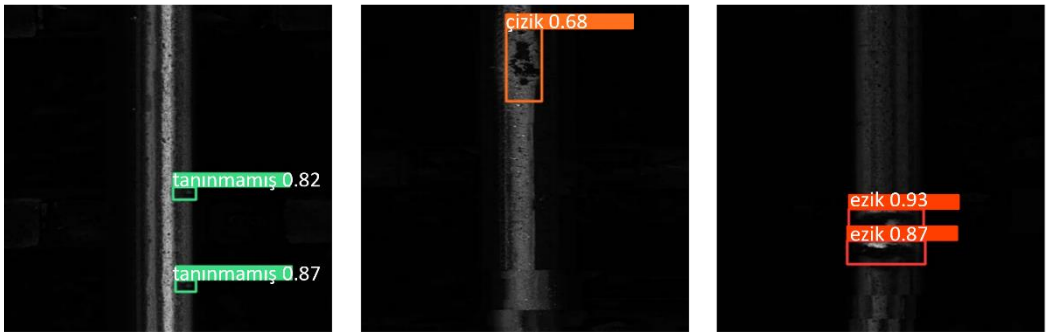
kusurların tespiti konusunda başarıyı arttırmıştır. mAP.5 değerindeki artış bunun ispatıdır. Şekil 4.56-61'te YOLOv6 alt mimarilerinin eğitim sonucu oluşturulan ağırlıkları kullanılarak elde edilen test sonuçları görülmektedir.

Tablo 4. 20. YOLOv6 alt mimarilerinin eğitim sonuçları

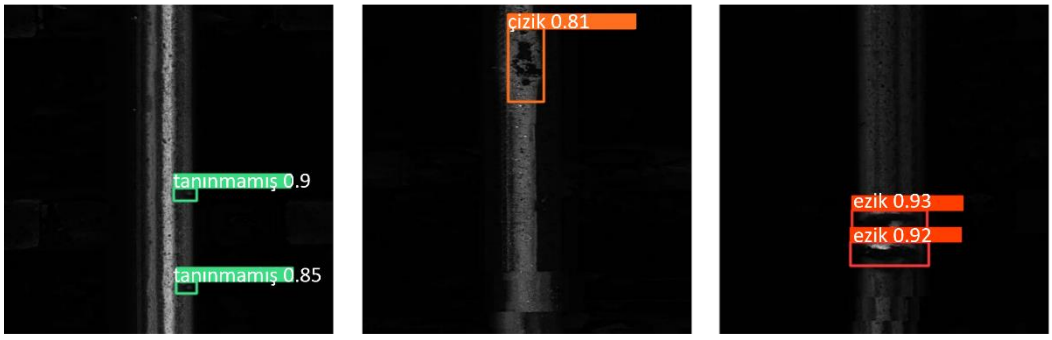
| **Model / Girdi Boyutu** | **mAP.5 Değeri** |
|---|---|
| YOLOv6s / 416 | 0.7074 |
| YOLOv6n / 416 | 0.7063 |
| YOLOv6t / 416 | 0.6858 |
| YOLOv6s / 640 | 0.7802 |
| YOLOv6n / 640 | 0.7473 |
| YOLOv6t / 640 | 0.7792 |



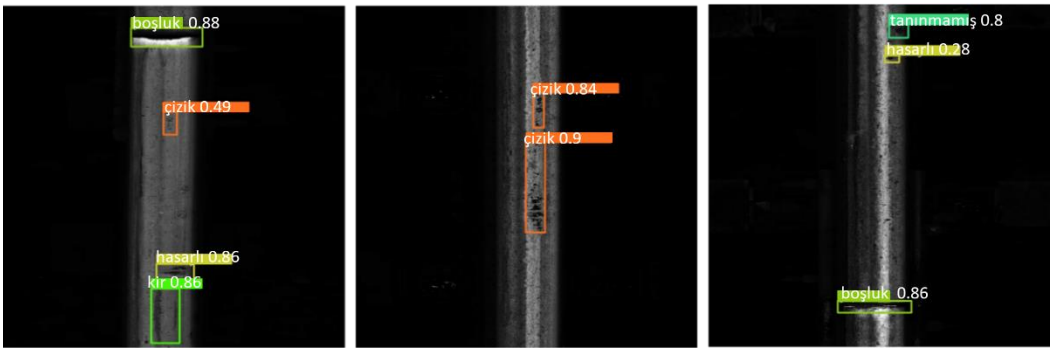

Şekil 4. 56. 416x416 girdi boyutlu YOLOv6s alt mimarisinin test sonuçları



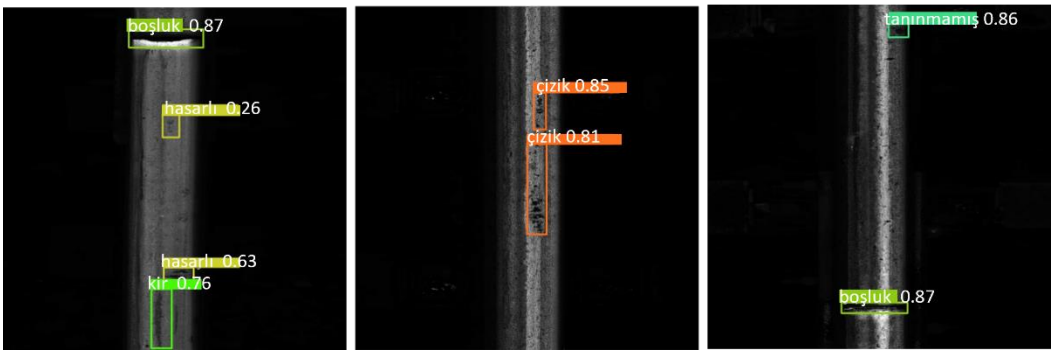

Şekil 4. 57. 416x416 girdi boyutlu YOLOv6n alt mimarisinin test sonuçları



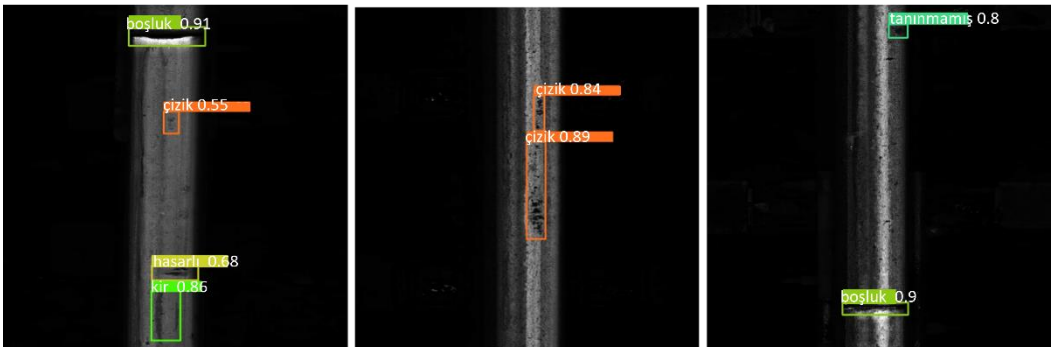

Şekil 4. 58. 416x416 girdi boyutlu YOLOv6t alt mimarisinin test sonuçları




Şekil 4. 59. 640x640 girdi boyutlu YOLOv6s alt mimarisinin test sonuçları




Şekil 4. 60. 640x640 girdi boyutlu YOLOv6n alt mimarisinin test sonuçları




Şekil 4. 61. 640x640 girdi boyutlu YOLOv6t alt mimarisinin test sonuçları

YOLOv4 modeli eğitim sonucunda 0.7338 mAP.5'lik oranı yakalamıştır. YOLOv5'in alt mimarileri YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l sırasıyla 0.7555, 0.755, 0.752, 0.752 mAP.5 oranı yakalamışlardır. YOLOv6'nın alt mimarisi olan YOLOv6s modeli girdi veri boyutunu attırdığımızda Tablo 4.53'te görüldüğü gibi 0.7802'lik mAP.5 oranı yakalamıştır. Çalışmada görüldüğü gibi YOLOv5 ve YOLOv6 modelleri YOLOv4'e göre daha başarılı sonuçlar vermişlerdir. İki modelin doğrulama performansları Tablo 4.21'deki gibidir. Doğrulama işlemi 90 adet görüntüyle yapılmıştır.

Tablo 4. 21. YOLOv5 modelleri ile topluluk modelleme yönteminin doğrulama performansları

| **Model** | **Girdi Boyutu** | **mAP.5** |
|---|---|---|
| YOLOv5n | 416 | 0.756 |
| YOLOv5s | 416 | 0.753 |
| YOLOv5m | 416 | 0.752 |
| YOLOv5l | 416 | 0.783 |
| YOLOv5n+ YOLOv5s+ YOLOv5m+ YOLOv5l | 416 | 0.809 |
| YOLOv5s+ YOLOv5m+ YOLOv5l | 416 | 0.811 |
| YOLOv6s | 416 | 0.734 |
| YOLOv6n | 416 | 0.702 |
| YOLOv6t | 416 | 0.704 |
| YOLOv6s | 640 | 0.770 |
| YOLOv6n | 640 | 0.738 |
| YOLOv6t | 640 | 0.785 |

Tablo 4.21'de görüldüğü gibi topluluk modelleme doğrulma YOLOv5 modelinde performansını arttırmıştır. YOLOv6 modelinde ise girdi veri boyutunu attırmak da doğrulama performansına pozitif etki yapmıştır. Buradaki dezavantaj ise eğitim süresinin artması olmuştur. Ortalama eğitim süresi 1.8 saatten 3.2 saate çıkmıştır. Çalışmanın bir diğer katkısı da literatürde demiryolu alandaki çalışımlarda YOLOv5 topluluk modelleme ve YOLOv6 yöntemleri fazla kullanılmamıştır. Tablo 4.22'de görüldüğü gibi önerilen topluluk modelleme yöntemi literatürdeki bazı yaklaşımlarla karşılaştırıldığında daha yüksek kesinlik ve geri çağırma değerleri elde etmiştir.

Tablo 4. 22. YOLOv5 topluluk modelleme yöntemi ile literatürdeki yaklaşımların performans karşılaştırması

| **Referans** | **Yaklaşım** | **Kesinlik** | **Geri Çağırma** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| [99] | BSM | 0.278 | 0.726 |
| [99] | CTFM | 0.841 | 0.732 |
| [99] | REM | 0.039 | 0.472 |
| Önerilen | YOLOv5n+ YOLOv5s+ YOLOv5m+ YOLOv5l | 0.843 | 0.763 |
| Önerilen | YOLOv5s+ YOLOv5m+ YOLOv5l | 0.849 | 0.751 |

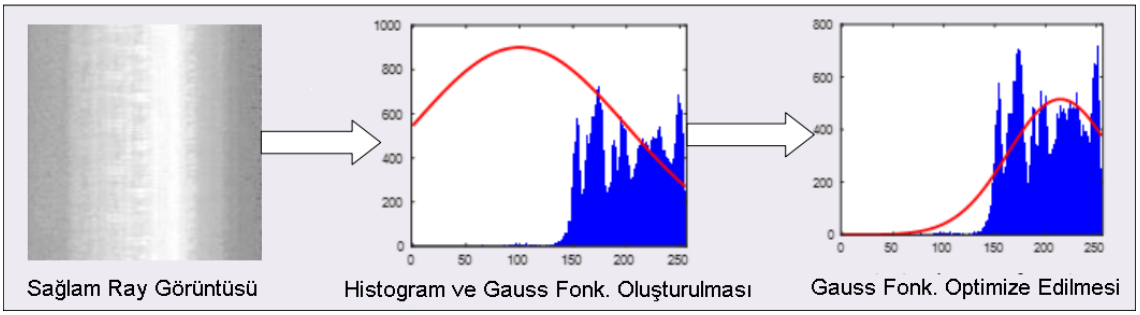
## 4.5. Bulanık Ölçüm ve Evrişimli Sinir Ağlarına Dayalı İki Aşamalı Ray Kusur Sınıflandırması İçin Deneysel Sonuçlar

Önerilen iki aşamalı hata tespit yöntemi, bir sağlıklı ve üç hata tipi içeren bir veri kümesine uygulanmıştır. Veri setinde eklem, hafif çömelme ve ray üzerinde şiddetli çömelme gibi kusurlar bulunmaktadır. Şekil 4.62, veri setindeki her sınıf için örnekleri temsil etmektedir.

(a) (b) (c) (d)

Şekil 4. 62. Her sınıf için sağlıklı ve hatalı görüntüler. (a) Sağlam, (b) Birleşim, (c) Hafif Çökme, (d) Şiddetli Çökme

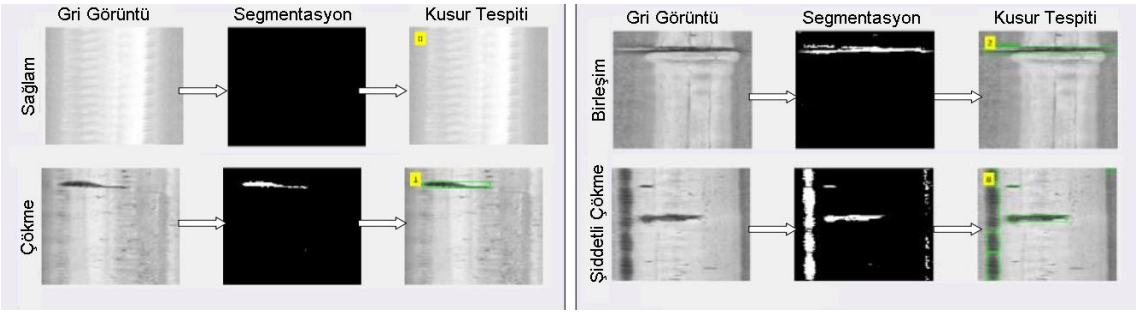
Şekil 4.62'de Sağlam, Birleşim, Hafif Çökme ve Şiddetli Çökme sınıfları sırasıyla 492, 408, 608 ve 330 örnekten oluşmaktadır. Bu nedenle veri seti 1838 örnekten oluşmaktadır. İlk olarak sağlam rayı modelleyen Gauss fonksiyonu sağlam bir numune alınarak elde edilir. Elde edilen Gauss fonksiyonu Şekil 4.63'te gösterilmiştir.




Şekil 4. 63. Histogram modelleme için optimize edici Gauss işlevi

Şekil 4.63'te, Gauss fonksiyonu önce rastgele parametrelerle başlatılır ve ardından Gauss fonksiyonunun ortalama ve standart sapma parametreleri optimize edilir. Elde edilen Gauss fonksiyonu, sağlıklı durum için bir bulanık üyelik fonksiyonu olarak belirlenir. Daha sonra kalan

histogram uzayında kusurlu durum için bir üyelik fonksiyonu oluşturulur. Şekil 4.64'te, sağlam ve kusurlu ray yüzeyleri için önerilen bulanık tabanlı segmentasyon (bölütleme) ve kusur tespit sonuçları verilmektedir.




Şekil 4. 64. Bulanık ölçüm tabanlı kusur tespiti

Şekil 4.64, her bir kusur için bir numunede kusurların nasıl tespit edildiğini gösterir. Tablo 4.23'te her sınıf için sağlıklı ve kusurlu bulunan örnek sayıları ve başarı oranları verilmiştir.

Tablo 4. 23. Kusur tespit performansı

| **Sınıf** | **Tahmin Sayısı** | | **Tespit Oranı (%)** |
|---|---|---|---|
| | **Sağlam** | **Kusurlu** | |
| Sağlam | 474 | 18 | 96.34 |
| Hafif Çökme | 1 | 607 | 99.83 |
| Birleşim | 2 | 406 | 99.50 |
| Şiddetli Çökme | 0 | 330 | 100.00 |
| Genel Tespit Oranı | | | 99.66 |

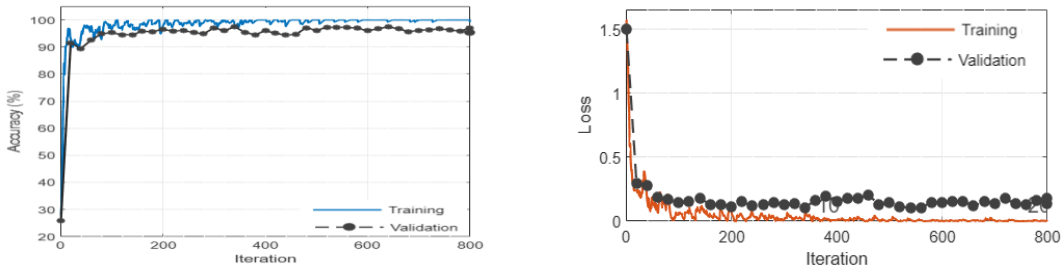
Tablo 4.23'te ortalama tespit oranı %99.66 idi. Veri kümesi farklı türde kusurlar içermektedir ve kusurlar için bir maske görüntüsü oluşturulmamıştır. Tespit oranı, yalnızca kusurlu alanların varlığına veya yokluğuna dayanıyordu. Yüzeyde bir kusur tespit edilirse, kusurun türünü belirlemek için derin öğrenme etkinleştirilir. Bu amaçla MobileNetv2 kullanılmıştır. Kullanılan ağın parametreleri, eğitim için kullanılan parametreler olan Tablo 4.24'te verilmiştir.

Tablo 4. 24. MobileNetv2'de kullanılan parametreler

| **Parametre** | **Değer** |
|---|---|
| Öğrenme Oranı | 0.001 |
| Eğitim Sayısı (Epoch) | 20 |

| | |
|---|---|
| Yığın Boyutu (Batch) | 32 |
| Doğrulama Frekansı | 20 |

Tablo 4. 24'te verilen parametrelere göre MobileNetv2 ağı eğitilir ve arızalar belirlenir. Veri seti %70 eğitim ve %30 doğrulama olarak ayrılmıştır. Eğitim ve doğrulama için doğruluk ve kayıp grafikleri Şekil 4.65'te verilmiştir.




(a) (b)

Şekil 4. 65. MobileNetV2 için eğitim ve hata grafikleri (a) Eğitim grafiği, (b) Hata (kayıp) grafiği

Şekil 4.65'te %95,46 doğrulama başarısı elde edilmiştir. Önerilen yöntemin performansı literatürde önerilen diğer yöntemlerle karşılaştırılmış ve daha iyi sonuçlar elde edilmiştir. Karşılaştırma sonuçları Tablo 4.25'te verilmiştir.

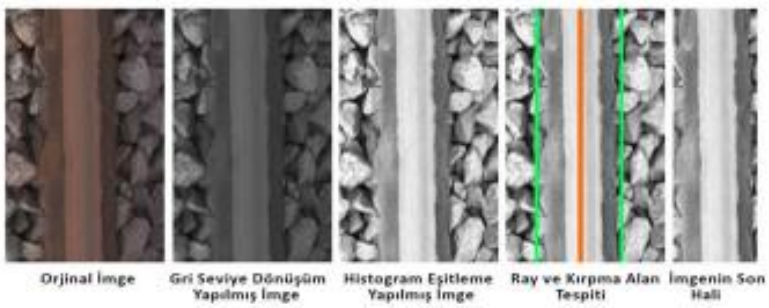
Tablo 4. 25. Önerilen yöntemin diğer çalışmalarla karşılaştırma sonuçları

| **Referans** | **Method** | **# Sınıflar** | **Doğruluk Oranı (%)** |
|---|---|---|---|
| [100] | Graph cut and faster RCNN | 2 | 95.34 |
| [101] | Image features+Deep feature extraction | 4 | 93.80 |
| [102] | Deep extractor | 2 | 93.04 |
| [103] | CNN+LSTM | 2 | 94.27 |
| Yöntemimiz | Bulanık ölçüm tabanlı algılama | 4 | 99.66 |
| | MobileNetV2 tabanlı sınıflandırma | | 95.46 |

## 4.6. İki Derin Öğrenme Yöntemiyle Ray Yüzeyi Kusurlarının Tespiti: Karşılaştırmalı Analizi İçin Deneysel Sonuçlar

Drone ile alınan ray görüntülerinden kusurların tespit edilebilmesi için ilk olarak ön işleme yapılması ve ray alanının görüntüden çıkarılması gerekmektedir. Ray dışındaki bölgelerin

piksel değerleri, çakıllar ve arasında kalan gölgeler nedeniyle çok daha fazla değişkenlik göstermektedir. Bu durum, söz konusu bölgelerin üzerinden geçen kolon içerisindeki piksel değerlerinin standart sapmasının yüksek çıkmasına sebep olmaktadır. En küçük standart sapma değerine sahip kolon, rayın olduğu bölgededir. Bu kolonun solundan ve sağından, imge 120px genişliğinde kalacak şekilde kırpma işlemi yapılmıştır. Süreç aşamaları Şekil 4.66'da görülmektedir.



