**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# DERİN ÖĞRENME İLE YERE NÜFUZ EDEN RADARLARDA HEDEF TESPİTİ

**YÜKSEK LİSANS TEZİ**

**Fatih KÖPRÜCÜ**

**İletişim Sistemleri Anabilim Dalı**

**Uydu Haberleşmesi ve Uzaktan Algılama**

**ŞUBAT 2021**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# DERİN ÖĞRENME İLE YERE NÜFUZ EDEN RADARLARDA HEDEF TESPİTİ

**YÜKSEK LİSANS TEZİ**

**Fatih KÖPRÜCÜ**
**(705171023)**

**İletişim Sistemleri Anabilim Dalı**

**Uydu Haberleşmesi ve Uzaktan Algılama**

**Tez Danışmanı: Doç.Dr. Işın YAZGAN ERER**

**ŞUBAT 2021**

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 705171023 numaralı Yüksek Lisans Öğrencisi Fatih KÖPRÜCÜ, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "DERİN ÖĞRENME İLE YERE NÜFUZ EDEN RADARLARDA HEDEF TESPİTİ" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

**Tez Danışmanı :** **Doç.Dr. Işın YAZGAN ERER**
İstanbul Teknik Üniversitesi

**Jüri Üyeleri :** **Prof. Dr. Selçuk PAKER** .....
İstanbul Teknik Üniversitesi

**Doç. Dr. Nur Hüseyin KAPLAN** .....
Erzurum Teknik Üniversitesi

**Teslim Tarihi : 22 Ocak 2021**
**Savunma Tarihi : 19 Şubat 2021**

***Aileme,***

## ÖNSÖZ

Tez çalışmalarım sırasında bana her konuda yardımcı olan değerli danışman hocam Doç.Dr. Işın YAZGAN ERER'e teşekkürlerimi sunarım.

Çalışmalarım sırasında desteğini esirgemeyen değerli hocam Dr. Deniz KUMLU'ya teşekkür ederim.

Her zaman yanımda olan ve bana destek olan anne ve babam, Fatma ve Ahmet KÖPRÜCÜ ve kıymetli kardeşlerim Hacer, Emine ve Yeter 'e, ayrıca eğitim sürecim boyunca her zaman motivasyon kaynağım olan nişanlım Tuğçe AY'a

Tüm kalbimle teşekkür ederim...

Ocak 2021 Fatih KÖPRÜCÜ

**İÇİNDEKİLER**

**Sayfa**

## KISALTMALAR

**YNR** : Yere Nüfuz Eden Radar
**SVD** : Singular Value Decomposition – Tekil Değer Ayrıştırması
**PCA** : Principal Component Analysis – Temel Bileşenler Analizi
**RPCA** : Robust Principal Component Analysis – Gürbüz Temel Bileşenler Analizi
**RNMF** : Robust Non-negative Matrix Factorization – Gürbüz Negatif Olmayan Matris Faktörizasyonu
**R-CNN** : Region Based Convolutional Neural Networks – Bölge Tabanlı Evrişimsel Sinir Ağları
**GPR** : Ground Penetrating Radar – Yere Nüfuz Eden Radar
**ESA** : Evrişimsel Sinir Ağı
**İB** : İlişki Bölgesi
**TB** : Tamamen Bağlı
**BÖA** : Bölge Öneri Ağı
**YOLO** : You Only Look Once – Yalnız Bir Kez Bak
**FPN** : Feature Pyramid Network – Öznitelik Piramit Ağı
**COCO** : Common Objects in Context – İçerik Dahilindeki Ortak Nesneler

**SEMBOLLER**

| | |
|---|---|
| **FL** | **:** Focal kayıp |
| **X** | **:** Temel bileşen matrisi |
| **A** | **:** MxN boyutunda bir matris |
| **Φ** | **:** Özvektör |
| **S** | **:** RNMF temel bileşen matrisi |
| **t** | **:** Zaman |
| $\mathbf{X_{hedef}}$ | **:** YNR hedef verisi |
| $\mathbf{X_{kargaşa}}$ | **:** YNR kargaşa verisi |
| **W** | **:** Baz matris |
| **H** | **:** Sabit matris |
| **λ** | **:** Düzenleme parametresi |
| **τ** | **:** İterasyon sayısı |
| **k** | **:** Ayrıştırma kuvveti |