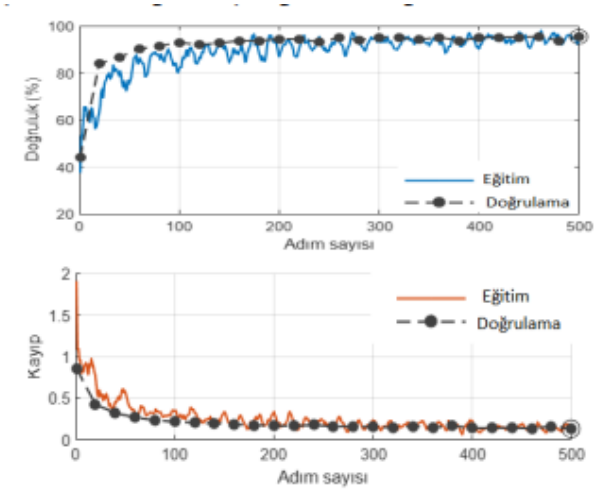

Şekil 4. 66. Ön işleme aşamaları

Ön işleme adımından sonra ESA yöntemleri ile rayın kusurlu olup olmadığı belirlenecektir. Bu amaçla ilk olarak eğitim parametrelerinin belirlenmesi gerekir. Tablo 4.26'da eğitim için kullanılan parametreler verilmiştir.

Tablo 4. 26. Önerilen yöntemin diğer çalışmalarla karşılaştırma sonuçları

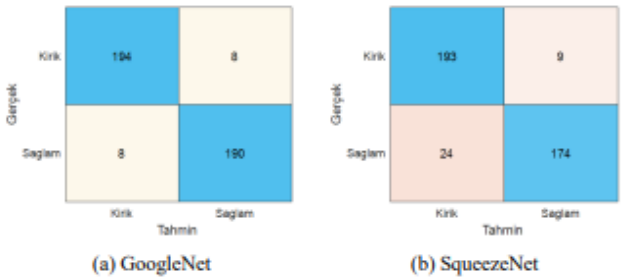
| **Parametre** | **Değer** |
|---|---|
| Yığın boyutu | 32 |
| Maksimum adım sayısı | 10 |
| Öğrenme oranı | 1e-4 |
| Doğrulama frekansı | 20 |

Eğitim parametreleri belirlendikten sonra ağ eğitilmiştir. Şekil 4.67'de GoogleNet için eğitim süreci görülmektedir.




Şekil 4. 67. GoogleNet eğitim süreci

Şekil 4.67'de GoogleNet doğrulama başarımı %96 olarak elde edilmiştir. Aynı parametreler ile SquuezeNet modeli de eğitilmiştir. Eğitim sonucunda doğrulama için elde edilen karmaşıklık matrisleri Şekil 4.68'de verilmiştir.




Şekil 4. 68. Karmaşıklık matrisler

Şekil 4.68'de elde edilen karmaşıklık matrisleri GoogleNet'in daha iyi sonuçlar verdiğini göstermektedir. Karmaşıklık matrislerinden elde edilen metrikler Tablo 4.27'de verilmiştir.

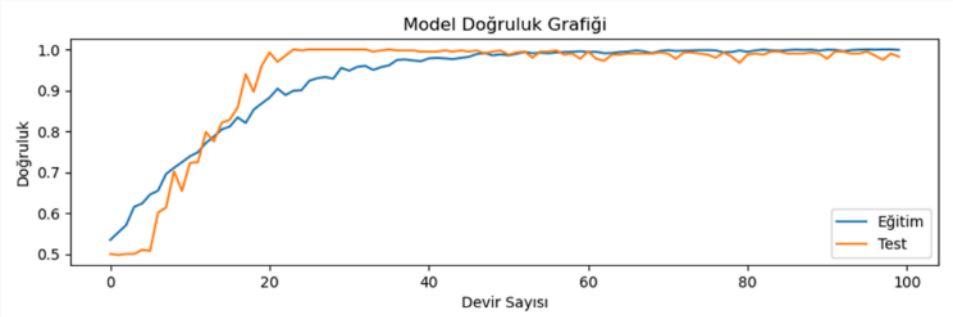
Tablo 4. 27. Farklı metrikler için sınıflandırma başarımları

| Metrik | GoogleNet | SqueezeNet |
|---|---|---|
| Doğruluk | 96.00 | 91.75 |
| Geri çağırma | 96.04 | 88.94 |
| Kesinlik | 96.04 | 95.54 |
| F1 | 96.04 | 92.12 |

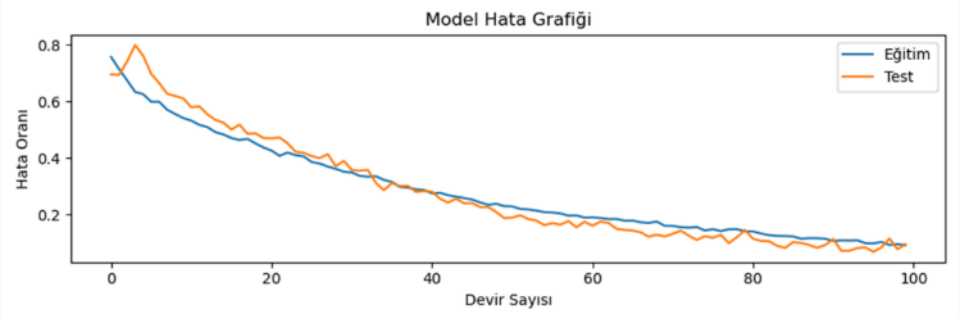
Tablo 4.27'de elde edilen sonuçlar değerlendirildiğinde Googlenet'in hem doğruluk hem de diğer metriklerde daha iyi sonuçlar verdiği görülmektedir.

## 4.7. DCGAN ve Siyam Sinir Ağını Kullanarak Demiryolu Bağlantı Elemanlarındaki Kusurların Tespiti

Siyam sinir ağlarının kullanılma amacı, bağlantı elemanı görüntülerinin deforme ve sağlam olarak sınıflandırılmasıdır. Yapılan uygulamalar, Python 3.8 de derlenmiştir. Windows işletim sistemi kullanılmıştır. Derin öğrenme uygulamaları, TensorFlow çerçevesinde uygulanmıştır. Siyam sinir ağı modelini eğitmek için NVIDIA GeForce GTX 1060 kullanılmıştır. Optimizasyon yöntemi olarak Adam Optimizer kullanılmıştır. Batch değeri 32 olarak seçilip öğrenme hızı 0,0001 olarak belirlenmiştir. Devir sayısı 100 olarak ayarlanmıştır. Eğitim işlemi sonucunda oluşan eğitim ve test görüntülerinin doğruluk grafikleri Şekil 4.69 ve Şekil 4.70'te verilmiştir.



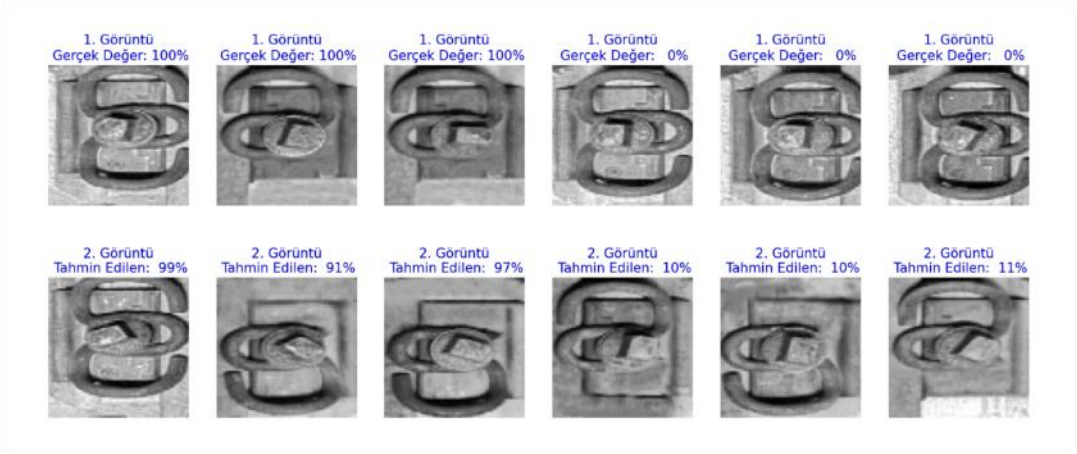

Şekil 4. 69. Deney sonucunda oluşan model doğruluk grafiği




Şekil 4. 70. Deney sonucunda oluşan model hata grafiği

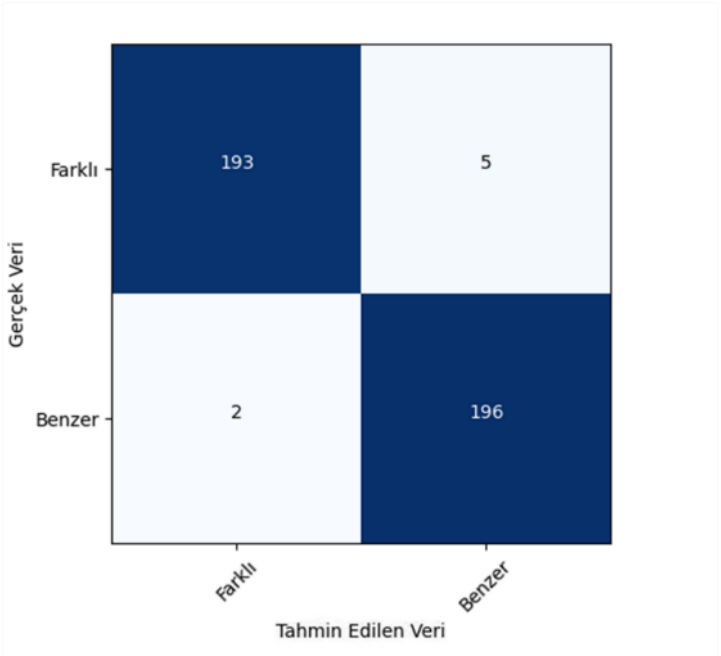
Eğitim işleminden sonra elde edilen model, iki görüntü arasındaki benzerliği tahmin etmek için kullanılmıştır. Ağın son katmanında Sigmoid aktivasyon fonksiyonu kullanıldığından 0 ile 1 aralığında bir çıktısı oluşacaktır. Sonucun 0 olması, iki görüntünün birbirinden tamamen farklı olduğu anlamına gelirken sonucun 1 olması ise iki görüntünün tamamen benzer olduğu anlamına gelmektedir. Örnek olarak, deforme ile sağlam görüntüler karşılaştırılırsa elde edilen sonuç 0'a yakınken iki adet sağlam bağlantı elemanını görüntüsü karşılaştırıldığında sonuç 1'e yakın olacaktır. Elde edilen sonuç 100 ile çarpılarak yüzde cinsinden değer elde edilmiştir. Veri kümesinden rastgele örnekler seçilip karşılaştırma yapılmıştır. Sonuçlar, Şekil 4.71'de

verilmiştir. Karşılaştırmada else edilen sonuç %50'nin altında ise görüntü farklı, %50'nin üzerinde ise görüntüler aynı olarak belirlenmiştir. Rastgele yapılan eşleştirme sayısı 396'dır. Karşılaştırma sonuçlarından elde edilen karmaşıklık matrisi Şekil 4.72'de verilmiştir.




Şekil 4. 71. Eğitilen siyam ağına göre iki görüntünün benzerlik oranına göre karşılaştırılması

Karmaşıklık matrisi kullanılarak doğruluk, duyarlılık, özgünlük ve F1 değerlerini ölçülmüştür.




Şekil 4. 72. Test görüntülerinin eşleştirilmesi sonucu oluşan karmaşıklık matrisi

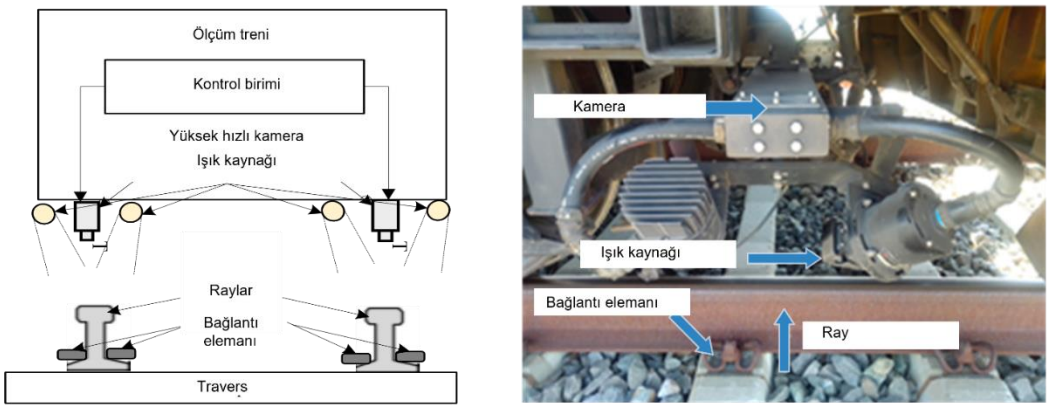
Tablo 4. 28. Önerilen yöntemin performans sonuçları

| Doğruluk (%) | Duyarlılık (%) | Kesinlik (%) | F1 (%) |
|---|---|---|---|
| 98,23 | 98,98 | 97,51 | 98,25 |

Önerilen yöntem ile elde edilen ölçüm sonuçları Tablo 4.28'de verilmiştir. Elde edilen deneysel sonuçlar, önerilen yöntemin yüksek doğruluk elde ettiğini göstermektedir. Ayrıca, kusur tespit hassasiyetinin yüksek olması önerilen yöntemin, bağlantı elemanlarında bulunan küçük kusurların da tespiti için kullanılabileceğini göstermektedir. Veri kümesi 2'den fazla sınıf olacak şekilde genişletilirse farklı kusur türlerinin tespitinde de kullanılabilir. Bu çalışmanın ana hedefi demiryolunun yüksek doğruluk oranlı bir denetim yöntemiyle denetlenmesi ve işgücü maliyetinin azaltılmasıdır.

## 4.8. Görüntü Ön İşleme ve Hafif Evrişimsel Sinir Ağı Kullanarak Demiryolu Bağlantı Elemanlarının Kusur Sınıflandırması İçin Deneysel Sonuçlar

Bir bağlantı elemanı kusur tespit sistemi, yüksek hızlı bir görüntü elde etme sistemi, bir görüntü işleme ve tanıma sistemi, bir veri yönetim sistemi ve bir merkezi kontrol sisteminden oluşur. Görüntü toplama sistemi, sürekli görüntü toplamak için bir ölçüm dizisinin altına yerleştirilmiş yüksek hızlı kameralar ve aydınlatma cihazlarından oluşur. Toplanan görüntüler merkezi kontrol sistemine iletilir ve GPS bilgileri ile kayıt altına alınır. Bu çalışmada, bir ölçüm trenine monte edilen Türkiye Cumhuriyeti Devlet Demiryolları Araştırma Merkezi (DATEM) deney sistemi kullanılarak farklı zamanlarda gerekli verileri elde ettik. Şekil 4.73, verilerin elde edildiği ölçüm sistemini temsil etmektedir.




a) Görüntü edinme sistemi

b) Yüksek hızlı kameraların ve ışık kaynağının ekipmanı

Şekil 4. 73. Ölçüm sisteminin yapısı

Şekil 4.73'te gösterildiği gibi, bağlantı elemanı görüntüleri, ölçüm dizisinin altına dikey bir açıyla yerleştirilmiş iki hatlı sahne kameraları ile toplanır. Sağ ve sol ışın görüntüleri ayrı kameralarla toplanır. Dış ortamdaki ışık farklılıklarını ortadan kaldırmak için her kamera iki LED ışıkla donatılmıştır. Kamera çözünürlüğü 2096 piksel, kameranın hat hızı 68.000 satır/saniye ve görüntü çözünürlüğü 1 mm x 0.15 mm'dir. Veri setinde sağlıklı bağlantı elemanlarına ait 525 görüntü, hasarlı bağlantı elemanlarına ait 594 görüntü ve eksik bağlantı elemanlarına ait 479

görüntü bulunmaktadır. Tüm görüntüler Türkiye'deki Ankara-Konya ve Ankara-Eskişehir demiryolu hatları boyunca toplanır ve manuel olarak etiketlenir.

Veri setinde iki tip bağlantı elemanı hatası olduğu bulunmuştur. Bir tip kısmen deforme olmuş bir bağlantı elemanıdır ve diğer tip tamamen kayıp bir bağlantı elemanıdır. Her durum için örnekler Şekil 4.74'te verilmiştir.

(a) Sağlıklı (b) Deforme (c) Kayıp

Şekil 4. 74. Üç farklı bağlantı elemanı durumu

Şekil 4.74'te üç farklı durum için gösterilen bağlantı elemanı görsellerinde kısmen aşınmış veya tamamen eksik olan bağlantı elemanları tren geçişi sırasında daha ciddi sorunlara neden olarak tren çalışma güvenliğini olumsuz etkilemektedir. Görüntü alma sistemi ile çekilen görüntüler raylar, traversler, bağlantı elemanları ve balastlardan oluşmaktadır. Bu çalışmanın amacı bağlantı elemanı hatalarını belirlemek olduğundan, bağlantı elemanı kısmı görüntüden çıkarılmalıdır. Ancak elde edilen görüntüler, trenin altındaki aydınlatma koşullarından ve günün farklı saatlerinde çekilen görüntülerin neden olduğu kontrast farklılıklarından dolayı gürültü içermektedir. Bu nedenle orijinal görüntüye kontrast iyileştirme yöntemi uygulanmıştır. Görüntü iyileştirme aşaması için kullanılan fminsearch fonksiyonunun başlangıç parametreleri kr=1 ve kg=1 olarak seçilmiştir.

Bağlantı elemanı tespiti için önce hem ray hem de travers tespit edilmelidir. Ray ve travers konumlarını elde etmek için Otsu yöntemi kullanılarak yüksek kontrastlı bir görüntü segmentlere ayrılır; ray konumu belirlenir ve rayın soluna yakın bir noktadan bir filtre kullanılarak traverslerin konumları belirlenir; yüksek kontrastlı görüntüden elde edilen traverslere LLBP algoritması uygulanır ve son olarak kırpılmış travers görüntülerinden morfolojik işlemler uygulanarak bağlantı elemanı konumları elde edilir. LLBP sonucunda elde edilen ikili görüntüde bağlantı elemanlarının konumunu belirlemek için morfolojik yakınlaştırma uygulanmıştır. Bu işlem için yapı elemanı kare olarak seçilmiş ve komşuluk değeri 15 olarak alınmıştır.

Ray görüntüsünden bağlantı elemanı bölgesi elde edildikten sonra veri seti oluşturulmuştur. Eğitilmiş ağ modeli, aşağıdaki eğitim parametresi değerleriyle oluşturulmuş veri seti LCNN

modeline verilerek elde edilir: ilk öğrenme oranı = 10-4, maksimum döngü = 30, mini-batch boyutu = 32, doğrulama frekansı = 15 ve optimizasyon = momentumlu stokastik gradyan inişi. Eğitim sırasında hem geliştirilmiş görüntüler hem de herhangi bir ön işleme tabi tutulmadan doğrudan trenin altından alınan görüntüler kullanılır. Ayrıca yansıma ve ölçekleme gibi veri büyütme her iki yönde de uygulanmaktadır. Veri seti, %70 eğitim ve %30 doğrulama olarak bölümlere ayrılmıştır. Doğrulama performansı her koşul için karşılaştırılır ve sonuçlar Tablo 4.29'da verilmiştir.

Tablo 4. 29. Görüntü iyileştirme öncesi ve sonrası karşılaştırmalı sonuçlar

| **Veri seti** | **Doğrulama Oranı (%)** |
|---|---|
| Orjinal görüntüler | 91.37 |
| Orijinal görüntülerde veri arttırma | 92.58 |
| İyileştirilmiş görüntüler | 93.15 |
| Arttırılmış ve iyileştirilmiş görüntüler | 96.82 |

Tablo 4.29'da geliştirilmiş görüntü üzerinde model eğitiminin doğrulama sonuçlarının orijinal görüntüye göre daha iyi olduğu görülmektedir. Bu sonuçlar şaşırtıcı değildir, çünkü orijinal görüntülerde, aydınlatma koşullarının etkileri nedeniyle görüntüyü temsil eden özelliklerin çıkarılması zordur. Ayrıca görüntü büyütme adımından sonra performansta artış olur. Tam bağlantılı, softmax ve sınıflandırma katmanları ile model eğitimi yeterli performansı sağlamamaktadır. Amacımız için, en iyi sınıflandırma yöntemini seçmek için sınıflandırma katmanından önce tam bağlantılı iki katmandan elde edilen öznitelikler birleştirilecektir. Tam bağlantılı katmanların çıktı boyutu da ağ performansını etkiler. Son tam bağlı katman, sınıf sayısını temsil eder ve çalışmamızda üç sınıf olduğu için üç olarak seçilmiştir. Ancak tam bağlı katman-1 (FC1) ve tam bağlı katman-2 (FC2) doğru seçilmelidir. Tablo 4.30'da FC1 ve FC2 için seçilen boyutlara göre elde edilen özniteliklerin sınıflandırma performansı verilmiştir.

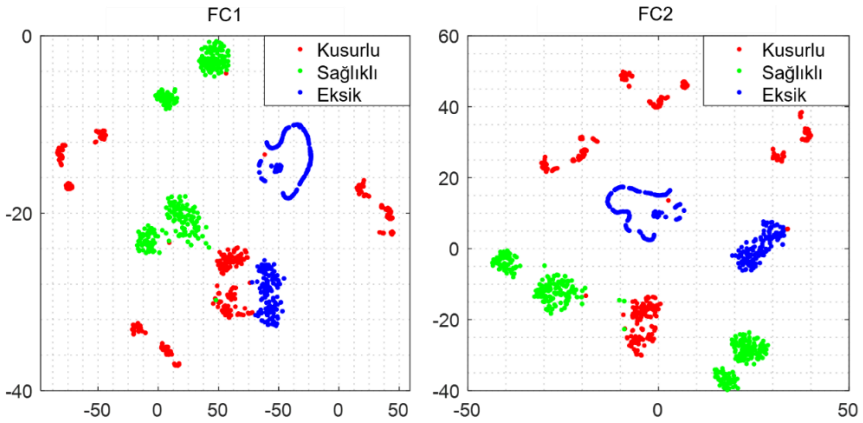
Tablo 4. 30. Farklı tam bağlantılı katmanlar için önerilen CNN'nin performansı

| **Tamamen bağlı katmanların boyutu** | | **Birleşik özelliklerin boyutu** | **Birleştirilmiş özelliklerin doğruluk oranı** |
|---|---|---|---|
| **FC1** | **FC2** | | |
| 100 | 50 | 150 | 93.11 |
| 150 | 100 | 250 | 94.15 |
| 200 | 150 | 350 | 95.00 |

| | | | |
|---|---|---|---|
| 250 | 200 | 450 | 96.61 |
| **300** | **250** | **550** | **99.70** |
| 350 | 300 | 650 | 97.2 |
| 400 | 350 | 750 | 97.9 |
| 450 | 400 | 850 | 97.9 |
| 500 | 450 | 950 | 97.1 |

Tablo 4.30'da, sınıflandırma için Matlab Classification Learner aracına seçilen öznitelikler verilmiştir. Burada, en yüksek performansı veren sınıflandırıcıya göre birleştirilmiş özniteliklerin sınıflandırma performansı seçilir. Tablo 4.30'da en iyi performans FC1 ve FC2 için sırasıyla 300 ve 250 öznitelik seçildiğinde elde edilir. CNN'leri eğitmek için yeterli veri olmadığında, sınıflandırma aşamasından önce önceden eğitilmiş bir CNN'nin katmanından elde edilen öznitelikler bir makine öğrenmesi yöntemine verilir ve sınıflandırma yapılır. Önceden eğitilmiş bir ağdan özellik çıkarımı, daha fazla eğitim gerektirmediğinden, düşük özellikli bir bilgisayarda bile yapılabilir. Ancak bu önceden eğitilmiş ağlarda elde edilen özniteliklerin boyutsallığı oldukça yüksektir. Bu çalışmada daha küçük bir boyut özelliği çıkarılarak bağlantı elemanı hata sınıflandırması yapılmıştır. Tablo 9'te, milyonlarca (M) cinsinden eğitilebilir parametre sayısı ve önerilen CNN mimarisinin özellik çıkarma performansı, son teknoloji ürünü önceden eğitilmiş ağlarla karşılaştırılmıştır.

Tablo 4.31'de verilen önceden eğitilmiş ağlar, özellikle küçük boyutlu ve daha az parametreli olanlardan seçilmiştir. Örneğin, SqueezeNet Alexnet'e yakın performans göstermesine rağmen 50 kat daha hızlı çalışır. GoogleNet'in mimarisi 22 katmana sahiptir ve model boyutu diğer CNN'lere göre düşüktür. Aynı şekilde VGG19 modeli de 16 katmanlı bir ağ modelidir ve Imagenet veri setinde biraz daha yüksek doğruluk oranına sahiptir. Tablo 4.31'de 737 örnek üzerinden öznitelik çıkarımları karşılaştırılmış ve bir görüntü için öznitelik çıkarma süresi verilmiştir. Buradan da görüleceği üzere önerilen ağ modelinin öznitelik çıkarma performansı diğer modellere göre daha iyidir ve gerçek zamanlı uygulamalarda kullanılması mümkün görünmektedir. İstatistiksel bir yöntem olan T-dağıtılmış stokastik komşu gömme (t-SNE), önerilen CNN modelinin tam bağlantılı iki katmanından elde edilen özniteliklerin dağılımını göstermek için kullanılır. Teknik, bir ağ aktivasyon katmanından elde edilen yüksek boyutlu verileri iki boyutlu olarak gösterir. t-SNE yöntemi, doğrusal olmayan eşlemeyi gerçekleştirirken mesafeleri korur. Böylece belirli bir katmandan elde edilen özelliklerin dağılımı öğrenilerek hatalı dağılım gösteren örnekler belirlenebilir. Şekil 4.75'te, önerilen CNN mimarisinin FC1 ve FC2 katmanının bir t-SNE temsili verilmektedir.




Şekil 4. 75. FC1 ve FC2'den çıkarılan özelliklerin t-sne temsili

Tablo 4. 31. Önerilen CNN ve temel önceden eğitilmiş CNN modelleri için özellik çıkarma süreci

| **CNN Modeli** | **Öğrenilebilir Katman Sayısı** | **Öğrenilebilir parametre sayısı** | **Özellik katmanı** | **Özellik sayısı** | **Zaman (ms)** |
|---|---|---|---|---|---|
| Squeeze Net | 18 | 1.24 M | pool10 | 1000 | 8.46 |
| GoogleNet | 22 | 6.7 M | pool5-drop_7x7_s1 | 1024 | 18.00 |
| VGG19 | 16 | 143 M | avg_pool | 2048 | 38.47 |
| LCNN | 8 | 0.9 M | FC1-FC2 | 550 | 6.58 |

Şekil 4.75'te gösterildiği gibi, FC1 katmanının t-SNE temsilinde sağlıklı ve kusurlu verileri ayırt etmek daha zor görünmektedir. FC2 katmanının t-SNE gösteriminde bazı bozulmamış ve deforme olmuş örneklerin daha net olarak ayırt edilebildiği görülmektedir. Önerilen yöntem FC1 ve FC2'den elde edilen öznitelikleri bir araya getirerek makine öğrenmesi yöntemleriyle sınıflandırmaktadır. Bu amaçla farklı sınıflandırma yöntemleri kullanılmıştır. En yüksek sınıflandırma doğruluğunu veren sekiz sınıflandırma yönteminin sonuçları verilmiştir. Önerilen yöntem, hem makine öğrenmesi yöntemleri kullanılarak çıkarılan özniteliklerin değerlendirilmesi hem de CNN'deki tam bağlantılı ve softmax katmanları ile karşılaştırılmıştır. Parametreler tüm önceden eğitilmiş ağlar için aynıdır. Ayrıca sınıflandırma aşamasında 5 kat çapraz doğrulama yapılmıştır. 5 katlı çapraz doğrulamada, veri seti her seferinde bir kısım test için ve kalan 4 kısım eğitim için olmak üzere 5 kısma bölünmüştür. Test verileri için elde edilen

performansların ortalaması alınarak doğruluk değeri elde edilmiştir. Önerilen CNN modelinin farklı önceden eğitilmiş ağlarla karşılaştırma sonuçları Tablo 4.32'de verilmiştir.

Tablo 4. 32. Farklı önceden eğitilmiş CNN modelleriyle karşılaştırma sonuçları

| **Sınıflandırıcı** | **CNN modellerini doğruluk oranları (%)** | | | |
|---|---|---|---|---|
| | **SqueezeNet** | **GoogleNet** | **VGG19** | **Bizim** |
| Fine Tree | 89.2 | 92.3 | 90.5 | 93.9 |
| Linear Discriminant | 86.3 | 89.6 | 88.8 | 92.6 |
| Gaussian Naïve Bayes | 91.4 | 95.4 | 96.7 | 96.8 |
| Cubic SVM | 96.9 | 97.8 | 98.4 | 99.7 |
| Weighted KNN | 93.3 | 94.4 | 95.6 | 96.2 |
| Ensemble Subspace KNN | 96.2 | 97.9 | 97.2 | 96.3 |
| Ensemble Boosted Tree | 50.6 | 53.3 | 60.4 | 51.4 |
| Ensemble RusBoosted Trees | 90.7 | 93.4 | 94.7 | 94.8 |
| Softmax | 92.4 | 93.7 | 94.8 | 96.8 |

Tablo 4.33'te seçilen önceden eğitilmiş ağların ilgili katmanlarından elde edilen sonuçlar, önerilen yöntemin sonuçları ile karşılaştırıldığında, dokuz sınıflandırma yönteminden yedisinde önerilen yöntemlerin daha iyi sonuçlar verdiği görülmektedir. Önerilen CNN mimarisi, daha az özellik kullanır ve performansı, son teknoloji ürünü önceden eğitilmiş ağlardan daha yüksektir. Cubic SVM, makine öğrenme yöntemleri arasında en iyi performansı verdi. Cubic SVM için doğruluk oranı %99,2'dir.

Önerilen yöntemin etkinliğini göstermek için literatürdeki ilgili yöntemleri de karşılaştırdık. Bu yöntemler geleneksel bilgisayarlı görme tekniklerinden ve derin öğrenmeye dayalı tekniklerden oluşmaktadır. Farklı yöntemlerle bağlantı elemanı hata sınıflandırma sonuçları Tablo 4.33'te verilmiştir.

Tablo 4. 33. Önerilen yöntemin literatürdeki temel yöntemler ile karşılaştırılması

| Method | Sınıf | P | R | F1 | Acc | FPS |
|---|---|---|---|---|---|---|
| Probabilistic structure modeling [104] | Sağlıklı | 0.97 | 0.97 | 0.97 | 96.50 | 14 |
| | Deforme | 0.83 | 0.80 | 0.81 | | |
| | Kayıp | 0.82 | 0.88 | 0.85 | | |
| Similarity based deep learning method [105] | Sağlıklı | 0.85 | 0.85 | 0.85 | 92.91 | 196 |
| | Deforme | 0.98 | 0.96 | 0.97 | | |
| | Kayıp | 0.98 | 0.99 | 0.99 | | |
| Hybrid Yolov4 [106] | Sağlıklı | 0.78 | 1.00 | 0.87 | 94.40 | 78 |
| | Deforme | 0.91 | 0.98 | 0.95 | | |
| | Kayıp | 0.82 | 0.89 | 0.85 | | |
| Template matching based nearest neighbor [107] | Sağlıklı | 0.92 | 0.97 | 0.94 | 96.34 | 6 |
| | Deforme | 0.67 | 0.87 | 0.75 | | |
| | Kayıp | 0.83 | 0.87 | 0.85 | | |
| Improved Alexnet [108] | Sağlıklı | 0.97 | 0.92 | 0.95 | 97.52 | 188 |
| | Deforme | 0.96 | 1.00 | 0.98 | | |
| | Kayıp | 1.00 | 0.99 | 0.99 | | |
| Contrast enhancement & Lightweight CNN | Sağlıklı | 1.00 | 1.00 | 1.00 | 99.7 | 221 |
| | Deforme | 0.99 | 0.99 | 0.99 | | |
| | Kayıp | 0.99 | 0.99 | 0.99 | | |

Tablo 4.33'teki sınıflandırma sonuçlarında önerilen modelin aynı probleme uygulanan mevcut modellerden daha iyi performans sağladığı görülmektedir. Ayrıca derin öğrenme yöntemlerinin sınıflandırma aşamasında daha iyi performans sağladığı görülmektedir. Olasılıksal yapı modeli, bağlantı elemanı için bir model oluşturarak hata tespitini gerçekleştirir. Ancak bu model birçok parametrenin ayarlanmasını gerektirir. Desen eşleştirme tabanlı en yakın komşu algoritması, Yönlendirilmiş Gradyanların Histogramından elde edilen özelliklerin büyük boyutu nedeniyle yüksek hesaplama gücü gerektirir. Bu nedenle, bu yöntemi kullanan kare hızı düşüktür. Hybrid Yolov4 kullanan uygulamada, bağlantı elemanı bileşeni geometrik bir nesne

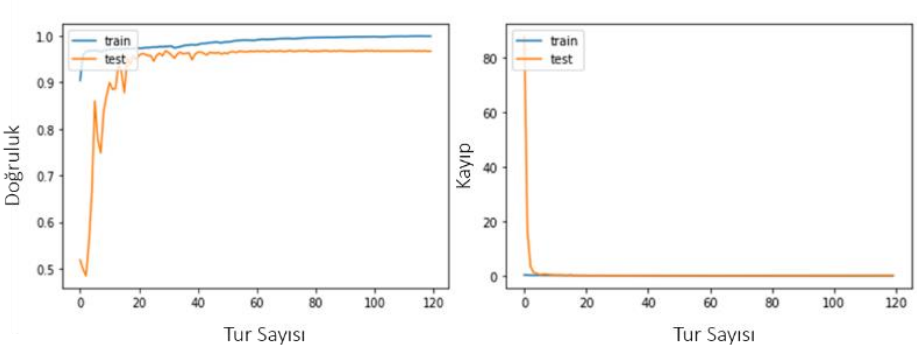
olarak etiketlenmiştir. Bu, hassas etiketleme gerektirdiğinden etiketlemenin yavaş olmasına neden olur. Ek olarak, Yolov4, aydınlatma koşullarına duyarlıdır ve örnek sayısı sınıflar arasında eşit olmayan bir şekilde dağıtıldığında performansı düşüktür. Benzerliğe dayalı derin öğrenme yöntemi, kusur verileri üretirken örnek bağlantı elemanı çiftlerini kullanır. Ancak oluşturulan hata türleri birbirine çok benzediği için numune sayısı artırılsa bile sadece sınırlı kusurlu bağlantı elemanlarının özellikleri öğrenilir. Yöntemimiz için örnek sayıları eşit olarak dağıtılır. Ayrıca aydınlatma ile ilgili problemler ön işleme aşamasında çözüldüğü için önerilen derin öğrenme yönteminin performansı arttı. Önerilen yöntemin literatürdeki yöntemlerle karşılaştırılması sırasında kare/saniye (FPS) değerleri dikkate alınmıştır. Önerilen derin öğrenme ağı, önceden eğitilmiş ağ yapılarından daha az katmana sahip olduğundan, nispeten yüksek bir kare hızına sahiptir (burada bildirilen uygulamada 221 FPS).

Tren raylarındaki kusurların önceden tespiti hem insan güvenliği hem de ekonomik performans için çok önemli bir konudur. Tren raylarındaki kusurlu bağlantı elemanlarını tespit etmek en büyük zorluklardan biridir. Bu çalışmada, bu sorunu çözmek için CNN tabanlı yeni bir derin ağ modeli önerilmiştir. Önerdiğimiz modelin dört ana aşaması vardır: (1) demiryollarından görüntüler toplanır, (2) yüksek kontrastlı ray görüntüleri elde etmek için görüntü iyileştirme uygulanır, (3) ray görüntülerinde bağlantı elemanı bileşenleri tam olarak algılanır ve (4) bağlantı elemanı görüntüleri, hafif bir evrişimli sinir ağı kullanılarak sınıflandırılır. Önerilen LCNN mimarisinin iki ana avantajı vardır. Birincisi, mimari olarak basittir ve daha az hesaplama süresi gerektirir. İkincisi, literatürdeki ana sınıflandırma modellerine kıyasla gelişmiş sınıflandırma performansı gösterir. Önerilen derin modelden elde edilen öznitelikleri sınıflandırmak için en bilinen makine öğrenmesi algoritmaları kullanılır.

Önerilen modelin performansını doğrulamak için bir dizi deney yapıldı. Bu deneylerde, mevcut ana derin mimarilerin performansları, önerilen modelin performansı ile karşılaştırıldı. Deneysel çalışmalar sonucunda önerilen modelin performansının literatürdeki son teknoloji modellere göre daha iyi olduğu gözlemlenmiştir. Önerilen yöntem ayrıca derin öğrenme tabanlı bağlantı elemanı tespit yöntemleri ile karşılaştırılmış ve daha iyi sonuçlar elde edilmiştir. Derin öğrenmeye dayalı tespit yöntemleri, sağlıklı bağlantı elemanlarının tespitine daha fazla odaklanır ve kusurlu bağlantı elemanlarının tespitine daha az uygundur. Bunun temel nedeni, sahada toplanan verilerin çoğunun sağlıklı bağlantı elemanları içermesi ve hatalı veya eksik bağlantı elemanlarının sayısının çok sınırlı olmasıdır. Bu nedenle, gelecekteki çalışmalar, farklı aydınlatma koşulları altında kusurlu ve eksik bağlantı elemanları üretmek için üretken derin öğrenme tekniklerini kullanarak eğitim verilerinin dengesini iyileştirmeyi amaçlayacaktır.

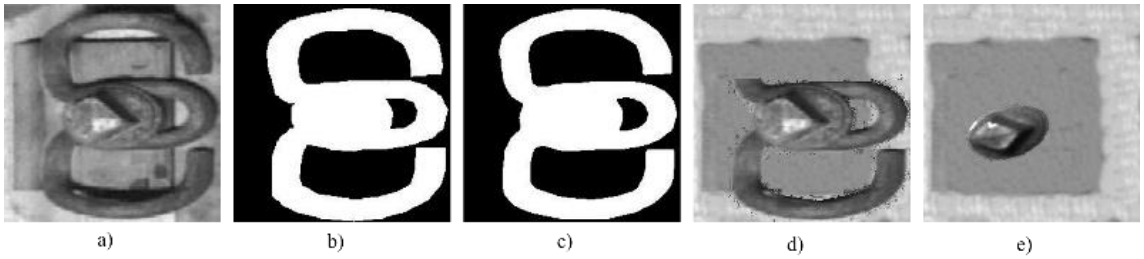
### 4.9. Derin Öğrenme Yöntemleri ile Demiryolu Bağlantı Elemanlarının Sınıflandırılması

Demiryolu hattından toplanan sağlıklı bağlantı elemanlarını içeren veri seti veri üretim teknikleri ile dengelenmiştir. Daha sonra derin öğrenme yöntemleri: CNN, VGG-16 ve ResNet50 demiryolu bağlantı elemanlarını içeren dengelenmiş veri setine uygulanmıştır. Çalışmadaki deney ortamı Windows 10 işletim sistemi, Intel(R) Core (TM) i5-1135G7 @ 2.40GHz işlemci, 16 GB bellek ve NVIDIA GeForce MX450 2GB ekran kartı şeklinde ayarlanmıştır. DATEM'den alınan veri seti 534 tane sağlıklı demiryolu bağlantı elamanı içermektedir. Sağlıklı görüntülere ek olarak 534 tanede arızalı ve eksik demiryolu bağlantı elemanı bölütleme sonucu üretilmiştir. Unet bölütleme modeli yeni bağlantı elemanı elde etmek için eğitilmiştir. Burada eğitim ve doğrulama seti için sırasıyla 0.8 ve 0.2 oranında bağlantı elemanı seçilmiştir. Optimizasyon fonksiyonu olarak Adam seçilmiştir. Eğitim için tur sayısı 120 olarak ayarlanmıştır. Unet mimarisinin doğruluk ve kayıp grafikleri Şekil 4.76'da görülmektedir.




Şekil 4. 76. Unet öğrenme grafikleri

Şekil 4.76'da görüldüğü gibi 10. döngüden sonra göze çarpan bir değişiklik olmasa da tur sayısının 120 seçilmesinin nedeni eğitim modelinin aşırı öğrenmeye gitmediğini göstermektir. Kayıp grafiği incelendiğinde kayıp değerinin 0'a yakın olduğu görülmektedir. Bu değer 0'a ne kadar yakınsa eğitimin o kadar başarılı gerçekleştiği anlamına gelmektedir. Şekil 4.77'de Unet ile elde edilmiş görüntü ve bu görüntüden elde edilen arızalı ve eksik demiryolu bağlantı elemanları görülmektedir. Unet modeli %99'luk doğruluk oranı yakalamıştır. Kusurlu verinin bulunamadığı durumlarda sağlıklı demiryolu bileşeni görüntülerini kullanarak kusurlu verilerin elde edilebildiği ispatlanmıştır. Böylelikle veri seti dengelenmiştir.