## ÇİZELGE LİSTESİ

**Sayfa**

## ŞEKİL LİSTESİ

**Sayfa**

# DERİN ÖĞRENME İLE YERE NÜFUZ EDEN RADARLARDA HEDEF TESPİTİ

## ÖZET

Derin öğrenme, bilgisayarların işlem kapasitelerinin artması sayesinde birçok alanda kullanılan bir yöntemdir. Her bir kullanım alanının değişkenlik gösteren ihtiyaçlarına bağlı olarak farklı derin öğrenme algoritmaları geliştirilmektedir. Çeşitli problemlere çözüm olarak uygulanabilen derin öğrenme, Yere Nüfuz Eden Radar (YNR) aracılığıyla elde edilen görüntülerde hedef tespitinde de kendine yer bulmuştur. YNR, özellikle belirli bir yüzeye gömülü hedeflerin algılanması, algılanan hedeflerin tanımlanması ve sınıflandırılmasında kullanılan araçlardan bir tanesidir. Farklı zeminlerde gömülü halde bulunan hedeflerin tespitinde, maddelerin elektriksel farklılıklarından yararlanan YNR, bu farklılıklar sonucu oluşan görüntüleri analiz ederek hedef tespitini amaçlamaktadır. Maddelerin elektriksel iletkenliği ayrıştırıcı özellik taşısa da, benzer iletkenlik değerlerine sahip olan maddelerin tespitinde zorluklar yaşanmaktadır. Bunlara ek olarak, çevreden yayılan ve hedef maddenin dışındaki maddelerden yayılan sinyaller ile zeminden kaynaklanan parlamalar hedef algılama problemini güçleştirmektedir. Karşılaşılan bu zorlukların giderilmesi ve etkileyen faktörlerin çeşitliliği nedeniyle derin öğrenme metotlarından faydalanılmıştır. Girdi olarak sağlanan veriler üzerinden öznitelikleri tanımlayarak, yeni verileri anlamlandırma yolunu kullanan derin öğrenme, YNR ile hedef tespiti aşamasında sınıflandırma amacıyla kullanılmıştır. Bu çalışmada hedef algılama problemine çözüm olarak Faster R-CNN, YOLO v3, EfficientDet ve YOLO v5 modellerine odaklanılmıştır.

YNR ile hedef tespitinde karşılaşılan problemlerden biri de çeşitli sebeplerle hedefin dışındaki maddelerden yansıyan sinyallerin oluşturduğu görüntüde oluşan kargaşa verisidir. Bu kargaşaları gidermek ve asıl hedefin ortaya çıkmasını kolaylaştırmak amacıya çeşitli kargaşa giderme yöntemleri kullanılmaktadır. Bu çalışmada kargaşa gidermede kullanılan SVD, PCA, RPCA ve RNMF yöntemlerinden faydalanılmıştır. Bu yöntemlerin derin öğrenme sürecinde hedef tespitine olan etkileri de incelenmiştir. Böylece kargaşa giderme sonrasında oluşan görüntülerden elde edilen sonuçlar gözlenebilmektedir.

Öncelikle ham halde kargaşa giderme uygulanmamış YNR görsellerinden oluşan veri setine sırasıyla SVD, PCA, RPCA ve RNMF yöntemleri uygulanmış ve elde edilen görseller bir araya getirilerek toplamda 5 adet veri seti elde edilmiştir. Sonrasında elde edilen veri setleri eğitim, validasyon ve test olmak üzere üç parçaya bölünerek homojen bir şekilde paylaştırılmıştır. Elde edilen eğitim veri setleri eşit sayıda görüntü barındırmaktadır ve aynı eğitim öncesi işlemlere tabi tutulmuştur. Sonrasında elimizdeki veri setleri Faster R-CNN, YOLO v3, EfficientDet ve YOLO v5 derin öğrenme modellerinin eğitimi için kullanılmış öznitelik çıktıları kaydedilmiştir. Eğitim sonucunda elde edilen derin öğrenme modellerini test veri seti ile test ederek sonuçlar karşılaştırılmıştır. Bu süreç sonunda farklı kargaşa giderme

yöntemleri uygulanmış, YNR verileri ile hedef bulma görevini yerine getiren yirmi farklı algılayıcı model elde edilmiştir. Sonuçlar göz önüne alındığında EfficientDet ve YOLO v5 modellerinin YNR hedef tespitinde üstün performans gösterdiği çıktılarla doğrulanmıştır. Kargaşa giderme yöntemlerinin hedef bulma üzerindeki olumlu etkisi sonuçlar ile aktarılmıştır.

Bu tezin amacı, YNR hedef tespitinde başarı oranının artırılması için derin öğrenme modellerinin kullanılması ve kargaşa giderme yöntemlerinin hedef doğruluğuna etkisinin gözlenmesidir. Bu doğrultuda farklı nesne tanıma modelleri eğitilerek sonuçlar incelenmiştir.

## TARGET DETECTION ON GROUND PENETRATING RADARS (GPR) WITH DEEP LEARNING

## SUMMARY

Deep learning has been actively used in many eras because the computational power of computers increased exponentially. Variety of deep learning algorithms are being developed according to the needs of areas to use. Since deep learning can be used as a solution for many problems, it can be considered as a method for solving target detection problems on Ground Penetrating Radar images. GPR is one of the tools that is especially used for buried object detection, underground target recognition and classification of different object types. GPR uses objects' electrical specialties to distinguish them and according to these differences, it can analyze the results and detect the target objects. Therefore buried objects can be identified on different surfaces.

Matters' varying electrical conductivities is a distinguishable data but some of the matters can exhibit similar conductivity therefore some problems might occur to detect target objects. Addition to that, signals that are revealed from the environment and non-target objects, reflections caused by the surface complicate the target detection problem. Deep learning is utilized to get rid of these challenges to solve different types of detection problems. Deep learning, extracts features by identifying data from input images then gathers information from new data to detect targets. It is used for the purpose of classifying objects on GPR target detection problems.