Şekil 4. 77. Unet sonucu elde edilen görüntüler a) Sağlıklı bağlantı elemanı b) Sağlıklı referans veri c) Segmentasyon sonucu d) Ön plana çıkarılmış arızalı bağlantı elemanı e) Ön plana çıkarılmış eksik bağlantı elamanı

Sınıflandırıcının performansı aşağıdaki dört standart metrik kullanılarak değerlendirilmektedir. Bu metrikler; doğru pozitif (True Positive-TP), yanlış pozitif (False Positive-FP), doğru negatif (True Negative-TN) ve yanlış negatif (False Negative-FN) tahmin değerleri kullanılarak değerlendirilmektedir.

$$\text{Doğruluk(A)} = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.5}$$

$$\text{Kesinlik(P)} = \frac{TP}{TP+FP} \tag{4.6}$$

$$\text{Duyarlılık(R)} = \frac{TP}{TP+FN} \tag{4.7}$$

$$F1 = 2 * \frac{PxR}{P+R} \tag{4.8}$$

$$\text{Özgüllük(P)} = \frac{TN}{TN+FP} \tag{4.9}$$

Yukarıdaki metriklerin tümü, karışıklık matrisinden türetilmiştir. Sınıflandırıcının performansını belirlemek için tek bir metrik yeterli değildir. Doğruluk, tüm sınıflardaki girdi verilerinin performansını ölçmek için ana metriktir. Modelin performansını sadece doğruluk metriği ile ölçmek çoğu zaman yanılgıya yol açabilmektedir. Bundan dolayı oluşturulan modelin performansının değerlendirilmesinde sadece doğruluk metriği değil buna ek olarak kullanılacak metriklere ihtiyaç duyulmaktadır. Diğer metrikler her sınıfa özeldir ve ilgili sınıfın sınıflandırma algoritmasında geri çağırma oranını hesaplar. Tablo 4.34'te VGG-16 mimarisinin performans değerlendirme ölçütleri görülmektedir.

Tablo 4. 34. VGG-16 mimarisinin performans metrikleri

| Sınıf/Metrik | P | R | F1 |
|---|---|---|---|
| Arızalı | 1.00 | 1.00 | 1.00 |
| Sağlıklı | 1.00 | 1.00 | 1.00 |
| Eksik | 1.00 | 1.00 | 1.00 |
| A | 100% | | |
| R | 1.00 | | |
| S | 1.00 | | |

Tablo 4.35'te ResNet50 mimarisinin performans değerlendirme ölçütleri görülmektedir.

Tablo 4. 35. ResNet50 mimarisinin performans metrikleri

| Sınıf/Metrik | P | R | F1 |
|---|---|---|---|
| **Arızalı** | 0.94 | 1.00 | 0.97 |
| **Sağlıklı** | 1.00 | 0.94 | 0.97 |
| **Eksik** | 1.00 | 1.00 | 1.00 |
| **A** | 98% | | |
| **R** | 1.00 | | |
| **S** | 0.94 | | |

Tablo 4.36'da oluşturulan CNN mimarisinin performans değerlendirme ölçütleri görülmektedir.

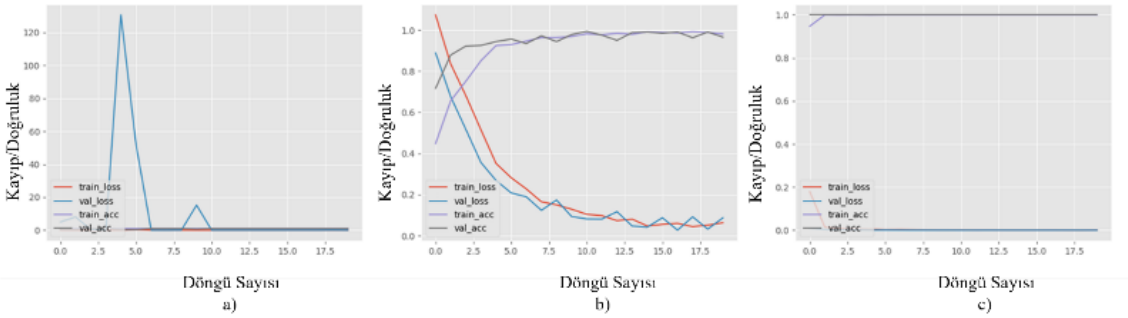
Tablo 4. 36. CNN mimarisinin performans metrikleri

| Sınıf/Metrik | P | R | F1 |
|---|---|---|---|
| Arızalı | 1.00 | 1.00 | 1.00 |
| Sağlıklı | 1.00 | 1.00 | 1.00 |
| Eksik | 1.00 | 1.00 | 1.00 |
| A | 100% | | |
| R | 1.00 | | |
| S | 1.00 | | |

Tablo 4.34 ve 4.36'da görüldüğü gibi, oluşturulan CNN ve VGG-16 mimarileri RestNet50 mimarisine göre daha iyi performans vermiştir. Önerilen bütün modeller yüksek doğruluk oranı yakalamışlardır. Oluşturulan CNN ve VGG-16 modelleri arızalı, eksik ve sağlıklı demiryolu bağlantı elemanlarını %100 doğruluk oranıyla sınıflandırmıştır. Önerilen modellerin öğrenme grafikleri Şekil 4.78'de görülmektedir. Performans metrikleri seyreltme değeri 0.3 alındığında elde edilmiştir. Seyreltme değerini arttırdığımızda herhangi bir performans kaybı yaşanmamıştır. Sonuçlar Tablo 4.37'de görülmektedir. Doğruluk oranlarındaki tutarlılık önerilen modellerin ezberleme yapmadığını göstermektedir.

Tablo 4. 37. Seçilen seyretme değerine göre başarım oranları

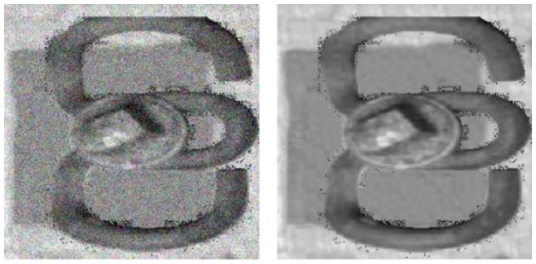
| Model/Metrik | Seyreltme Değeri (0.3) | Seyreltme Değeri (0.5) |
|---|---|---|
| CNN (A) | 100% | 100% |
| CNN (R) | 1.00 | 1.00 |
| CNN (S) | 1.00 | 1.00 |
| RestNet50 (A) | 98% | 98% |
| RestNet50 (R) | 1.00 | 1.00 |
| RestNet50 (S) | 0.94 | 0.94 |
| VGG-16 (A) | 100% | 100% |
| VGG-16 (R) | 1.00 | 1.00 |
| VGG-16 (S) | 1.00 | 1.00 |




Şekil 4. 78. Önerilen modellerin öğrenme grafikleri a) CNN b) ResNet50 c) VGG-16

DATEM'den alınan veri seti iyileştirilme ve veri üretim tekniklerinden geçirilmiş daha sonra ise derin öğrenme modelleriyle eğitilmiş ve test edilmiştir. Bölütleme sonucu üretilen verilere sentetik gürültü eklendiğinde önerilen modellerin performanslarında kayba rastlanılmamıştır.

Şekil 4.79’da sentetik gürültü eklenmiş ve bölütleme sonucu oluşturulmuş bağlantı elemanları görülmektedir.



a) Gürültülü Veri b) Üretilmiş Veri

Şekil 4. 79. a) Sentetik gürültü eklenmiş bağlantı elemanı b) İyileştirilmiş ve üretilmiş bağlantı elemanı

Tablo 4.38’de görüldüğü gibi gürültü ekleme işlemi başarım oranlarına negatif olarak yansımamıştır. Üretilen (arızalı ve eksik bağlantı elemanları) kusurlu görüntülerin farklı çeşitlilikte olması önerilen modellerin başarılı olmasında temel etken olmuştur. Böylelikle gürültülü veri setlerinin performans sorunlarının çözülebileceği kanıtlanmıştır. Önerilen CNN modelinin performansı, literatürdeki temel yaklaşımlarla da karşılaştırılmıştır. Karşılaştırma sonuçları Tablo 4.39’da verilmiştir.

Demiryollarındaki arızaların önceden tespiti her zaman hem ekonomi hem de insan sağlığı için çok önemli bir konu olmuştur. Tren raylarındaki arızalı ve eksik bağlantı elemanlarının tespiti en büyük problemlerden bir tanesidir. Bu çalışmada, bu sorunu çözmek için CNN tabanlı yeni bir derin ağ modeli önerilmiştir. Önerdiğimiz modelin iki ana aşaması vardır. İlk olarak, demiryollarından toplanan dengesiz veri seti Unet yardımıyla dengelenmiştir. Sonra ise farklı derin öğrenme modelleriyle arızalı raylar sınıflandırılmıştır. Önerilen CNN mimarisinin iki ana avantajı vardır. Birincisi mimari olarak basittir ve daha az hesaplama süresine ihtiyaç duyar. İkincisi, literatürdeki önemli sınıflandırma modellerine kıyasla daha iyi bir sınıflandırma performansı göstermesidir. Ayrıca bu çalışmada ray üzerindeki bağlantı elemanları ile ilgili yeni bir veri seti oluşturulmuştur. Önerilen modelin performansını doğrulamak için bir dizi deney yapılmıştır. Bu çalışmada ayrıca, önerilen modellerin performansları gürültülü veri seti üzerinde test edilmiş ve önerilen modellerin performanslarında düşüş gözlemlenmemiştir.

Tablo 4. 38. Önerilen modellerin gürültülü ve üretilmiş veri seti üzerindeki performansları

| Veri seti | Mimari | Etiket / P | | Etiket / R | | Etiket / F1 | | A | Döngü Sayısı |
|---|---|---|---|---|---|---|---|---|---|
| Gürültülü Veri Seti | VGG-16 | Sağlıklı | 1.00 | Sağlıklı | 1.00 | Sağlıklı | 1.00 | 1.00 | 20 |
| | | Arızalı | 1.00 | Arızalı | 1.00 | Arızalı | 1.00 | | |
| | | Eksik | 1.00 | Eksik | 1.00 | Eksik | 1.00 | | |
| Gürültülü Veri Seti | ResNet50 | Sağlıklı | 1.00 | Sağlıklı | 0.94 | Sağlıklı | 0.97 | 0.98 | 20 |
| | | Arızalı | 0.95 | Arızalı | 1.00 | Arızalı | 0.97 | | |
| | | Eksik | 1.00 | Eksik | 1.00 | Eksik | 1.00 | | |
| Gürültülü Veri Seti | Oluşturulmuş CNN | Sağlıklı | 1.00 | Sağlıklı | 1.00 | Sağlıklı | 1.00 | 1.00 | 20 |
| | | Arızalı | 1.00 | Arızalı | 1.00 | Arızalı | 1.00 | | |
| | | Eksik | 1.00 | Eksik | 1.00 | Eksik | 1.00 | | |
| Üretilmiş Veri Seti | VGG-16 | Sağlıklı | 1.00 | Sağlıklı | 1.00 | Sağlıklı | 1.00 | 1.00 | 20 |
| | | Arızalı | 1.00 | Arızalı | 1.00 | Arızalı | 1.00 | | |
| | | Eksik | 1.00 | Eksik | 1.00 | Eksik | 1.00 | | |
| Üretilmiş Veri Seti | ResNet50 | Sağlıklı | 1.00 | Sağlıklı | 0.93 | Sağlıklı | 0.97 | 0.98 | 20 |
| | | Arızalı | 0.94 | Arızalı | 1.00 | Arızalı | 0.97 | | |
| | | Eksik | 1.00 | Eksik | 1.00 | Eksik | 1.00 | | |
| Üretilmiş Veri Seti | Oluşturulmuş CNN | Sağlıklı | 1.00 | Sağlıklı | 1.00 | Sağlıklı | 1.00 | 1.00 | 20 |
| | | Arızalı | 1.00 | Arızalı | 1.00 | Arızalı | 1.00 | | |
| | | Eksik | 1.00 | Eksik | 1.00 | Eksik | 1.00 | | |

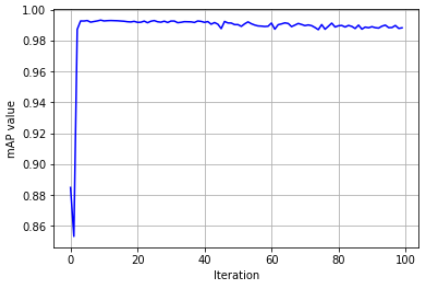
Tablo 4. 39. Önerilen CNN Modelinin Literatürdeki Modellerle Karşılaştırılması

| Model | Etiket / P | | Etiket / R | | Etiket / F1 | | A |
|---|---|---|---|---|---|---|---|
| Probabilistic structure | Sağlıklı | 0.97 | Sağlıklı | 0.97 | Sağlıklı | 0.97 | 0.96 |
| | Arızalı | 0.97 | Arızalı | 0.80 | Arızalı | 0.81 | |

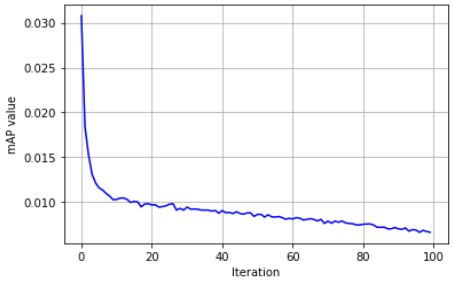
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| modeling [104] | Eksik | 0.97 | Eksik | 0.88 | Eksik | 0.85 | |
| Similarity based deep learning method [105] | Sağlıklı | 0.85 | Sağlıklı | 0.85 | Sağlıklı | 0.85 | 0.92 |
| | Arızalı | 0.98 | Arızalı | 0.96 | Arızalı | 0.97 | |
| | Eksik | 0.98 | Eksik | 0.99 | Eksik | 0.99 | |
| Template matching based nearest neighbor [107] | Sağlıklı | 0.92 | Sağlıklı | 0.97 | Sağlıklı | 0.94 | 0.96 |
| | Arızalı | 0.67 | Arızalı | 0.87 | Arızalı | 0.75 | |
| | Eksik | 0.83 | Eksik | 0.87 | Eksik | 0.85 | |
| Improved Alexnet [108] | Sağlıklı | 0.97 | Sağlıklı | 0.92 | Sağlıklı | 0.95 | 0.95 |
| | Arızalı | 0.96 | Arızalı | 1.00 | Arızalı | 0.98 | |
| | Eksik | 1.00 | Eksik | 0.99 | Eksik | 0.99 | |
| Bizim CNN | Sağlıklı | 1.00 | Sağlıklı | 1.00 | Sağlıklı | 1.00 | 1.00 |
| | Arızalı | 1.00 | Arızalı | 1.00 | Arızalı | 1.00 | |
| | Eksik | 1.00 | Eksik | 1.00 | Eksik | 1.00 | |

## 4.10. Gerçek Zamanlı Bağlantı Elemanı Kusurlarının Tespiti için Derin Öğrenme Tabanlı Hibrit Yaklaşım İçin Deneysel Sonuçlar

Önerilen derin öğrenme tabanlı yaklaşımla, bağlantı elemanlarını tespit etmek ve kusurları belirlemek için 8461 bağlantı elemanı görüntüsü içeren bir veri seti kullanıldı [109]. Veri seti, bir demiryolu aracına yerleştirilen kamera ile alınmıştır. Algılama ve tanıma ağları, 8 GB belleğe sahip GTX2070 Max Q GPU kartına sahip bir mobil iş istasyonu bilgisayarında eğitilmiştir. YOLOv4-Tiny modelinin ImageNet üzerinde transfer öğrenimi ile önceden eğitilmiş bir modeli kullanılmıştır. Model eğitiminde 64 batch büyüklüğü alınmış ve 100 epokluk eğitim gerçekleştirilmiştir. Öğrenme oranı 1e-4 olarak alınmış ve optimize edici olarak SGD seçilmiştir. Şekil 4.80, YOLOv4-Tiny eğitimindeki mAP ve kayıp grafiğini göstermektedir.



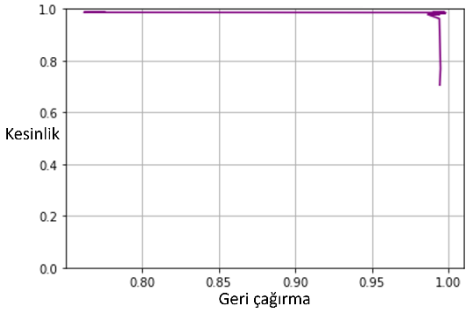

(a)




(b)

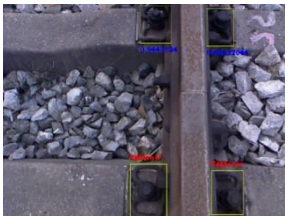
Şekil 4. 80. YOLOv4-Tiny'un mAP ve kayıp grafiği. (a) mAP değeri, (b) Eğitim kaybı

Şekil 4.80'de elde edilen kayıp ve mAP grafiğine göre bağlantı elemanlarının tespit oranı oldukça yüksektir. Ayrıca, kesinlik-hatırlama eğrisi, performans ölçümü için kullanılabilecek başka bir ölçüdür. Bu eğri 1'e ne kadar yakınsa, modelin performansı o kadar yüksek olur. YOLOv4-Tiny modeli için kesinlik-geri çağırma eğrisi Şekil 4.81'de verilmiştir.

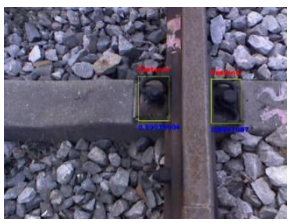



Şekil 4. 81. YOLOv4-Tiny için kesinlik-geri çağırma eğrisi

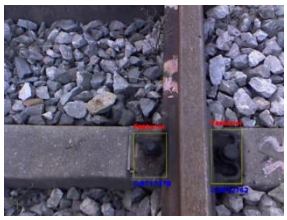
Bu eğri 1'e yakın olduğu için hem kesinlik hem de hatırlama açısından iyi performans gösterdiği söylenebilir. Şekil 4.82'de farklı numuneler için elde edilen tespit sonuçları verilmiştir.



(a)



(b)



(c)

Şekil 4. 82. Farklı koşullar için bağlantı elemanı algılama sonuçları. (a) 4 adet sağlıklı bağlantı elemanı, (b) 1 adet sağlıklı 1 adet kusurlu bağlantı elemanı, (c) 1 adet eksik 1 adet sağlıklı bağlantı elemanı

Şekil 4.82'de YOLOv4-Tiny modelinin sağlamlığını test etmek için arızalı ve eksik bağlantı elemanları da verilmiştir. Bağlantı elemanının eksik veya kısmen kırıldığı durumlarda algoritma algılama işlemini gerçekleştirir. Bağlantı elemanları tespit edildikten sonra, kusur tespiti için CNN modeli çalıştırılır. Bağlantı elemanı için sağlıklı, kusurlu ve eksik bağlantı elemanları dikkate alınır. Her bir durum için veri setindeki örnek sayıları Tablo 4.40'ta verilmiştir.

Tablo 4. 40. Her bir sınıf için örnek sayıları

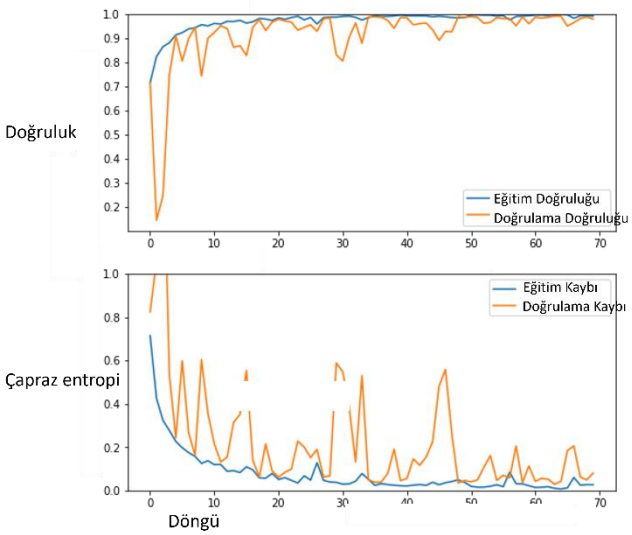
| **Sınıf** | **Örnek Sayısı** |
|---|---|
| Sağlıklı | 5000 |
| Kusurlu | 1000 |
| Eksik | 1000 |

Tablo 4.40'ta kusurlu numune sayısı sağlıklı numune sayısından daha azdır. Özellikle performans ölçütlerindeki sınıf dengesizliğini hesaba katmak için veri seti bu şekilde seçilmiştir. Önerilen derin sinir ağının eğitim parametreleri Tablo 4.41'de verilmiştir.