GPR data can be viewed by three different scan methods: A-scan, B-scan and C-scan. A-scan is one dimensional data consist of a single measured waveform. B-scan is two dimensional view that merged by a set of A-scan data measured along the direction. C-scan is three dimensional data that consists of B-scan data merged along fixed z axis. These scan methods can be used for many different purposes depends on the detection problem. B-scan data occurs as hyperbolas on the image. Therefore, it is easy to distinguish targets on B-scan data by detecting hyperbolas. In this study, B-scan images used for training, validation and test processes. Since the targets appears on GPR images as hyperbolas, detecting the hyperbolas help the detection model to distinguish target from the other components on the image.

In this study, four of the deep learning models are utilized for detecting targets. Two of them are older relatively (Faster R-CNN and YOLO v3) and the others are two of the newest deep learning object detection models (EfficientDet and YOLO v5). These deep learning models are focuses on detecting hyperbolas on the GPR images.

One of the problems that encountered on GPR target detection is clutter data occurred on the image that is caused by signals that reflected from non-target objects. This clutter data makes it difficult to distinguish the target from the whole image. Various clutter removal methods are used to get rid of clutter data and make it easier to detect the main target. Clutter removal methods basically considers the GPR image data as matrix and separates target data form non-target data by using various

equations. In this study, SVD, PCA, RPCA and RNMF methods are utilized as clutter removal techniques. The effects of these clutter removal methods are examined on target detection process with deep learning. Thus, the results of the images that clutter removal methods applied on can be observed separately.

As the first step of the study, one data set created by raw GPR images then four different data sets are created by collecting images that SVD, PCA, RPCA and RNMF clutter removal methods applied on that raw GPR images. Then each data set is segmented into three groups such as train, validation and test. The images are shared as 70%, 20% and 10% respectively. Data sets have the same amount of images and identical pre-training processes are applied on them. The resolution of the each image on the datasets are set as 416x416 pixels. Then targets on the images are labeled and locations of the targets are noted. After that, each data set is used for training of Faster R-CNN, YOLO v3, EfficientDet and YOLO v5 models and feature outputs of the models are saved for test operations. Numerical results of the each training data are noted for comparison purposes. Trained deep learning GPR target detection models are tested on test images and the results are compared. After these processes we maintained twenty different GPR target detection deep learning models that were gained by four different models trained with four different clutter removal methods and one raw GPR data. The test results are shared with both figures and numerically.

The results are compared according to both different deep learning models and clutter removal methods. Object detection test results are shared as images and detected targets are exposed with the labels and confidence scores. Additionally, models' performance compared with numerical values on a table respectively.

As the results considered, it is concluded that clutter removal methods affect the performance of deep learning models positively. If its compared the raw GPR images with clutter removal applied images, it is clearly seen that the targets are more distinguishable on clutter removed images. Therefore this situation has positive effects on the models to understand the patterns of the target and gather the target features. Clutter removal effect is shown both visually and numerically on models performance.

Additionally, when comparing clutter removal methods, it is stated that the models which are trained with newer clutter removal methods applied data set have better performance. SVD and PCA methods are relatively older methods but even the models that are trained with these data sets have better performance than the models trained with raw GPR. RPCA method is one of the newest methods and the performance of the models that trained with RPCA applied data set has better performance than the older methods. However, the best clutter removal method that used in this study for the GPR target detection can be concluded as RNMF.

The deep learning models that are utilized in this study have different structures. Therefore their GPR target detection performances are varying according to that. Faster R-CNN and YOLO v3 models can be considered as traditional methods for object detection relative to EfficientDet and YOLO v5. The results states that, EfficientDet and YOLO v5 models have better performance than R-CNN and YOLO v3 models. Especially, newer models' structure allow them to train and detect faster then the older ones because of their calculation algorithms. EfficientDet and YOLO v5 models can detect targets more precisely and with more confidence score on GPR image data.

As the results considered, the performance of the EfficientDet and YOLO v5 models on GPR target detection noted as better than Faster R-CNN and YOLO v3 models. Additionally, it is shown that the effect of the clutter removal methods on target detection is positive for deep learning models and RNMF method has the best among the clutter removal methods utilized in this study.

The purpose of this study is using deep learning models to improve precision of GPR target detection and to examine the effects of the clutter removal methods on precision of the target detection. Thus different object detection models were trained and the results are analyzed.

# 1. GİRİŞ

Günümüz dünyasında artan veri üretimi ile beraber oluşturulan verilerin işlenmesi de önemli noktalardan biridir. Veri çokluğuna ek olarak verinin anlamlandırılmasında derin öğrenme devreye giren seçeneklerden biri olarak göze çarpmaktadır. Derin öğrenme metotları son dönemde çeşitli sektörlerde yaygın olarak kullanılmaktadır. Birçok alanda tanımlanmış olan problemlerin çözümünde başvurulan yöntemlerden biri olan derin öğrenme, özellikle bilgisayarların işlem kapasitelerinin artması ile beraber popülerlik kazanmıştır. Teorik olarak kökeni 1980'lere [1] dayanmasına rağmen, özellikle son yıllarda hayatımızın her alanında karşılaştığımız derin öğrenme, özellikle görüntü işlemede yüz algılama [2] vb. alanlarda kendine yer bulmuştur.