Tablo 4. 41. Hafif evrişimli CNN ağının parametreleri

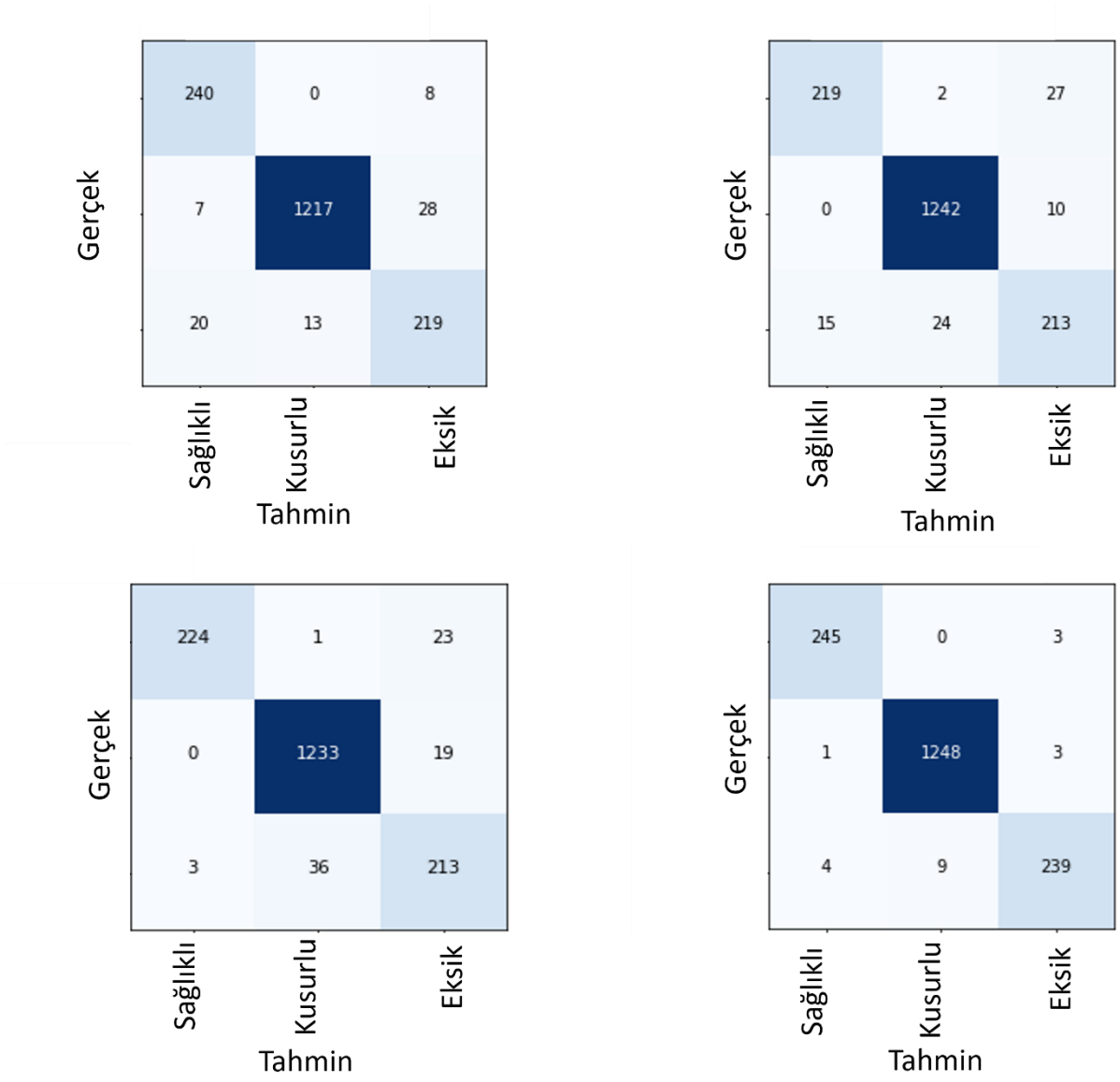
| **Parametre** | **Değer** |
|---|---|
| Öğrenim oranı | 1e-3 |
| Optimizasyon fonksiyonu | Adam |
| Batch boyutu | 32 |
| Epok | 70 |

Derin öğrenme ağı Tablo 4.41'deki parametrelere göre çalıştırılmıştır. Buna göre, eğitim ve doğrulama için sınıflandırma performansı ve kayıp fonksiyonu Şekil 4.83'te verilmiştir.




Şekil 4. 83. Eğitim ve doğrulama veri seti için doğruluk ve kayıp grafiği

Şekil 4.83'te verilen doğruluk ve kayıp değerlerinde, belirli bir iterasyondan sonra hem eğitim hem de doğrulama veri setlerinde doğruluğun sabitlendiği görülmektedir. Önerilen CNN tabanlı hata sınıflandırma ağının performansı da literatürde bilinen transfer öğrenme yöntemleri ile karşılaştırılmıştır. Bu amaçla önerilen yöntem MobilenetV2 [110], InceptionV3 [111] ve VGG16 [112] gibi yöntemlerle karşılaştırılmıştır. Her bir yöntem için karmaşıklık matrisleri Şekil 4.84'te verilmiştir.




Şekil 4. 84. Farklı CNN modellerinin karmaşıklık matrisleri

Şekil 4.84'te elde edilen karışıklık matrisleri üzerinde sınıflandırma için performans metrikleri karşılaştırılmıştır. Bu amaçla, karışıklık matrislerinin doğruluk, kesinlik, hatırlama ve F1 kriterleri karşılaştırılmıştır. Ayrıca her bir yöntem için parametre sayısı ve işlem süreleri değerlendirilmiştir. Bu karşılaştırma sonuçları Tablo 4.42'de verilmiştir.

Tablo 4. 42. Farklı transfer öğrenme modelleriyle performans karşılaştırması

| Model | Kesinlik | Geri Çağırma | F1 | Doğruluk | İşlem Zamanı (ms) | Eğitilebilen Parametre (Milyon) | Model Boyutu (MB) |
|---|---|---|---|---|---|---|---|
| InceptionV3 | 91.66 | 93.66 | 92.33 | 96.00 | 6.0 | 23.8 | 97 |
| VGG16 | 93.33 | 91.00 | 92.00 | 95.00 | 6.4 | 138.4 | 204 |
| MobileNetV2 | 92.33 | 90.66 | 91.66 | 96.00 | 2.17 | 3.4 | 16.6 |
| Önerilen CNN | 98.33 | 98.00 | 97.66 | 99.00 | 1.53 | 0.3 | 1.71 |

Bağlantı elemanı hata tespiti için başarı oranı yüksek modeller seçilmiştir. Bu modellerden bazıları çok sayıda parametreye sahiptir. Ancak önerilen model, darboğaz bloklarının yanı sıra kullandığı derinlik ve noktasal evrişim katmanları sayesinde küçük bir ağ yapısına sahiptir. Ayrıca son katmanda kullanılan tam bağlantılı katman sayesinde ağın genelleme özelliği artırılmıştır. Önerilen yöntemin etkinliğini göstermek için, bağlantı elemanı hata tanıma için literatürde önerilen yaklaşımlarla karşılaştırmalar yapılmıştır. Karşılaştırma sonuçları Tablo 4.43'te verilmiştir.

Tablo 4. 43. Bağlantı elemanı tanıma yöntemlerinin karşılaştırma sonuçları

| **Referans** | **Metot** | **Sınıf Sayısı** | **Doğruluk Oranı (%)** | **FPS** |
|---|---|---|---|---|
| [113] | Derin öğrenme tabanlı segmentasyon ve şablon eşleştirme | 3 | 93.30 | 21.10 |
| [114] | Daha hızlı RCNN tabanlı bağlantı elemanı denetimi | 3 | 97.90 | 260.00 |
| [115] | Kontrast geliştirme ve geliştirilmiş CNN | 3 | 96.80 | 221.00 |
| [116] | İki CNN kullanarak görsel benzerlik | 4 | 92,69 | 196.00 |
| [109] | RGBD-görüntü tabanlı SVM | 3 | 95.83 | 71.37 |
| [117] | UNET tabanlı segmentasyon ve geliştirilmiş AlexNet | 3 | 97.52 | 188.00 |
| Önerilen | YOLOv4_Tiny+Geliştirilmiş CNN Modeli | 3 | 98.57 | 279.06 |

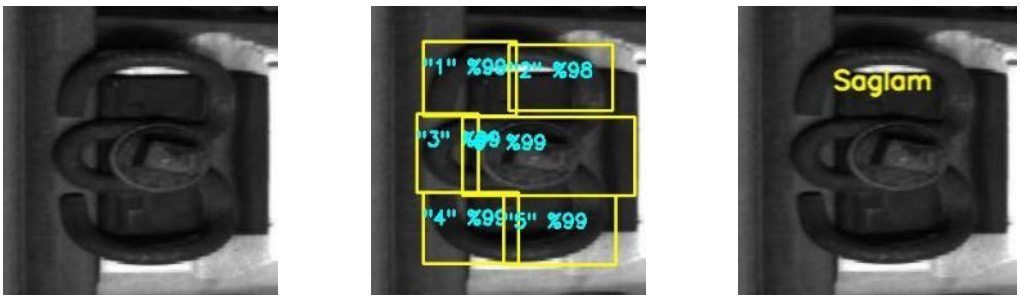
Tablo 4.43'te verilen karşılaştırma sonuçlarında, [113]'te önerilen yaklaşım, önce YOLACT tabanlı bölütleme yöntemiyle bağlantı elemanlarını böler ve daha sonra desen eşleştirme ile hatalı bağlantı elemanları bulur. Yöntem, ayrıntılı etiketleme gerektirir ve rayın sağındaki ve solundaki elemanlar kusurluysa yanlış sonuçlar verir. [114]'de sunulan çalışmada, ray bağlantı elemanı tespiti için Faster RCNN yöntemi kullanılmış ve hem hata sınıflandırması hem de tespiti gerçekleştirilmiştir. [115]'deki çalışmada, bağlantı elemanının konum belirlemesini daha doğru hale getirmek için görüntü iyileştirme uygulanmıştır. Ancak kullanılan veri setinin boyutu küçük ve dengelidir. [116]'da önerilen yaklaşım, iki aşamalı bir evrişimli sinir ağı önermektedir. Son aşamada ise ayrılmış ağ yapısında hem sınıflandırma hem de benzerlik karşılaştırmaları yapılır. [117]'de, hatalı veri sayısını artırmak için bölütleme işlemi yapıldıktan sonra, bağlantı elemanı elde edilerek kusur oluşturulmakta ve belirlenen bir arka plana yerleştirilmektedir. Buradaki temel sorun, bağlantı elemanının arka planının sabit olması ve kullanılan görüntülerin benzer olması ve ağın hafızaya almasıdır. [109]'daki çalışmada, sınıflandırma için RGBD görüntüleri elde edilmiştir ve bu görüntülerin alınması ekstra bir yük getirmektedir. Tablo 4.43'teki karşılaştırma sonuçları incelendiğinde, önerilen yöntemin hem doğruluk hem de hız açısından daha iyi olduğu görülmektedir.

## 4.11. Demiryolu Bağlantı Elemanlarında Bulunan Kusurların YOLOv4 ve Bulanık Mantık Kullanarak Tespiti İçin Deneysel Sonuçlar

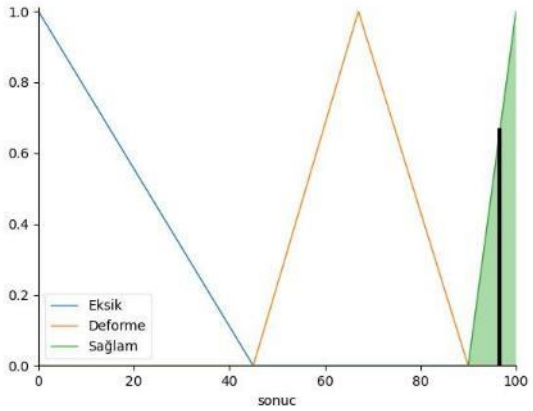
Demiryolundan elde edilen görüntülerden, bağlantı elemanlarında farklı çeşitlerde kusurlar olduğu gözlenmiştir. Elde edilen bağlantı elemanı görüntüleri sağlam, deforme ve eksik olmak üzere üç sınıfa ayrılmıştır. Önerilen algoritma, üç ayrı sınıf için de test edilmiştir. Şekil 4.85'te sağlam, Şekil 4.86'da deforme ve Şekil 4.87'de eksik bağlantı elemanlarının bulunduğu senaryolar denenip sonuçlar gösterilmiştir. Bulanık mantık çıkışı %0 ile %45 arasında ise "eksik", %45 ile %90 arasında ise "deforme", %90 üzeri ise "sağlam" olarak sınıflandırılmıştır.

Şekil 4.85.a'da verilen bağlantı elemanı sağlam olduğu için Şekil 4.85.b'de görüldüğü gibi bağlantı elemanının parçalarının güven değerler yüksek değerli çıkmıştır. Bulanık mantık sonucu da yüksek çıktığından görüntü sağlam olarak nitelendirilmiştir (Şekil 4.85.c). Şekil 4.85.d'de verilen sonuç grafiğinde görüldüğü gibi göre bağlantı elemanı %90 üzeri sağlamlık oranı elde edilmiştir. Şekil 3.26.a'da verilen bağlantı elemanının alt bölümü eğildiği için Şekil 4.85.b'de görüldüğü gibi üstte kalan dört parça yüksek oranlı güven değerleri oluştururken alttaki iki parçanın düşük güven değerleri düşüktür. Bu da sonucun daha düşük değerlerde çıkmasını sağlamaktadır. Bu nedenle, Şekil 4.85.c'de görüldüğü gibi görüntü, deforme olarak sınıflandırılmıştır. Şekil 4.85.d'de verilen sonuç grafiğine göre bağlantı elemanın sağlamlık yüzdesi %45 ile %90 aralığındadır.

Şekil 4.86.a'da verilen görüntüde ise bağlantı elemanı kopmuştur. Bu nedenle, sadece ortadaki vida yüksek güven değeri oluştururken diğer parçalar düşük değerli ya da sıfırdır. Yerinde olmayan parçaların güven değerlerinin %0 olması sonuç yüzdesini düşürmektedir. Şekil 4.86.d'de görüldüğü gibi sonuç değeri %45'ten düşük olduğu için eksik olarak sınıflandırılmıştır. Sınıflandırma sonucu Şekil 4.86.c'de gösterilmiştir.



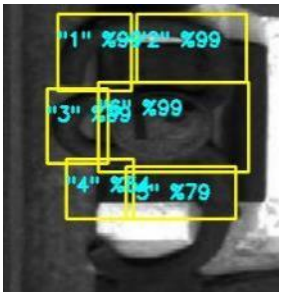

a) b) c)



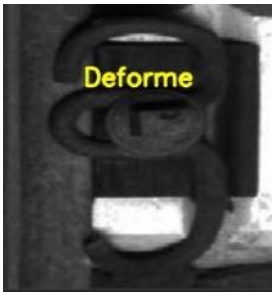

d)

Şekil 4. 85. Sağlam bağlantı elemanı için elde edilen sonuçlar a) Giriş görüntüsü, b) YOLOv4 çıktısı, c) Sonuç d) Sonuç grafiği

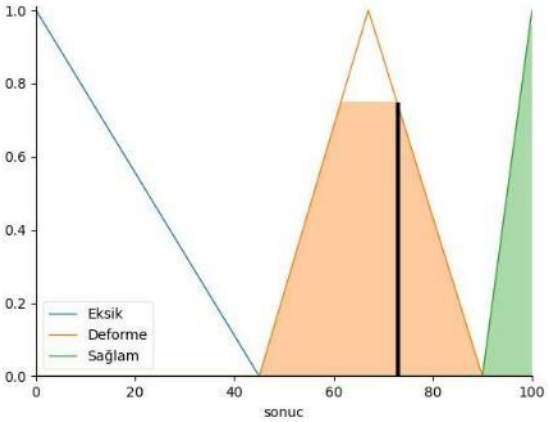
a)




b)

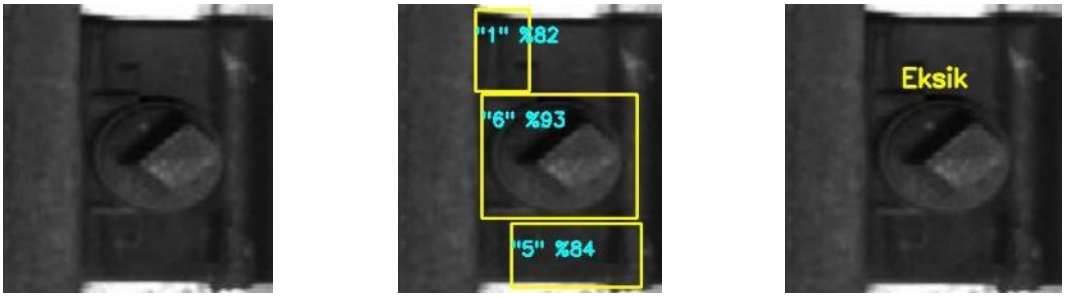



c)

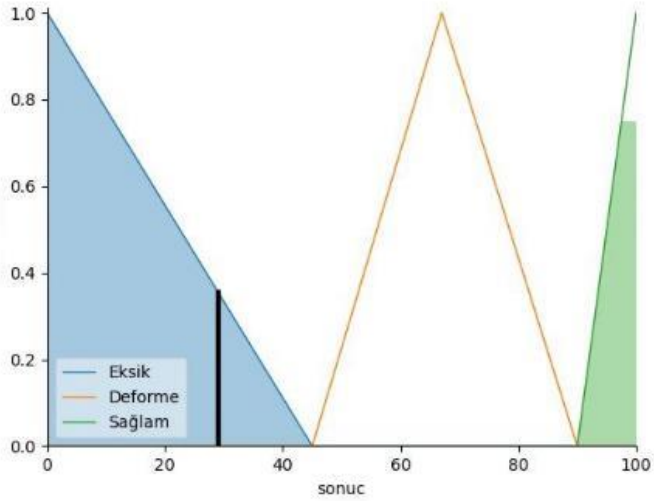



d)

Şekil 4. 86. Deforme bağlantı elemanı için elde edilen sonuçlar a) Giriş görüntüsü, b) YOLOv4 çıktısı, c) Sonuç d) Sonuç grafiği




a) b) c)




d)

Şekil 4. 87. Eksik bağlantı elemanı için elde edilen sonuçlar a) Giriş görüntüsü, b) YOLOv4 çıktısı, c) Sonuç d) Sonuç grafiği

Yapılan çalışmada, tahmin başarı ölçütü için karmaşıklık matrisi kullanılmıştır. Karmaşıklık matrisi Tablo 4.44'te verilen parametrelerden oluşmaktadır.

Tablo 4. 44. Karmaşıklık matrisi ölçüm değerleri için parametreler

| Parametre | Açıklama |
|---|---|
| Doğru Pozitif (DP) | Gerçek sonuç sağlam olup sağlam olarak tahmin edilenler |
| Doğru Negatif (DN) | Gerçek sonuç deforme olup deforme olarak tahmin edilenler |
| Yanlış Pozitif (YP) | Gerçek sonuç sağlam olup deforme olarak tahmin edilenler |
| Yanlış Negatif(YN) | Gerçek sonuç deforme olup sağlam olarak tahmin edilenler |

Sınıflandırma sonuçlarını değerlendirmek için üç metrik ve kullanılmıştır. Bu metrikler doğruluk(accuracy), hassasiyet(precision), geri çağırma(recall) ve F1 değerleridir. Uygulamanın, karmaşıklık matrisi sonuçları Tablo 4.45'te verilmiştir.

Tablo 4. 45. Bağlantı elemanı kusur tespiti için karmaşıklık matrisi

| | | **Tahmin edilen sınıf** | |
|---|---|---|---|
| | | Sağlam | Kusurlu |
| **Doğru Sınıf** | Sağlam | 199 | 1 |
| | Kusurlu | 2 | 198 |

Tablo 4. 46. Dört ayrı ölçüt için performans değerleri

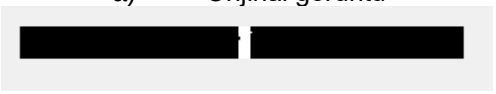
| **Doğruluk(%)** | **Hassasiyet(%)** | **Geri Çağırma(%)** | **F1 (%)** |
|---|---|---|---|
| 99.25 | 99 | 99.5 | 99.24 |

Önerilen algoritma, 400 bağlantı elemanı görüntüsü üzerinde test edilmiştir. 400 görüntü içerisinde 200 adet sağlam ve 200 adet kusurlu görüntü bulunmaktadır. Sonuçlar Tablo 4.46'da gösterilmiştir. Önerilen algoritma 400 adet görüntüden 397'sini doğru, 3'ünü ise yanlış değerlendirmiştir. Tablo 4.46'da farklı ölçütler için doğruluk değerleri verilmiştir. Elde edilen sonuçlar, önerilen YOLOv4 ve bulanık mantık tabanlı algoritmanın bağlantı elemanı kusur tespiti için yüksek performans ile çalıştığını göstermektedir.

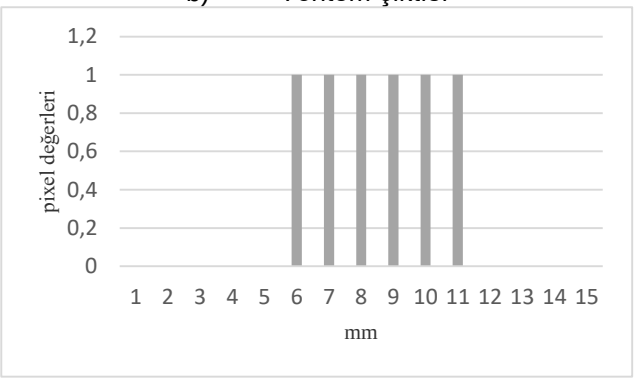
## 4.12. Demiryolu Genleşme Aralıklarının Görüntü İşleme Teknikleri ile Tespiti ve Ölçümü İçin Deneysel Sonuçlar

Önerilen yöntemi test etmek için demiryolu hattı boyunca sabit yükseklikte hareket eden IHA tarafından elde edilen görüntüler kullanılmıştır. Bu görüntüler MATLAB programında ön işleme tabi tutulmuş ve raylar kenar algılama algoritması ile tespit edilmiştir. Tespit edilen raylar kesilerek ikili formata çevrildi ve negatifi alındı. Görüntülerdeki piksel değerleri ölçülerek kalibre edilerek mm'ye çevrildi. Yöntemin çıktısı olarak elde edilen ray piksel değerleri sütun grafiğine aktarılmıştır. Grafikteki 0 noktasının görüntüsündeki raylar, 1 noktasındaki sütunların her biri 1 mm'lik genişleme boşluklarını temsil edecek şekilde düzenlenmiştir. Önerilen yöntemin literatürdeki diğer yöntemlere göre en önemli avantajı rayların takibinde trenden bağımsız hareket eden IHA kullanılmasıdır. IHA kullanarak görüntü elde etmek, insan emeğinin olmaması ve trenden bağımsız olarak istenilen her durumda görüntüleme gibi avantajlar sunuyor. Raylar kesilerek kaldırılarak çevresel gürültüler önlenmiş, bu daha sağlıklı ölçüm elde edilmiştir. Farklı mesafelerde genişleme boşlukları olan görüntüler, yöntem çıktıları ve yöntem çıktı sütun grafikleri Şekil 4.88, 4.89 ve 4.90'da verilmiştir.

a) Orijinal görüntü



b) Yöntem çıktısı




c) Yöntem çıktı sütun grafiği

Şekil 4. 88. Normal genişleme boşluğu

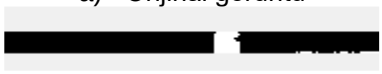
Genişleme aralıkları, rayların montajı sırasında hava sıcaklığı, rayların ulaşabileceği maksimum hava sıcaklığı ve ray demirinin genleşme katsayısı dikkate alınarak belirlenir. 30 metre ray için tesisat hava sıcaklığının 30oC, maksimum ray sıcaklığının 50oC ve demirin uzama katsayısının 1,15 olduğu bölgede bırakılacak genleşme boşluğu miktarı:

- e = (L * t * bir*) / 100
- e = (30 * (50-25) * 1,15) / 100
- e = 5,75 mm
- e = ~ 6mm

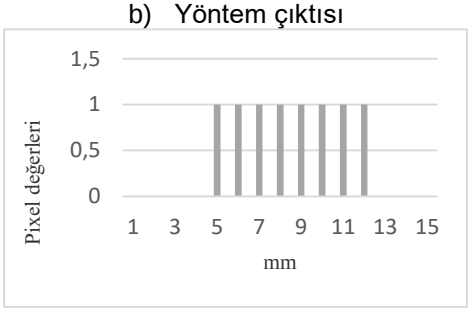
Şekil 4.88.a'daki orijinal görüntü ve 4.88.b'deki yöntemin çıktısı 4.88.c'deki çubuk grafiğe aktarılmıştır. Verilen ray görüntüsü grafiksel olarak incelendiğinde kolon sayısına göre genişleme aralığı 6mm olarak belirlenmiştir. Bu aralığın 30m ray montaj sıcaklığı 30oC ve demir uzama katsayısı 1,15 maksimum 50oC ray sıcaklığında hatasız olduğu belirlendi. Şekil 4.89'daki görüntü aynı koşullar için grafiksel olarak incelendiğinde genişleme aralığı 8mm olarak belirlenmiştir. Bu aralık, belirtilen koşullar için olması gerekenden daha büyük olduğu için hatalı olarak algılandı. Yine aynı koşullar için Şekil 4.90'daki görüntü işlenerek grafiksel olarak analiz edilmiş ve genleşme boşluğu 3mm olarak belirlenmiştir. Belirtilen 3 mm genleşme

aralığı bu koşullar için çok küçük olduğundan kusurlu bir genleşme aralığı olarak belirlenmiştir. Farklı ortam koşullarında geliştirilen yöntem ile aralıkları belirlemek mümkündür.

a) Orijinal görüntü



b) Yöntem çıktısı

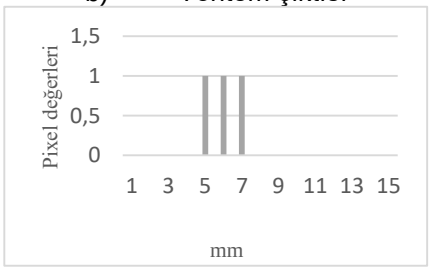



c) Yöntem çıktı sütun grafiği

Şekil 4. 89. Geniş genişleme boşluğu

a) Orijinal görüntü

b) Yöntem çıktısı




c) Yöntem çıktı sütun grafiği

Şekil 4. 90. Küçük genişleme boşluğu

Görüntü işleme sonucunda elde edilen görüntüde ray yüzeyi siyah pikseller ve boşluklar beyaz piksellerle gösterilmiştir. Böylece genleşme boşluklarının tespiti ve ölçümü kolaylaştı. (0,0)

noktasından başlayarak x ekseni boyunca görüntünün x ekseni boyunca ilerlemesi ile okunan piksel değerleri (beyaz için 1, siyah için 0) sütun grafiğine aktarıldı. Görüntüler grafiksel olarak analiz edildi. Genişletme boşlukları, sütun grafiğinde 1. noktada gösterilir. Raylar 0 noktasında gösterilmiştir. Her sütun 1 mm'lik bir genişleme boşluğunu temsil eder. Deneysel sonuçlar, önerilen yöntemin genleşme boşluklarının ölçülmesinde etkili sonuçlar verdiğini göstermiştir.

Tablo 4. 47. Farklı derin öğrenme modellerinin performans karşılaştırma sonuçları

| | **1. çalışma [120]** | **2. çalışma [121]** | **Önerilen yaklaşım** |
|---|---|---|---|
| Veri toplama yöntemi | Drone kamera | Sabit kamera | Drone kamera |
| İncelenen hasar türü | Demiryolu hattı arızalı elemanlar | Genişleme boşluğu | Genişleme boşluğu |
| Önerilen yöntem | Bilgisayarla görme, evrişimli sinir ağları (CNN) | Matematiksel morfoloji, Görüntü Segmentasyonu | Rayların görüntüden belirlenmesi ve kırpılması, İkili formata dönüştürülmesi, Piksel değerlerinin grafiğe aktarılması. |
| Deneysel sonuçlar | Bilgisayarla görme: %96.09, evrişimli sinir ağları (CNN): %69 ila %97 çok katmanlı | Test ettikleri 100 görüntü için %89 doğruluk elde ettiklerini gösterdiler. | Piksel değerlerine göre farklı boyutlardaki genişleme boşluklarının grafik analizi yapılmıştır. |

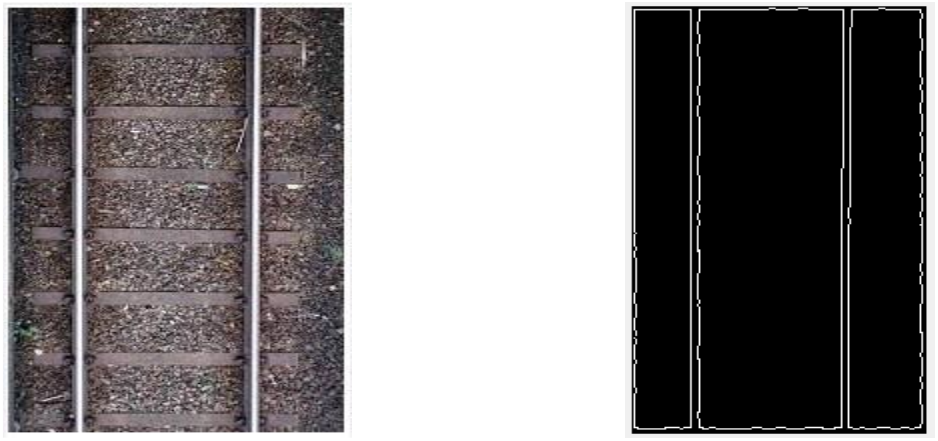
Demiryolu kontrolleri sık sık yapılmalı ve arızalı bileşenler tespit edilip onarılmalıdır. Manuel kontroller yavaştır ve genellikle güvensizdir. Bu nedenle temassız görüntü işleme teknikleri ile kontrollerin yapılması için çalışmalar yapılmaktadır. Düzenli olarak kontrol edilmesi gereken ray elemanlarından biri de genleşme boşluklarıdır. Raylarda sıcaklık nedeniyle bükülmeyi önlemek için genleşme boşlukları bırakılmıştır. Ölçümler düzenli olarak yapılmalıdır.

Bu çalışmada, demiryolu hatlarının belirlenmesi ve hatlarda bırakılan genleşme boşluklarının o bölge için optimum seviyede olup olmadığının tespiti için görüntü işleme tabanlı bir ölçüm

yöntemi önerilmiştir. Bunun için sabit yükseklikte bir drone'dan alınan görüntüler kullanıldı. Elde edilen görüntüler MATLAB'da ön işleme tabi tutulmuş, önce demiryolu hattı belirlenmiş, ardından genişleme boşlukları ölçülmüştür. Önerilen yöntem için ray görüntüleri denenmiş ve sonuçlar grafiksel olarak analiz edilmiştir. Genişleme boşluğu, belirli koşullar için hesaplanan optimum genişleme boşluğuna göre görüntü işleme teknikleri ile ölçülmüştür. Ve grafiksel olarak analiz edildi. Deneysel sonuçlar, önerilen yöntemin demiryolunun raylarının belirlenmesinde ve genleşme boşluğunun ölçülmesinde iyi sonuçlar verdiğini göstermektedir. Önerilen yöntemin en önemli avantajı drone kullanılarak elde edilen görüntülerden genişleme boşluklarını tespit edebilmesi, raylara temas etmeden görüntüleyebilmesi ve tren servislerinden bağımsız olarak görüntüleyebilmesidir. Drone ile görsel veri elde etmek insan emeğine olan bağımlılığı da azaltmaktadır.

### 4.13. Görüntü İşleme ile Demiryolu Hat Açıklığının Belirlenmesi İçin Deneysel Sonuçlar

Önerilen yöntemin denenmesi için drone kullanılarak elde edilen demiryolu hattı kayıtları bilgisayara aktarılmıştır. Daha hassas ölçüm sağlamak için kayıttaki istenmeyen titreşimler kaldırılmıştır. MATLAB kullanılarak görüntü kaydından çerçeveler oluşturulmuş ve raylar görüntü işleme yöntemleri ile netleştirilmiştir. Geliştirilen yöntem kaydedilen görüntüler üzerinde test edilmiştir. Rayların belirtildiği ekran çıktılarının bir örneği Şekil 4.91'de verilmiştir. Görüntü işleme ile belirlenen raylar arasındaki hat aralıkları ile ilgili hataların tespiti için çalışılmıştır.



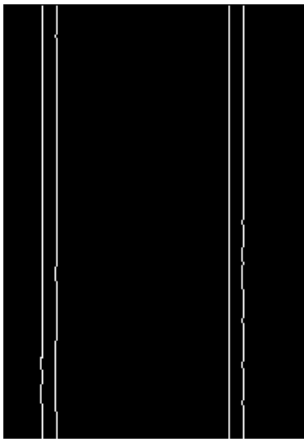
a) Orijinal görüntü b) Keskin kenar algılama

Şekil 4. 91. Önerilen yaklaşma çıktısı – I

Çalışmada aşağıya bakan drone kamera ile izlenen raylardan elde edilen görüntüler kullanıldı. Bu, rayları perspektif bir görünümden ziyade düz olarak görmemizi sağladı. Bu sayede hat açıklığı ölçülürken hata oranı azaltılarak daha sağlıklı ölçümler yapılmasına olanak

sağlanmıştır. Farklı bir ray görünümü ve ekran çıktı örneği Şekil 4.92'de verilmiştir. İz mesafesinin görüntü işleme teknikleri ile belirlenmesine yönelik çalışmalar incelendiğinde farklı yöntemlerin kullanıldığı görülmüştür. Bunlardan biri, trene yerleştirilen kamera ile ray ve bitişiğindeki rayın görüntülerini alarak morfolojik işlemlerle ray aralığını belirlemekti. Çıktılarını grafiksel olarak analiz ettiler. Başka bir çalışmada ise, drone ile farklı yüksekliklerden izlenen demiryolu üzerindeki ray açıklığının GSD hesaplamasını kullanarak ölçülerini belirlediler. Deneysel yöntemlerde her yükseklikte meydana gelen ölçüm hatalarını vermişlerdir. Bu çalışmada, diğer çalışmalardan farklı olarak, sabit bir yükseklikten drone ile görüntülenen demiryolu görüntüleri, titreşim ve ray dışı çevresel gürültülerden arındırılmıştır. Güneşli ve normal hava koşullarında farklı hat açıklığı ölçülerinde elde edilen görüntüler analiz edilmiştir.

a) Orijinal görüntü



b) Keskin kenar algılama

Şekil 4. 92. Önerilen yaklaşma çıktısı – II

Raylar arasındaki iz aralığını belirlemek için piksel değerleri hesaplandı ve mm'ye dönüştürüldü. 1435 mm'lik standart palet açıklığı ile karşılaştırılmıştır. Güneşli ve normal gün ışığında elde edilen demiryolu görüntüleri için işlemler tekrarlanmıştır. Yöntem, güneşli hava için 15 görüntü ve normal gün ışığı için 15 görüntü üzerinde test edildi. Elde edilen sonuçlar Tablo 4.48 ve Tablo 4.49'da verilmiştir.

Tablo 4. 48. Güneşli hava görüntüleri için sonuçlar

| No | Hasarlı/Hasarsız | Gösterge Tipi | Doğruluk(D/Y) |
|---|---|---|---|
| **1** | Hasarsız | Standart | D |
| **2** | Hasarlı | Daralma hatası | D |
| **3** | Hasarsız | Standart | Y |
| **4** | Hasarsız | Standart | D |
| **5** | Hasarsız | Standart | D |
| **6** | Hasarlı | Genişletme hatası | D |

| | | | |
|---|---|---|---|
| **7** | Hasarlı | Genişletme hatası | D |
| **8** | Hasarsız | Standart | Y |
| **9** | Hasarsız | Standart | D |
| **10** | Hasarsız | Standart | D |
| **11** | Hasarsız | Standart | D |
| **12** | Hasarsız | Standart | D |
| **13** | Hasarlı | Genişletme hatası | Y |
| **14** | Hasarsız | Standart | D |
| **15** | Hasarsız | Standart | D |

Tablo 4.48, güneşli havalarda ve bilinen gerçek çizgi boşluklarıyla kaydedilen görüntüler için yapılan ölçümlerden elde edilen sonuçları göstermektedir. Tablonun ilk sütununda kullanılan resmin numarası verilmiştir. İkinci sütunda görselde rayda hasar olup olmadığı, hasar var ise üçüncü sütunda hasar tipi yoksa standart ölçülerde olduğu belirtilir. Ölçülen hat açıklığı 1435 mm ise standart olarak, 1435 mm'den az ise daralma hatası, daha fazla ise genişleme hatası olarak kaydedilir. Dördüncü sütunda yapılan ölçümün doğru olup olmadığı doğru (D) veya yanlış (Y) olarak belirtildi. Örneğin standart boyutta olduğu bilinen görüntü, yöntemin çıktısı olarak standart boyutta ise doğru (D) olarak kaydedilmiştir. Bilinen değerden farklı ise yanlış (Y) olarak kaydedilmiştir. Buna göre güneşli havalarda elde edilen görüntüler için önerilen yöntemin %80 doğrulukla sonuçlar ürettiği belirlenmiştir.

Tablo 4. 49. Normal gün ışığı görüntüleri için sonuçlar

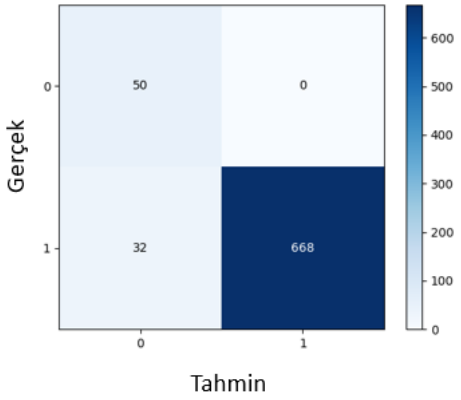
| No | Hasarlı/Hasarsız | Gösterge Tipi | Doğruluk(D/Y) |
|---|---|---|---|
| **1** | Hasarsız | Standart | D |
| **2** | Hasarsız | Standart | D |
| **3** | Hasarlı | Genişletme hatası | Y |
| **4** | Hasarsız | Standart | D |
| **5** | Hasarsız | Standart | D |
| **6** | Hasarsız | Standart | D |
| **7** | Hasarlı | Genişletme hatası | D |
| **8** | Hasarlı | Genişletme hatası | D |
| **9** | Hasarsız | Standart | Y |
| **10** | Hasarlı | Daralma hatası | D |
| **11** | Hasarsız | Standart | D |
| **12** | Hasarsız | Standart | D |
| **13** | Hasarsız | Standart | D |
| **14** | Hasarsız | Standart | D |
| **15** | Hasarlı | Daralma hatası | D |

Tablo 4.49'da, normal gün ışığında kaydedilen ve gerçek hat açıkları bilinen demiryolu görüntüleri için yapılan ölçümlerden elde edilen sonuçlar verilmiştir. Kolonlar Tablo 4.48'de belirtilen sıraya göre tasarlanmıştır. Normal gün ışığında elde edilen görüntüler için önerilen yöntemin %87 doğrulukla sonuç verdiği belirlenmiştir. Önerilen yöntemin nasıl test edildiğini tabloda göstermek için güneşli için on beş resim ve normal gün ışığı için on beş resim kullanıldı. Daha sonra önerilen yöntemin etkinliğini test etmek için kullanılan görsel veri sayısı artırılmıştır. Bunun için Python OpenCv kütüphanesinde yansıtma ve döndürme gibi veri çoğullama teknikleri kullanılmıştır. Güneşli hava için 100 görüntü test edildi ve %86 doğruluk oranı elde edildi. Normal gün ışığı için 100 görüntü test edildi ve %92 doğruluk elde edilmiştir.