Derin öğrenme metotlarının kullanımının yaygınlaşması çeşitli problemlerin çözümlerinde derin öğrenmeye başvurulmasının da önünü açmıştır. Gündelik hayatımızda yüz tanıma [3], akıllı araçlar [4] gibi çözümlerin parçası olmakla beraber askeri alanda da hedef algılama [5] gibi konularda sıklıkla kullanılmaktadır. Özellikle oluşturulan görsel üzerinde belirli bir hedef araması yaparken çeşitli derin öğrenme modelleri çözüm olarak sunulmuştur. Temel olarak hedef nesne görsel üzerinde aranırken, tüm görsel üzerinde tarama yaparak hedefe ait parçaların saptanması yoluna gidilir. Derin öğrenme ile nesne algılama modelleri çeşitlilik göstermekle beraber hedefin türüne veya problemin derinliğine bağlı olarak model seçimi yapılabilir. Derin öğrenme mimarisinin karmaşıklığına bağlı olarak hedef tespiti aşamalarında hedefe ait öznitelikler saptanarak hedefin sınıflandırılması gerçekleştirilir.

Nesne tanıma probleminin derin öğrenme ile çözülebilmesi için nesneye özel bir veri seti oluşturulmalıdır. Oluşturulan veri seti birtakım eğitim öncesi işlemlerden geçerek eğitime hazır hale getirilir. Eğitim sırasında nesneyi tanımlayan öznitelikler çıkarılır ve sonraki aşamalarda yeni hedeflerin saptanmasında kullanılmak üzere kaydedilir. Eğitim sonrası elde edilen model verisi kullanılarak yeni görseller üzerinde hedef tespiti yapılması sağlanır. Bu çalışmada nesne tanıma modelleri

olarak Faster R-CNN, YOLO v3, EfficientDet ve YOLO v5 kullanılmıştır. Kullanılan modellere sonraki bölümlerde değinilecektir.

YNR özellikle, engel arkası ve yeraltı görüntülemede sıklıkla kullanılan yöntemlerden biridir. Maddenin elektriksel farklılıklarını kullanarak hedefi içinde bulunduğu ortamdan ayrıştırır. Yaymış olduğu elektromanyetik dalganın hedeften yansımalarını toplayarak anlamlı veri elde etme işini yapar [6]. Verici anten, alıcı anten, kontrolcü ve ekran birimlerinden oluşan YNR cihazı dalganın iletimi, toplanması, verilerin anlamlandırılması ve görsel olarak sunulması işlemlerini yapar. Verici anten yardımıyla gönderilen elektromanyetik dalgaların bir kısmı hedefe erişip yansırken bir kısmı yüzeye çarparak geri döner. Alıcı anten, hedeften yansıyan ve yüzeyden seken dalgaları toplar ve kontrolcüye (controller) iletir. Alıcı antenden gelen dalgalar kontrolcüde dijital bir veri haline getirilir ve ekran birimine aktarılır. Ekran biriminde oluşan veri görselleştirilir. YNR sonucunda oluşan tek boyutlu veriye A-tarama adı verilir. A-taramaların bir araya gelmesiyle oluşan iki boyutlu veriye B-tarama denir. Oluşan çok sayıdaki B-taramaların toplanmasıyla elde edilen üç boyutlu veri C-tarama olarak adlandırılır. Hedef tanıma probleminin türüne bağlı olarak kullanılacak tarama çeşidine karar verilir.

YNR verisi elde edilirken, ölçüm yapılan ortam şartlarına ve hedefin içinde bulunduğu ortama bağlı olarak toplanan sinyallerde bozulmalar yaşanabilmektedir. Hedef dışındaki etmenlerden gelen kargaşa verileri oluşan YNR çıktısını etkilemektedir. Kargaşa verileri içeren YNR verisi içerisinde hedef tanıma güçleşmektedir. Bu nedenle hedef tanıma evresinin hemen öncesinde kargaşa giderme yöntemleri uygulanmaktadır. Kargaşa giderme yöntemleri temel olarak toplam YNR verisi içerisinde, hedef verisi ve kargaşa verisini ayrıştırarak daha sağlıklı bir çıktı elde etmeyi hedefler. Bu çalışmada SVD, PCA, RPCA ve RNMF yöntemleri kargaşa giderme amaçlı kullanılmıştır.

Bu çalışmada, YNR verileri derin öğrenme modellerinin eğitiminde kullanılarak, hedef tanıma problemi ele alınmıştır. Buna ek olarak kargaşa giderme yöntemlerinin derin öğrenme modellerinin eğitimine olan etkileri de araştırılmıştır. Eğitimler sonucunda oluşturulan modellerin doğruluk değerleri karşılaştırılarak, kargaşa giderme yöntemlerinin hedef tanıma problemindeki yeri ele alınmıştır. Çalışmalar sırasında üç farklı veri seti, üç farklı derin öğrenme modeli ile eğitilmiş ve sonuçlar paylaşılmıştır.