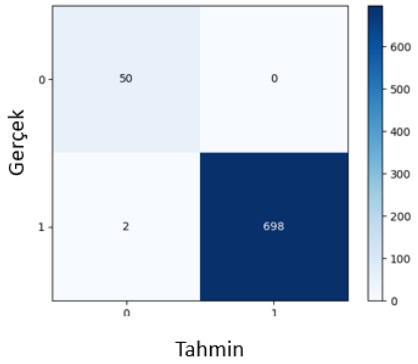
Görüntü işleme ile demiryollarındaki arızaların tespiti ile ilgili çalışmalar her geçen gün artmaktadır. Bu çalışmada, raylar arasındaki ray aralıklarında bazı fiziksel ve çevresel etkilerden dolayı meydana gelen hasarların tespit edilebilmesi için görüntü işleme tabanlı bir model önerilmiştir. Önerilen modelde demiryolu hattı, drone'un trenden bağımsız olarak sabit bir yükseklikten uçması ile kaydedilmiştir. Kayıt sırasında oluşabilecek titreşimler ortadan kaldırılarak çerçeveler oluşturuldu. Görüntülerdeki raylar, MATLAB'da görüntü işleme yöntemleri olan gauss filtreleme ve canny kenar çıkarma algoritmaları kullanılarak belirlendi. Belirtilen raylar arasındaki mesafe piksel sayısına göre ölçülmüştür. Kalibre edilen ve mm'ye dönüştürülen değerler, 1435 mm'lik standart palet açıklığı ile karşılaştırıldı. Raylarda genleşme ve büzülme hataları belirlendi. Yöntem, güneşli ve normal gün ışığında kaydedilen görüntüler için ayrı ayrı test edildi. Deney sonuçlarında verilen doğruluk değerleri güneşli hava için %80, normal gün ışığı görüntüleri için %87 olarak belirlenmiştir. Bu çalışmadaki görüntüler drone kullanılarak elde edilerek zaman ve insan emeğinden tasarruf sağlanmıştır. Drone'nun sabit bir yükseklikten uçabilmesi sağlanarak yükseklik değişikliklerinden kaynaklanan ölçüm hataları engellenmiştir. Ve görüntülerdeki titreşimler kaldırılarak daha hassas ölçüm sağlandı. Bütün bunlar geliştirilen sistemin avantajlı yönleridir. Böylece geliştirilen yöntem ile drone kullanılarak elde edilen demiryolu görüntülerinden farklı hava koşullarında ray açıklığına bağlı olarak yüksek doğrulukta hasar tespiti sağlanmıştır.

### 4.14. Mask R-CNN Kullanılarak Demiryolu Travers Aralığının Ölçülmesi

Önerilen algoritma toplanan görüntüler üzerinde test edilmiştir. Algoritmanın karmaşık çalışma koşullarında ve dinamik ortamlarda tanıma performansını doğrulamak için bu çalışmada daha önce İHA tarafından manuel olarak toplanan demiryolu görüntüleri kullanılmıştır. Önerilen algoritma 700 tanesi hatasız ve 50 tanesi hatalı olmak üzere 750 görüntü üzerinde test edilmiştir. Şekil 4.93, Otsu tabanlı ikili maske ile elde edilen sınıflandırma sonucunu göstermektedir. Şekil 4.94, Mask R-CNN yöntemiyle elde edilen ikili maskeyi kullanan sınıflandırma sonucunu içerir.




Şekil 4. 93. Otsu metodunun sınıflandırma sonuçları




Şekil 4. 94. Mask R-CNN metodunun sınıflandırma sonuçları

Tablo 4.50, Otsu yöntemiyle elde edilen ikili maske ile Mask R-CNN ile elde edilen ikili maskenin performans sonuçlarını karşılaştırır. Tablo 4.50'de görülebileceği gibi, Mask R-CNN daha yüksek doğruluk elde etti.

Tablo 4. 50. İki farklı travers segmentasyon yöntemi için sonuçların karşılaştırılması

| **Segmentasyon** | **Doğruluk** | **Geri Çağırma** | **Kesinlik** | **F1** |
|---|---|---|---|---|
| Otsu Metot | 95,73 | 100 | 95,43 | 97,66 |
| Mask R-CNN | 99,73 | 100 | 99,71 | 99,86 |

# 5. SONUÇLAR VE ÖNERİLER

Bu projede demiryolu alt aksamında oluşan kusurların tespiti için otonom IHA kullanan görme tabanlı bir sistem önerilmiştir. Önerilen sistem ile temassız ve hattı meşgul etmeyecek bir şekilde demiryolu hattından otonom bir IHA ile otomatik veri toplanması için algoritmalar geliştirilmiştir. IHA'nın demiryolu hattı üzerinde otonom bir şekilde uçuşu sağlayabilmesi için hem görüntü işleme hem de derin öğrenme tabanlı yöntemler geliştirilmiştir. Elde edilen görüntülerden ray ve bağlantı elemanlarında oluşan farklı kusurların tespiti için derin öğrenme tabanlı yaklaşımlar geliştirilmiştir. Toplanan verilerden analizlerin otomatik olarak yapılabilmesi için derin öğrenme tabanlı yaklaşımlar önerilmiştir. Ayrıca hazırlanan bir ara yüz üzerinden toplanan veriler üzerinden analizler de yapılmaktadır.

Gazebo simülasyon ortamında oluşturulan bir modeli kullanarak otonom IHA'dan alınan görüntüler ile ray takibi yapabilmek için bir yaklaşım önerilmiştir. Önerilen yaklaşım Gabor filtresi ve kenar çıkarımı yaparak ray tespiti yapmakta ve ufuk noktası tespit edilerek IHA'nın ufuk noktasını takibi sağlanmıştır. IHA'nın konumu ile ufuk noktası arasındaki mesafeyi minimize etmek için PID denetleyici kullanılmıştır. Simülasyon ortamında Parrot Anafi IHA kullanılmıştır. Dolayısıyla gerçek sahada test yapabilmek için kodların sadece fiziksel IHA adresine gönderilmesi yeterlidir.

Proje kapsamında oluşturulan ilk IHA iki adet kameraya sahiptir. Bu kameralar seyrüsefer kamerası ve IHA alt veri toplama kamerasıdır. Geliştirilen IHA'nın ön görüş kamerası ile elde edilen görüntülerden IHA'nın otonom uçuşunu sağlayabilmek için derin Hough dönüşümü tabanlı bir yaklaşım önerilmiştir. Bu yaklaşımın temel avantajı daha belirgin çizgilerin tespit edilmesi sağlanmaktadır. Bu yaklaşım ile sağ ve sol ray bileşenleri tespit edilerek ufuk noktası belirlenmektedir. Bu yöntem kenar çıkarımı ve klasik Hough dönüşümündeki gibi gün ışığı değişimlerinden etkilenemediği için ray takibinin sağlam bir şekilde yapılabilmesi sağlanmıştır. Alt görüş kamerası ile elde e dilen görüntülerden ise ray tespit edilerek ray üzerindeki kusurlar tespit edilmiştir. Ayrıca UNET tabanlı bölütleme algoritması ile ray kusurlarının belirlenmesi için bir yaklaşım sunulmuştur.

Üç farklı ray yüzey kusurunun sınıflandırılması için MobileNetV2 ve SqueezeNet tabanlı hibrit bir derin öğrenme yaklaşımı sunulmuştur. Görüntü ön işleme adımında kullanılan filtreleme ile ray bileşeninin çıkarımı daha doğru bir şekilde yapılmaktadır. Ray kusurlarının belirlenmesi için aynı görüntüler hem MobilenetV2 hem de SqueezeNet ağına verilmiş ve özellik birleştirme yapılarak destek vektör makinalar ile kusur sınıflandırması yapılmıştır. Özellik birleştirme yöntemi tek bir derin öğrenme modeli kullanmaya göre daha iyi sonuçlar vermiştir. Önerilen yaklaşımda oluşturulan veri seti hem GitHub sayfasında hem de proje web sayfasında araştırmacılara sunulmuştur.

Demiryolu genleşme aralıklarının ölçümü için görüntü işleme tabanlı bir yaklaşım önerilmiştir. IHA ile elde edilen görüntülerinden ray tespiti için kenar çıkarımı kullanılmıştır. Daha sonra tespit edilen ray görüntüsünün negatifi alınarak ray açıklığı tespit edilmektedir. IHA'dan alınan görüntülerden hat açıklığının belirlenmesi için de görüntü işleme tabanlı bir yaklaşım sunulmuştur. Önerilen yaklaşımda sağ ve sol raylar kenar çıkarımı ile tespit edilmiş daha sonra iki ray arasındaki mesafe tespit edilerek mesafenin bir eşik değerden büyük olup olmamasına göre ray genişleme arızası belirlenmiştir.

Gazebo ortamında IHA ile ray takibi ve tespiti için gerçek zamanlı çalışan BiseNetV2 tabanlı bir semantik bölütleme yöntemi önerilmiştir. BiseNetV2 bölütleme algoritması Railsem19 veri seti ile eğitilerek eğitilmiş model elde edilmiştir. Gazebo ortamında her iki ray ve rayın ortası bölütlenerek ray tespiti yapılmaktadır. Önerilen yaklaşımın avantajı gerçek zamanlı olarak bölütleme yönteminin çalışmasıdır.

Ray bağlantı elemanlarının tespiti için proje kapsamında yapılan bir çalışmada siyam ağları ile birlikte DCGAN yöntemi ile görüntü arttırma işlemi yapılmıştır. Özellikle kusurlu bağlantı elemanlarının az olduğu durumlarda DCGAN ile yapay görüntüler oluşturularak veri kümesindeki dengesizlik giderilmiştir. Daha sonra Siyam ağı ile görüntülerin birbirlerine olan benzerlik metriğine göre kusur tespiti yapılmıştır.

Ray bağlantı elemanlarının tespiti ve kusur tespiti için düşük ağırlıklı bir evrişimli sinir ağı ile görüntü ön işleme tabanlı bir yaklaşım önerilmiştir. Bu yaklaşımın avantajı ray ve travers çıkarımı için ön işleme adımında kullanılan filtreden dolayı rayın ve traversin kolay bir şekilde tespit edilmesidir. Traversler tespit edildikten sonra elde edilen traverslere çizgi lokal ikili örüntü yöntemi uygulanarak bağlantı elemanının konumu belirlenmekte ve daha sonra elde edilen bağlantı elemanı hafif ağırlıklı evrişimli sinir ağına verilerek kusurlar tespit edilmektedir.

Kusurlu ray bağlantı elemanlarının çoğaltımı için UNet tabanlı bir yaklaşım önerilmiştir. Unet ile bağlantı elemanı bölütlenmekte ve daha sonra bölütlenmiş olan bağlantı elemanının belirli kısımları kesilmektedir. Elde edilen kusurlu bağlantı elemanı bir varsayılan arka plan üzerine yerleştirilerek kusurlu veri üretilmektedir. Daha sonra oluşturulan veri seti farklı transfer öğrenme yöntemleri ve oluşturulan evrişimli sinir ağı modeline verilerek sınıflandırma başarımları ölçülmüştür.

Bağlantı elemanlarındaki problemleri gerçek zamanlı olarak tespit etmek amacıyla YoloV4-Tiny modeli ile birlikte düşük ağırlıklı bir kusur sınıflandırma ağı önerilmiştir. Önerilen modelde öncelikle bağlantı elemanlarının konumu YoloV4-Tiny modeli ile tespit edilmektedir. Yolov4-Tiny modeli daha düşük ağırlıklı olup gerçek zamanlı olarak bağlantı elemanlarını tespit edebilmektedir. Ayrıca bağlantı elemanı kusurlu veya çıkmış da olsa nesne tespit yöntemi bu elemanları da bulmaktadır. Konumları belirlenen bağlantı elemanları kırpılarak MobilenetV2'deki ters çevrilmiş artık blokları kullanan düşük ağırlıklı evrişimsel sinir ağına verilerek kusur durumu belirlenmektedir.

Yolo'nun son dönemde çıkardığı farklı modellerin ray yüzey kusurlarını tespit etmedeki başarımını ölçmek için bir veri seti üzerinde Yolov4, Yolov5 ve Yolov6 modelleri kullanılmıştır. Veri setinde kusurlar dışında ayrıca ray yüzeyindeki kusur ile karıştırılabilecek yağ lekesi vb. durumlar da tespit edilmektedir. Kullanılan Yolo modellerinde Yolov5'in topluluk öğrenme yaklaşımının diğer modellere göre daha iyi sonuç verdiği deneysel olarak ispatlanmıştır.

Ray yüzey kusurlarında oluşan kusurların iki aşamalı olarak tespiti için bulanık ölçüm ve derin öğrenme tabanlı bir yaklaşım sunulmuştur. Bulanık ölçüm yöntemi ile sağlam ray yüzeyinin histogramı Gauss bir fonksiyon olarak modellenmekte ve kusurlu ray için bir bulanık küme oluşturulmaktadır. Daha sonra piksel komşuluklarının üyeliklerine bakılarak piksellerin kusurlu olup olmadığı tespit edilmektedir. Kusur tespiti yapıldıktan sonra kusur tipi MobilenetV2 tabanlı evrişimsel sinir ağı üzerinden transfer öğrenme yöntemi ile belirlenmektedir.

Proje kapsamında önerilen yöntemler ve elde edilen sonuçlar Science Citation Expanded (SCI-E) indeksine giren 2 uluslararası dergide basılmış, iki adet SCI-E dergi makalesi ise değerlendirme aşamasındadır. Yapılan çalışmalardan birinin deneysel sonuçları bitirilmiş olup uluslararası dergi için yazım işlemleri devam etmektedir. Proje süresi boyunca hazırlanan çalışmalardan 11 adeti uluslararası ve 1 adeti ulusal olmak üzere 12 adet konferans çalışması gerçekleştirilmiştir. Ayrıca proje kapsamında geliştirilen yöntemlerden 5 adet Ulusal TR dizin makalesi çıkarılmıştır. Projede 16 ay süresince çalışan tam zamanlı yüksek lisans öğrencisi proje ile ilgili verilen "Gerçek Zamanlı Ray Takibi için Otonom IHA Algoritmalarının Geliştirilmesi ve Gürbüz Derin Öğrenme Tabanlı Kusur Tespiti" isimli tezini Temmuz 2022'de sunmuştur. Projede yarı zamanlı olarak 24 ay süresince çalışan Doktora öğrencisi de proje konusu ile alakalı ""Otonom IHA Tabanlı Demiryolu Bakım Sistemi için Yenilikçi Derin Öğrenme Yöntemlerinin Geliştirilmesi" isimli doktora tez önerisini vermiş olup tez çalışmalarına başlamıştır. Ayrıca iki yüksek lisans öğrencisinin tez önerisi proje konusu ile ilgili olarak verilmiştir. Yüksek lisans öğrencilerinden biri gerekli çalışmaları yapmış olup tez yazımı devam etmektedir. Diğer Yüksek Lisans öğrencisi ise şu ana kadar 2 adet çalışma yapmış olup tez çalışmalarına devam etmektedir.

Projenin son döneminde hem yapılan çalışmaların akademik ve endüstri tarafındaki kişilere anlatılması ve bu alanda farkındalık oluşturmak için bir günlük çalıştay düzenlenmiştir. Çalıştaya 2 farklı firma ve Demiryolları Araştırma ve Teknoloji Merkezinden konuşmacılar katılmıştır. İlk olarak demiryolu araştırma merkezinin faaliyetleri anlatılmış, daha sonra otonom IHA sistemleri ile ilgili yapay zekâ temelli uygulamalardan bahsedilmiştir. Diğer firma ise demiryollarında ray kusurlarının akustik tabanlı tespiti üzerine bir sunum yapmıştır. Son olarak proje kapsamında yapılan çalışmalar ve elde edilen sonuçlar sunulmuştur. Proje süresince yapılan çalışmaların ticarileşmesi için de çalışmalar yapılmıştır. TUBİTAK Big Damla programına ürünün ticarileştirilmesi için başvuru yapılmış fakat proje kabul edilmemiştir. Üniversitemiz Araştırma Üniversitesi olup proje ekibinin Raylı Sistemler konusunda yaptığı

çalışmalardan dolayı Raylı Sistemler üniversitenin öncelikli alanlarından olmuştur. Ayrıca bu kapsamda hazırlanan yüksek bütçeli bir proje Üniversitenin Bilimsel Araştırma birimine sunulmuş ve kabul edilmiştir.

## KAYNAKLAR


**[1]** Tang, R., De Donato, L., Besinović, N., Flammini, F., Goverde, R. M., Lin, Z., ... & Wang, Z. (2022). A literature review of Artificial Intelligence applications in railway systems. *Transportation Research Part C: Emerging Technologies*, *140*, 103679.

**[2]** Zhuang, L., Qi, H., & Zhang, Z. (2021). The Automatic Rail Surface Multi-Flaw Identification Based on a Deep Learning Powered Framework. *IEEE Transactions on Intelligent Transportation Systems*.

**[3]** Debevec, R.(2019). A Smart UAV Platform for Railroad Inspection. Electronic Theses and Dissertations. 6475.

**[4]** Li, H., Wang, F., Liu, J., Song, H., Hou, Z., & Dai, P. (2022). Ensemble model for rail surface defects detection. *PLoS one*, *17*(5), e0268518.

**[5]** Wei, X., Yang, Z., Liu, Y., Wei, D., Jia, L., & Li, Y. (2019). Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study. *Engineering Applications of Artificial Intelligence*, *80*, 66-81.

**[6]** Franca, A. S., & Vassallo, R. F. (2020). A method of classifying railway sleepers and surface defects in real environment. *IEEE Sensors Journal*, *21*(10), 11301-11309.

**[7]** Guan, S., Zhu, Z., & Wang, G. (2022). A Review on UAV-Based Remote Sensing Technologies for Construction and Civil Applications. *Drones*, *6*(5), 117.

**[8]** Sheikh, M., & Örtengren, A. (2018). UAVs for railway infrastructure operations and maintenance activities.

**[9]** Araki, M. (2009). PID control. *Control Systems, Robotics and Automation: System Analysis and Control: Classical Approaches II*, 58-79.

**[10]** Prasanna, S. (2019). Deep learning deployment with NVIDIA TensorRT. *NVIDIA Deep Learning Institute, New York*.

**[11]** Hinton, G. E., Krizhevsky, A., & Sutskever, I. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*(1106-1114), 1.

**[12]** Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv e-prints, page. *arXiv preprint arXiv:1311.2524*.

**[13]** Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

**[14]** Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

**[15]** Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., ... & Wu, J. (2020, May). Unet 3+: A full-scale connected unet for medical image segmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1055-1059). IEEE.

**[16]** Resendiz, E., Hart, J. M., & Ahuja, N. (2013). Automated visual inspection of railroad tracks. *IEEE transactions on intelligent transportation systems*, *14*(2), 751-760.

**[17]** Toliyat, H. A., Abbaszadeh, K., Rahimian, M. M., & Olson, L. E. (2003). Rail defect diagnosis using wavelet packet decomposition. *IEEE Transactions on Industry Applications*, *39*(5), 1454-1461.

**[18]** Zhang, H., Jin, X., Wu, Q. J., Wang, Y., He, Z., & Yang, Y. (2018). Automatic visual detection system of railway surface defects with curvature filter and improved Gaussian mixture model. *IEEE Transactions on Instrumentation and Measurement*, *67*(7), 1593-1608.

**[19]** Zhang, Z., Liang, M., & Wang, Z. (2021). A deep extractor for visual rail surface inspection. *IEEE Access*, *9*, 21798-21809.

**[20]** Yang, H., Wang, Y., Hu, J., He, J., Yao, Z., & Bi, Q. (2021). Deep learning and machine vision-based inspection of rail surface defects. *IEEE Transactions on Instrumentation and Measurement*, *71*, 1-14.

**[21]** Gan, J., Wang, J., Yu, H., Li, Q., & Shi, Z. (2018). Online rail surface inspection utilizing spatial consistency and continuity. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(7), 2741-2751.

**[22]** Tu, Z., Wu, S., Kang, G., & Lin, J. (2021). Real-time defect detection of track components: Considering class imbalance and subtle difference between classes. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1-12.

**[23]** Yang, H., Wang, Y., Hu, J., He, J., Yao, Z., & Bi, Q. (2021). Segmentation of Track Surface Defects Based on Machine Vision and Neural Networks. *IEEE Sensors Journal*, *22*(2), 1571-1582.

**[24]** Zhang, D., Song, K., Wang, Q., He, Y., Wen, X., & Yan, Y. (2020). Two deep learning networks for rail surface defect inspection of limited samples with line-level label. *IEEE Transactions on Industrial Informatics*, *17*(10), 6731-6741.

**[25]** Wu, Y., Qin, Y., Wang, Z., & Jia, L. (2018). A UAV-based visual inspection method for rail surface defects. *Applied sciences*, *8*(7), 1028.

**[26]** Bojarczak, P., & Lesiak, P. (2021). UAVs in rail damage image diagnostics supported by deep-learning networks. *Open Engineering*, *11*(1), 339-348.