## 1.1 Tezin Amacı

Bu çalışma, derin öğrenme modelleri kullanarak YNR verisi üzerinde hedef saptama problemine odaklanmaktadır. Buna ek olarak eğitim öncesi YNR veri setine uygulanan kargaşa giderme yöntemlerinin hedef bulma problemine olan etkisi de gözlenmiştir. Bu amaçlar ışığında farklı derin öğrenme modelleri, Kargaşa giderme uygulanmamış bir adet veri seti ve SVD, PCA, RPCA ve RNMF metotları uygulanmış veri setleri ile eğitilerek performansları karşılaştırılmıştır. Elde edilen sonuçlar ışığında kargaşa giderme yöntemlerinin derin öğrenme modellerinin doğruluğunun artırılmasındaki önemi ortaya konulmuştur.

Bu çalışmanın hedefi, derin öğrenme modelleri kullanılarak YNR verilerinde hedef tanıma probleminin çözümü ve kargaşa giderme yöntemlerinin bu çözüme olan katkılarının araştırılmasıdır.

## 1.2 Tezin Akışı

Bölüm 2 ile beraber derin öğrenme konusuna giriş yapılmaktadır. Derin öğrenme yöntemlerinin kısa bir tarihçesi aktarılmıştır. Hemen sonrasında çalışmalarda kullanılan modeller ve bu modellerin ortaya çıkmasına olanak sağlayan diğer modeller tanıtılmış ve mimariler aktarılmıştır. Bölüm 3'te YNR yöntemine giriş yapılmış ve tarama türlerinden bahsedilmiştir. YNR çalışma prensibi anlatılarak kargaşa giderme problemine değinilmiştir. Sonrasında bu çalışmada kullanılan kargaşa giderme yöntemleri tanıtılarak hesaplama prensipleri aktarılmıştır. Bölüm 4'te seçilmiş olan derin öğrenme modellerinin farklı YNR veri setleri ile eğitim aşamaları anlatılmıştır. Son olarak Bölüm 5'te çıkarılan sonuçlar paylaşılarak yorumlanmıştır.

# 2. DERİN ÖĞRENME İLE NESNE ALGILAMA

## 2.1 Giriş

Nesne tanıma problemi, farklı nesneleri sınıflandırmaya [7] ek olarak, nesnelerin görüntü üzerindeki lokasyonlarının da tahmini esasına dayanır [8]. Nesne algılama yüz algılama [9], yüz tanıma [10] ve yaya algılama [11] gibi birçok alanda da kullanılmaktadır. Bunlara ek olarak birçok alanda anlamlı veri elde etmek için faydalanılan yöntemlerin başında gelir. Son zamanlarda popüler olan sürücüsüz araçlar [12] nesne algılayarak güvenli sürüş sağlarlar. Aynı zamanda sürüş esnasında çevresindeki objelerin hareketlerini de takip ederken nesne tanımına başvururlar [13]. Zamanla nesne tanıma, görüntü sınıflama [14], surat ifadelerinden duygu analizi [15] gibi alanlarda da kendine yer bulmuştur. Bu denli geniş alanlarda kullanılmasına bağlı olarak yeni nesne algılama yöntemleri de ortaya çıkmıştır. Öznitelik çıkarmada kullanılan yöntemlerin farklılaşması ile beraber algoritmalar çeşitlendirilmiştir [16,17].

Görüntü üzerindeki nesnenin algılanabilmesi için öncelikle nesnenin koordinatlarının saptanması gerekir. Bu çabaya nesne lokalizayonu adı verilir. Sonrasında görüntüde saptanan nesnenin hangi sınıfa ait olduğunun bulunması için sınıflandırma verileri gerekmektedir. Temelde nesne algılama şu süreçlerden oluşur: (i) bilgilendirici bölge seçimi, (ii) öznitelik çıkarma, (iii) sınıflandırma.

### 2.1.1 Bilgilendirici bölge seçimi

Nesne algılamada kullanılan görüntüler içerisindeki nesneler farklı açılarda yer alabildiği gibi farklı boyutlarda da olabilir. Bu nedenle tüm görüntü, farklı boyutlarda oluşturulmuş pencereler kaydırılarak taranır. Bu yöntem sayesinde nesnelerin görüntü üzerindeki olası pozisyonları saptanır. Farklı boyutlardaki pencerelerin tüm görsel üzerinde kaydırılması hesaplama anlamında büyük bir yükü beraberinde getirir. Nesne içermeyen çok sayıda pencere de hesaplamaya dahil edilir. Ancak pencere sayısı azaltıldığında bazı nesnelerin gözden kaçma ve hesaplamaya dahil

edilmeme ihtimali oluşur. Bu durum modelin doğruluğunu negatif yönde etkileyecektir. Bu nedenle farklı boyutlarda pencerelerin kullanımı daha yaygındır.

### 2.1.2 Öznitelik çıkarma

Farklı objelerin tanınabilmesi için nesnelerin görsel özniteliklerinin çıkarılması gerekmektedir. Anlamsal olarak nesne ayrımını sağlayan görsel özellikler ön plana çıkarılmalıdır. Nesnelerin görsel olarak ayrımının yapılmasında Haar-like [18] öznitelikler gibi birçok yöntemden faydalanılır. İnsan beynindeki yapılara benzer olarak, karmaşık hücre yapılarında öznitelikler işlenerek ilgili nesnenin temsili bir yapısı oluşturulur [19]. Ancak öznitelik çıkarılırken kullanılan görüntülerdeki problemler (parlama, arka planlar, kayma vb.) ve nesnenin görüntüsünde oluşabilecek farklılıklar, öznitelik tanımlayıcının doğruluğuna zarar verebilir.

### 2.1.3 Sınıflandırma

Görseldeki hedef nesneyi ait olmadığı diğer sınıflardan ayırmak için sınıflandırıcı kullanılır. Ayrıca sınıflandırıcı, görsel tanımlamayı daha anlamlı ve aydınlatıcı hale getirmekle yükümlüdür. Hedef nesnenin ilişkili sınıfını tahmin ederken çeşitli yöntemlere başvurulur. AdaBoost [20], Destekli Vektör Makinesi (DVM) [21] ve Bozunabilir Parça-tabanlı Model (BPM) [22] sınıflandırma için sıklıkla kullanılır. Özellikle daha yeni bir yöntem olan BPM, nesneye ait olan parçalardaki bozulmaları egale etmekte kullanılır. Nesneyi parçalar halinde ele alarak yüksek doğruluklu tanıma elde etmeyi amaçlar. Sınıflandırma yönteminin doğruluğu hedef tanıma modelinin başarısına doğrudan katkı yapar.

## 2.2 Derin Öğrenme Kısa Tarihçe

Derin öğrenme modelleri isminden de anlaşılacağı üzere derin kurgusal yapılardan oluşur. Sinir ağlarının karmaşıklığı problem çözümünde daha efektif sonuçlara erişilmesini sağlar. Her ne kadar son dönemlerde oldukça sık kullanılan bir yöntem olsa da derin öğrenme kavramı 1940'lara kadar dayanan bir geçmişe sahiptir [23]. Başlangıçta insan beyninin öğrenme yapısı taklit edilerek genel problemlerin çözümünde kullanılmak üzere bir hesaplama modeli amaçlanmıştır. 1986 yılında geriye-yayılım algoritmasının ortaya sürülmesiyle [24] birlikte popülerlik kazanmıştır. Ancak derin öğrenmenin hesaplamalarda kullanılabilmesi için yeterli

teknolojik altyapı olmaması nedeniyle aktif olarak kullanılması güçleşmiştir. 2000'li yıllardaki teknolojik atılımla beraber bilgisayarların işlem gücünün artması derin öğrenmenin uygulanabilmesinde önemli bir pay sahibi olmuştur. 2006 yılından itibaren [25] üzerinde yapılan çalışmalarla beraber derin öğrenme popüler olmuştur. Derin öğrenmenin çeşitli sektörlerde aktif olarak kullanılmaya başlanmasının başlıca sebepleri olarak şunları sayabiliriz: (i) Yüksek performanslı ve paralel olarak hesaplama yapabilen grafik işlemci birimlerinin ortaya çıkması, (ii) Imagenet [26] gibi büyük ölçekli ve çeşitli alanlarda eğitime hazır olarak sunulan görsel veri setlerinin yaygınlaşması, (iii) yapay sinir ağı yapılarında ve eğitim algoritmalarındaki gelişmeler.

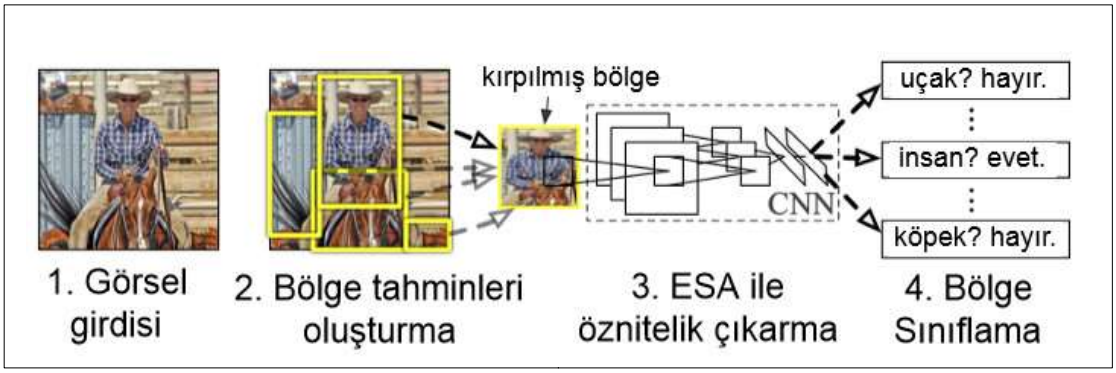
Kısıtlanmış Boltzmann makinesi [27] ve Otomatik Kodlayıcı [28] gibi hızlı öğrenme algoritmalarının geliştirilmesi ile beraber öğrenme sürecinin başlangıç aşamaları daha efektif hale getirilmiştir. Öznitelik çıkarıcıların doğruluğunu yükseltmeyi amaçlayan çalışmalar[29] literatürde yer almıştır. Çok katmanlı derin öğrenme sinir ağları Toplu Normalleştirme [30] ile verimli hale getirilmiştir. Bu gelişmelere ek olarak GoogleNet [31], ResNet [32], VGG [33] gibi yapay sinir ağı mimarilerinin performanslarının geliştirme çalışmaları devam etmiştir.