**[27]** Wu, Y., Qin, Y., Qian, Y., Guo, F., Wang, Z., & Jia, L. (2022). Hybrid deep learning architecture for rail surface segmentation and surface defect detection. *Computer-Aided Civil and Infrastructure Engineering*, *37*(2), 227-244.

**[28]** Zhong, H., Liu, L., Wang, J., Fu, Q., & Yi, B. (2022). A real-time railway fastener inspection method using the lightweight depth estimation network. *Measurement*, *189*, 110613.

**[29]** Liu, J., Teng, Y., Shi, B., Ni, X., Xiao, W., Wang, C., & Liu, H. (2021). A hierarchical learning approach for railway fastener detection using imbalanced samples. *Measurement*, *186*, 110240.

**[30]** Bai, T., Yang, J., Xu, G., & Yao, D. (2021). An optimized railway fastener detection method based on modified Faster R-CNN. *Measurement*, *182*, 109742.

**[31]** Qi, H., Xu, T., Wang, G., Cheng, Y., & Chen, C. (2020). MYOLOv3-Tiny: A new convolutional neural network architecture for real-time detection of track fasteners. *Computers in Industry*, *123*, 103303.

**[32]** Jing, G., Qin, X., Wang, H., & Deng, C. (2022). Developments, challenges, and perspectives of railway inspection robots. *Automation in Construction*, *138*, 104242.

**[33]** Liu, J., Ma, Z., Qiu, Y., Ni, X., Shi, B., & Liu, H. (2021). Four discriminator cycle-consistent adversarial network for improving railway defective fastener inspection. *IEEE Transactions on Intelligent Transportation Systems*.

**[34]** Liu, J., Teng, Y., Ni, X., & Liu, H. (2021). A fastener inspection method based on defective sample generation and deep convolutional neural network. *IEEE Sensors Journal*, *21*(10), 12179-12188.

**[35]** Flammini, F., Pragliola, C., & Smarra, G. (2016, November). Railway infrastructure monitoring by drones. In *2016 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)* (pp. 1-6). IEEE.

**[36]** Ikshwaku S., Srinivasan A., Varghese A., and Gubbi J., “Railway Corridor Monitoring Using Deep Drone Vision,” doi: 10.1007/978-981-13-1135-2_28.

**[37]** Büyükkelek ,A. F., Dağadası M., Yusefı A., Türkmenoğlu Y. ve A. Durdu, "DJI Tello ile ROS Tabanlı Haritalandırma Simülasyonu", Avrupa Bilim ve Teknoloji Dergisi, c. -, sayı. -, ss. 504-508, Eki. 2020, doi:10.31590/ejosat.820154.

**[38]** Z. He, P. Tang, W. Jin, C. Hu and W. Li, "Deep Semantic Segmentation Neural Networks of Railway Scene," 2018 37th Chinese Control Conference (CCC), 2018, pp. 9095-9100, doi: 10.23919/ChiCC.2018.8483877.

**[39]** Courdier E., Fleuret F. (2021) Real-Time Segmentation Networks Should be Latency Aware. In: Ishikawa H., Liu CL., Pajdla T., Shi J. (eds) Computer Vision – ACCV 2020. ACCV 2020. Lecture Notes in Computer Science, vol 12622. Springer, Cham. https://doi.org/10.1007/978-3-030-69525-5_36.

**[40]** Zendel O., Murschitz M., Zeilinger M., Steininger D., Abbasi S. and Beleznai C., "RailSem19: A Dataset for Semantic Rail Scene Understanding," 2019 IEEE/CVF Conference on Computer Vision

and Pattern Recognition Workshops (CVPRW), 2019, pp. 1221-1229, doi: 10.1109/CVPRW.2019.00161.
**[41]** Yu C., Gao C., Wang J., Yu G., Shen C., and Sang N., “BiSeNet v2: Bilateral network with guided aggregation for real-time semantic segmentation,” 2020, arXiv:2004.02147. [Online]. Available: https://arxiv.org/abs/2004.02147.
**[42]** Rezatofighi H., Tsoi N., Gwak J., Sadeghian A., Reid I., and Savarese S., ‘‘Generalized intersection over union: A metric and a loss for bounding box regression,’’ Feb. 2019, arXiv:1902.09630. [Online]. Available: https://arxiv.org/abs/1902.09630.
**[43]** Wang Y., Wang L., Hu Y. H. and Qiu J., "RailNet: A Segmentation Network for Railroad Detection," in IEEE Access, vol. 7, pp. 143772-143779, 2019, doi: 10.1109/ACCESS.2019.2945633.
**[44]** Belyaev S., Popov I., Shubnikov V., Popov P., Boltenkova E. and Savchuk D., "Railroad semantic segmentation on high-resolution images," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1-6, doi: 10.1109/ITSC45102.2020.9294722.
**[45]** Kaya Ç, Yıldız O. “Makine Öğrenmesi Teknikleriyle Saldırı Tespiti: Karşılaştırmalı Analiz”. Marmara Fen Bilimleri Dergisi.
**[46]** Ferrari, C., & Canny, J. F. (1992, May). Planning optimal grasps. In ICRA (Vol. 3, No. 4, p. 6).
**[47]** Johannes T. (2013). Least-Squares Intersection of Lines. UIUC.
**[48]** Vuola, A. O., Akram, S. U., & Kannala, J. (2019, April). Mask-RCNN and U-net ensembled for nuclei segmentation. In 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019) (pp. 208-212). IEEE.
**[49]** Güçlü E., Aydın İ., Şahbaz K., Akın E., Karaköse M., “Demiryolu bağlantı elemanlarında bulunan kusurların YOLOv4 ve bulanık mantık kullanarak tespiti,” Demiryolu Mühendisliği, no. 14, pp. 249-262, July. 2021. doi: 10.47072/demiryolu.939830
**[50]** Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J. (2019). Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 9157-9166).
**[51]** Duda, R. O. & Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. Comm. ACM, 15, 11–15.
**[52]** Zhao, K., Han, Q., Zhang, C. B., Xu, J. & Cheng, M. M. (2021). Deep hough transform for semantic line detection. IEEE Transactions on Pattern Analysis and Machine Intelligence.
**[53]** Jin, X., Wang, Y., Zhang, H., Zhong, H., Liu, L., Wu, Q.J. & Yang, Y. (2020). DM-RIS: Deep multimodel rail inspection system with improved MRF-GMM and CNN. IEEE Transactions on Instrumentation and Measurement, 69, 1051-1065.
**[54]** Taştimur, C., Karaköse, M., Akın, E. & Aydın, İ. (2016). Rail defect detection with real time image processing technique. 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), 411-415.
**[55]** Gu, G., Ko, B., Go, S., Lee, S. H., Lee, J., & Shin, M. (2021). Towards Real-time and Light-weight Line Segment Detection. arXiv preprint arXiv:2106.00186.
**[56]** Budak O., "SegNet Mimarisi ile Bilgisayarlt Tomografi Gori.nti.lerinden Karaciger Bolgesinin Boli.tlenmesi", Ftrat Oniversitesi Mi.hendislik Bilimleri Dergisi, c. 31, sayt. 1, ss. 215-222, Mar. 2019.
**[57]** F. Yu, V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions." International Conference on Learning Representations (ICLR),2016.
**[58]** Olaf R., Fischer P., Brox T., “U-Net: Convolutional Networks for Biomedical Image Segmentation”, Medical Image Computing and Computer-Assisted Intervention,pp. 234-241,2015.
**[59]** MathWorks (2021, 15 Nisan). Image processing toolbox for MATLAB 2021a, The MathWorks Inc.
**[60]** R.C. Gonzalez, R.E. Woods, S. L. Eddins, S. L. Digital image processing using MATLAB. Pearson Education India, 2004.
**[61]** Mery, D. & Pedreschi, F. (2005). Segmentation of colour food images using a robust algorithm. Journal of Food Engineering, 66(2005), 353-360.
**[62]** Lagarias, J.C., Reeds, J. A., Wright, M. H. & Wright, P. E. (1998). Convergence properties of the nelder-mead simplex method in low dimensions. SIAM Journal of Optimization. 19(1998) 112–147.
**[63]** Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510-4520.
**[64]** Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. abs/1602.07360

**[65]** Sikonja, M. R. & Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. Machine Learning, 53(2003) 23–69.
**[66]** Hastie, T., Tibshirani, R. & Friedman, J. (2008). The Elements of Statistical Learning (İkinci Versiyon). New York: Springer.
**[67]** Tang, L., Tian, Y., Li, W. & Pardalos, P.M. (2020). Structural improved regular simplex support vector machine for multiclass classification. Applied Soft Computing, 91(2020) 106235.
**[68]** Aydın, İ., Karaköse, M. & Akın, E. (2011). A multi-objective artificial immune algorithm for parameter optimization in support vector machine. Applied Soft Computing, 11(2011), 120-129.
**[69]** Platt, J. (1999). Advances in Kernel Methods-Support Vector Learning, chapter Fast training of support vector machines using sequential minimal optimization. MIT Press, 36(1999) 185-208.
**[70]** Li H, Wang F, Liu J, Song H, Hou Z, et al. (2022) Ensemble model for rail surface defects detection. PLOS ONE 17(5): e0268518. https://doi.org/10.1371/journal.pone.0268518.
**[71]** Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0.
**[72]** Akdağ, A. (2017). Derin öğrenme algoritmaları kullanılarak gerçek zamanlı silah tanıma uygulaması, Yüksek Lisans Tezi, Necmettin Erbakan Üniversitesi.
**[73]** Güçlü E., Aydın İ., Şahbaz K., Akın E. ve Karaköse M., "Demiryolu Bağlantı Elemanlarında Bulunan Kusurların YOLOv4 ve Bulanık Mantık Kullanarak Tespiti", Demiryolu Mühendisliği, sayı. 14, ss. 249-262, Tem. 2021, doi:10.47072/demiryolu.939830.
**[74]** Güney, E. (2021). Sürücü asistan sistemleri için mobil gpu tabanlı gerçek zamanlı durum analizi ve tespit uygulamaları, Yüksek Lisans Tezi, Sakarya Üniversitesi.
**[75]** Kotu, Vijay, and Bala Deshpande. “Chapter 2 - Data Science Process.” ScienceDirect, Morgan Kaufmann, 1 Jan. 2019.
**[76]** Conley, Gary, et al. “Using a Deep Learning Model to Quantify Trash Accumulation for Cleaner Urban Stormwater.” Computers, Environment and Urban Systems, vol. 93, Apr. 2022, p. 101752, 10.1016/j.compenvurbsys.2021.101752.
**[77]** R. Padilla, S. L. Netto and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), 2020, pp. 237-242, doi: 10.1109/IWSSIP48289.2020.9145130.
**[78]** Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C.: Mobilenetv2: In-verted residuals and linear bottlenecks. In: Proceedings of the conference on computer vision and pattern recognition, pp. 4510-4520, IEEE, Salt Lake City, Utah (2018).
**[79]** Chollet, F., “Xception: deep learning with depthwise separable convolutions”, In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258), doi: 10.1109/CVPR.2017.195.
**[80]** Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. A., Chen, L. C., “Mobilenetv2: Inverted residuals and linear bottlenecks”, In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520),2018, doi: 10.1109/CVPR.2018.00474.
**[81]** Büyükkelek, A. F., Yusefi, A., Dağadasi, M., Türkmenoğlu, Y. & Durdu, A. (2020). DJI Tello ile ROS Tabanlı Haritalandırma Simülasyonu. Avrupa Bilim ve Teknoloji Dergisi, Özel Sayı, 504-508.
**[82]** 3D Warehouse, Search for railway track | 3D Warehouse. (2021, 27 Şubat). Erişim Adresi: https://3dwarehouse.sketchup.com
**[83]** Y. Wang, L. Wang, Y. H. Hu and J. Qiu, "RailNet: A Segmentation Network for Railroad Detection," in IEEE Access, vol. 7, pp. 143772-143779, 2019, doi: 10.1109/ACCESS.2019.2945633.
**[84]** S. Belyaev, I. Popov, V. Shubnikov, P. Popov, E. Boltenkova and D. Savchuk, "Railroad semantic segmentation on high-resolution images," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1-6, doi: 10.1109/ITSC45102.2020.9294722.
**[85]** Singh, A. K., Dwivedi, A. K., Nahar, N., & Singh, D. (2021, July). Railway Track Sleeper Detection in Low Altitude UAV Imagery Using Deep Convolutional Neural Network. In 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS (pp. 355-358). IEEE.
**[86]** Franca, A. S., & Vassallo, R. F. (2020). A method of classifying railway sleepers and surface defects in the real environment. IEEE Sensors Journal, 21(10), 11301-11309.
**[87]** Sevi M. & Aydın, İ. (2020). COVID-19 Detection Using Deep Learning Methods. 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 1-6.

**[88]** Faghih-Roohi, S., Hajizadeh, S., Núñez, A., Babuska, R. & De Schutter, B. (2016). Deep convolutional neural networks for detection of rail surface defects. 2016 International Joint Conference on Neural Networks (IJCNN), 2584-2589.
**[89]** Shang, L., Yang, Q., Wang, J., Li, S. & Lei, W. (2018). Detection of rail surface defects based on CNN image recognition and classification. 2018 20th International Conference on Advanced Communication Technology (ICACT), 45-51.
**[90]** Zhang, D., Song, K., Xu, J., He, Y., Niu, M. & Yan, Y. (2021). MCnet: Multiple Context Information Segmentation Network of No-Service Rail Surface Defects. IEEE Transactions on Instrumentation and Measurement, 70, 1-9.
**[91]** Wei, X., Yang, Z., Liu, Y., Wei, D., Jia, L. & Li, Y. (2019). Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study. Engineering Applications of Artificial Intelligence, 80, 66-81.
**[92]** Wu, Y., Qin, Y., Wang, Z. & Jia, L. (2018). A UAV-based visual inspection method for rail surface defects. Applied Sciences, 8, 1-20.
**[93]** Min, Y., Xiao, B., Dang, J., Yue, B. & Cheng, T. (2018). Real time detection system for rail surface defects based on machine vision. EURASIP Journal on Image and Video Processing, 1(2018) 1-11.
**[94]** Ye, J., Stewart, E., Zhang, D., Chen, Q. & Roberts, C. (2020). Method for automatic railway track surface defect classification and evaluation using a laser-based 3D model. IET Image Processing, 14, 2701-2710.
**[95]** Zhang, H., Jin, X., Wu, Q. J., Wang, Y., He, Z. & Yang, Y. (2018). Automatic visual detection system of railway surface defects with curvature filter and improved Gaussian mixture model. IEEE Transactions on Instrumentation and Measurement, 67(7), 1593-1608.
**[96]** Li, Q. & Ren, S. (2010). A visual detection system for rail surface defects. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(6), 1531-1542.
**[97]** Ni, X., Liu, H., Ma, Z., Wang, C., & Liu, J. (2021). Detection for rail surface defects via partitioned edge feature. IEEE Transactions on Intelligent Transportation Systems, 1-17.
**[98]** Jin, X., Wang, Y., Zhang, H., Zhong, H., Liu, L., Wu, Q.J. & Yang, Y. (2020). DM-RIS: Deep multimodel rail inspection system with improved MRF-GMM and CNN. IEEE Transactions on Instrumentation and Measurement, 69, 1051-1065.
**[99]** Yu H., et al., A Coarse-to-Fine Model for Rail Surface Defect Detection, in IEEE Transactions on Instrumentation and Measurement, 68(3), 656-666, 2019, doi: 10.1109/TIM.2018.2853958.
**[100]** Yang, H., Wang, Y., Hu, J., He, J., Yao, Z., & Bi, Q.: Deep Learning and Machine Vision-Based Inspection of Rail Surface Defects. IEEE Transactions on Instrumentation and Measurement, 71, 1-14 (2021).
**[101]** Aydin, I., Akin, E., Karakose, M.: Defect classification is based on deep features for railway tracks in sustainable transportation. Applied Soft Computing, 111, 107706 (2021).
**[102]** Zhang, D., Song, K., Wang, Q., He, Y., Wen, X., & Yan, Y.: Two deep learning networks for rail surface defect inspection of limited samples with line-level labels. IEEE Transactions on Industrial Informatics, 17(10), 6731-6741 (2020).
**[103]** Niu, M., Song, K., Huang, L., Wang, Q., Yan, Y., & Meng, Q.: Unsupervised saliency detection of rail surface defects using stereoscopic images. IEEE Transactions on Industrial In-formatics, 17(3), 2271-2281 (2020).
**[104]** Feng, H., Jiang, Z., Xie, F., Yang, P., Shi, J., & Chen, L. (2013). Automatic fastener classification and defect detection in vision-based railway inspection systems. IEEE transactions on instrumentation and measurement, 63(4), 877-888.
**[105]** Liu, J., Huang, Y., Zou, Q., Tian, M., Wang, S., Zhao, X., ... & Ren, S. (2019). Learning visual similarity for inspecting defective railway fasteners. IEEE Sensors Journal, 19(16), 6844-6857.
**[106]** Guo, F., Qian, Y., & Shi, Y. (2021). Real-time railroad track components inspection based on the improved YOLOv4 framework. Automation in Construction, 125, 103596.
**[107]** Dou, Y., Huang, Y., Li, Q., & Luo, S. (2014). A fast template matching-based algorithm for railway bolts detection. International Journal of Machine Learning and Cybernetics, 5(6), 835-844.
**[108]** Liu, J., Teng, Y., Ni, X., & Liu, H. (2021). A Fastener Inspection Method Based on Defective Sample Generation and Deep Convolutional Neural Network. IEEE Sensors Journal, 21(10), 12179-12188.

**[109]** Zhong, H., Liu, L., Wang, J., Fu, Q., and Yi, B., “A real-time railway fastener inspection method using the lightweight depth estimation network,” Measurement, vol. 189, p. 110613, Feb. 2022, doi: 10.1016/j.measurement.2021.110613.

**[110]** Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. A., Chen, L. C., “Mobilenetv2: Inverted residuals and linear bottlenecks”, In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520),2018, doi: 10.1109/CVPR.2018.00474.

**[111]** Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., “Rethinking the inception architecture for computer vision”, In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826), 2016, doi: 10.1109/CVPR.2016.308.

**[112]** Franca, A. S. and Vassallo, R. F., "A Method of Classifying Railway Sleepers and Surface Defects in Real Environment," in IEEE Sensors Journal, vol. 21, no. 10, pp. 11301-11309, 15 May15, 2021, doi: 10.1109/JSEN.2020.3026173.

**[113]** Tu, Z., Wu, S., Kang, G. and Lin, J., "Real-Time Defect Detection of Track Components: Considering Class Imbalance and Subtle Difference Between Classes," in IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-12, 2021, Art no. 5017712, doi: 10.1109/TIM.2021.3117357.

**[114]** Wei, X., Yang, Z., Liu, Y., Wei, D., Jia, L., and Li, Y., “Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study,” Engineering Applications of Artificial Intelligence, vol. 80, no. Complete, pp. 66–81, Apr. 2019.

**[115]** Aydın, İ., Sevi, M., Salur, M. U., and Akın, E., "Defect classification of railway fasteners using image preprocessing and alightweight convolutional neural network," Turkish Journal of Electrical Engineering and Computer Sciences: Vol. 30: No. 3, Article 26. https://doi.org/10.55730/1300-0632.3817.

**[116]** J. Liu et al., "Learning Visual Similarity for Inspecting Defective Railway Fasteners," in IEEE Sensors Journal, vol. 19, no. 16, pp. 6844-6857, 15 Aug.15, 2019, doi: 10.1109/JSEN.2019.2911015.

**[117]** Liu, J., Teng, Y., Ni, X. and Liu, H., "A Fastener Inspection Method Based on Defective Sample Generation and Deep Convolutional Neural Network," in IEEE Sensors Journal, vol. 21, no. 10, pp. 12179-12188, 15 May15, 2021, doi: 10.1109/JSEN.2021.3062021.

**[118]** Jiang, Z., Zhao, L., Li, S., Jia, Y., “Real-time object detection method based on improved YOLOv4-tiny”. arXiv preprint arXiv:2011.04244, 2020, doi: 10.48550/arXiv.2011.04244.

**[119]** Chollet, F., “Xception: deep learning with depthwise separable convolutions”, In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258), doi: 10.1109/CVPR.2017.195.

**[120]** K. Kumar, and A. Kaashyap, ‘’Improving Train Track Safety using Drones, Computer Vision and Machine Learning,’’ arXiv preprint arXiv:2006.11379, 2020.

**[121]** S. Islam, R. A. Khan, and R. Biswas, ‘’Automatic measurement of rail line expansion joint gaps,’’ In 2014 IEEE International Conference on Vehicular Electronics and Safety, pp. 34-39, IEEE, December 2014.