Bugün gelinen noktada derin öğrenme yöntemlerinin çeşitli alanlarda kullanımına olanak sağlayan yukarıda bahsedilen çalışmalardır. Yalnızca küçük ölçekli veri setlerinde değil, her geçen gün artan verilerin ihtiyacını karşılayacak ve daha büyük ölçekli veri setleri ile de uyumlu olarak çalışabilecek derin öğrenme metotları ortaya çıkmıştır. Yapılan çalışmaların başarısı sayesinde milyonlarca veriyi yüksek doğrulukla işleyebilen modeller ortaya çıkmıştır.

## 2.3 Derin Öğrenme ile Nesne Tanıma Modelleri

### 2.3.1 R-CNN (Region Based Convolutional Neural Network)

R-CNN [34] Ross Girshick tarafından 2014 yılında önerilmiştir. Nesne tanıma aşamaları Şekil 2.1'de gösterildiği üzere dört adımda tamamlanır: (i) Görsel girdisi, (ii) Bölge tahminleri oluşturma, (iii) ESA ile öznitelik çıkarma, (iv) Sınıflama ve lokalizasyon. Sonraki kısımda R-CNN süreçleri gösterilmektedir (Şekil 2.1).




**Şekil 2.1 :** R-CNN Süreçleri [34].

Bölge tahminleri oluşturma gerçekleştirilirken, R-CNN, seçici arama [35] metodunu kullanarak her bir görsel için yaklaşık olarak 2 bin bölge tahmini oluşturur. Seçici arama metodu, farklı boyutlardaki olası hedefleri ele alan pencerelerin doğruluğunu artırmak ve arama yapılan alanı daraltmak amacıyla yukarı yönlü gruplamayı [36] kullanır.

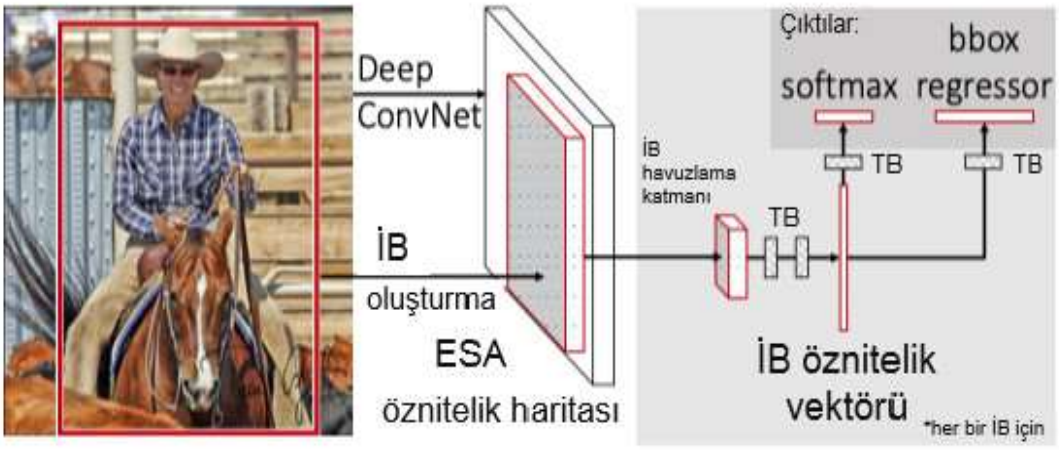
ESA ile öznitelik çıkarım aşamasında, her bir bölge tahmini eğrilerek ya da kırpılarak sabit bir çözünürlük seviyesine getirilir. Sonrasında ESA'dan faydalanarak nesneyi tanımlayan özniteliklerin çıkarımı yapılır. ESA'nın hızlı öğrenme kapasitesi, her bir bölge tahmini için yüksek seviyede anlamsal öznitelik çıkarımına olanak sağlar.

Sınıflama ve lokalizasyon R-CNN modelinin son aşamasıdır. Farklı bölge tahminleri bu aşamada, pozitif bölge ya da negatif arka plan bölgesi olma durumuna göre puanlanır. Puanlanan bölgeler daha sonra sınırlayıcı pencere regresyonu ile ayarlanır ve filtrelenir. Filtrelenerek son halini alan pencereler sayesinde nesnenin lokasyonuna erişilir.

Seçici arama ile bölge kestirimi temelde benzerlik varsayımına dayalıdır. Bu nedenle basit bir lokasyon tahminlemesi yapar. Lokalizasyonun doğruluğunu artırmak için, ek olarak bir sınırlayıcı pencere regresyonu kullanılır. Böylece sınırlayıcı pencerenin merkez koordinatları, genişlik ve yükseklikleri tahmin edilir. R-CNN tarafından kullanılan iki aşamalı algılayıcının her bir bölge tahmini için ayrı ayrı uygulanması hesaplamaların yavaş yürütülmesine sebep olmaktadır. Bu nedenle sonrasında yapılan çalışmalar ile bu probleme çözüm aranmıştır.

### 2.3.2 Fast R-CNN

Fast R-CNN, kendinden önceki R-CNN modelindeki bazı performans sorunlarını gidermek için oluşturulmuştur. R-CNN modelinde çok sayıda bölge tahminleri için tekrarlı olarak hesaplamalar yapılmaktadır. Bu tekrarlı hesaplamaları azaltmak adına Fast R-CNN modelinde değişikliğe gidilmiştir. Bölge tahminlerinin tamamı tek bir görselden elde edilmesi sebebiyle, ESA' nın tüm görsel üzerinde bir kez gezdirilip sonrasında hesaplamaları diğer bölge tahminleri ile paylaşması sağlanarak performans iyileştirilmesine gidilmesi amaçlanmıştır. Ancak farklı bölge tahminlerinin farklı boyutlarda olmasından dolayı, eğer aynı ESA öznitelik çıkarıcı tüm bölgelerde kullanılırsa öznitelik haritaları da farklı boyutlarda olacaktır. Tamamen bağlı katmanlarının sabit boyutlardaki girdilerle çalışmasından dolayı, farklı boyutlardaki öznitelik haritaları, ileri düzeyde sınıflama ve regresyon için kullanılan tamamen bağlı katmanları kullanmamızı engellemektedir. 2015 yılında yapılan bir çalışma [37] ile, tamamen bağlı katmanlarının dinamik olarak ayarlanma problemine çözüm bulunmuştur.




**Şekil 2.2 :** Fast R-CNN Süreçleri [30].

Fast R-CNN, İlişki Bölgesi Havuzlama adı verilen bir katmana sahiptir (Şekil 2.2). Bu havuzlama katmanı öznitelik haritalarını sabit boyutlu bir vektör haline getirir. Böylece tamamen bağlı katmanlarının, ilişki bölgelerinin boyutundan bağımsız olarak, sınıflama ve pencere regresyonunda kullanımına olanak sağlar [38].

İlişki Bölgesi Havuzlama şu adımları izler:

- Önerileri alarak, öneri bölgelerini tanımlar

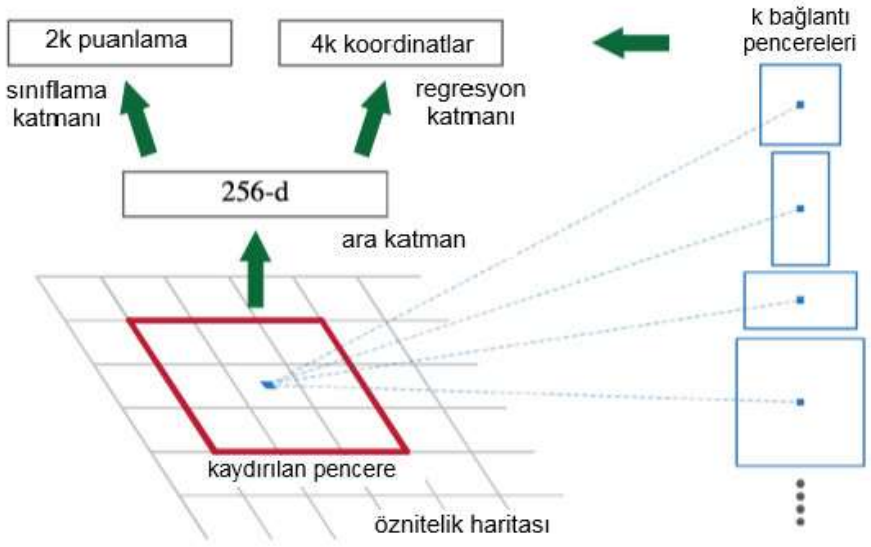
• Bölgelerden sabit sayıda alt-bölgeler oluşturur

• Sabit sayıda çıktı elde etmek için max havuzlama (pooling) uygular.

Fast R-CNN, boyuttan bağımsız ilişki bölgesi havuzlama katmanı ve bölgeler arası öznitelik paylaşımı sayesinde, kendinden önceki model olan R-CNN ile benzer lokalizasyon doğruluğuna sahipken yaklaşık 10 kat daha hızlı eğitilebilmekte ve yaklaşık 100 kat daha hızlı çıkarım yapabilmektedir. Bu performans artışı Fast R-CNN modelini popüler bir seçenek haline getirmiştir.

### 2.3.3 Faster R-CNN

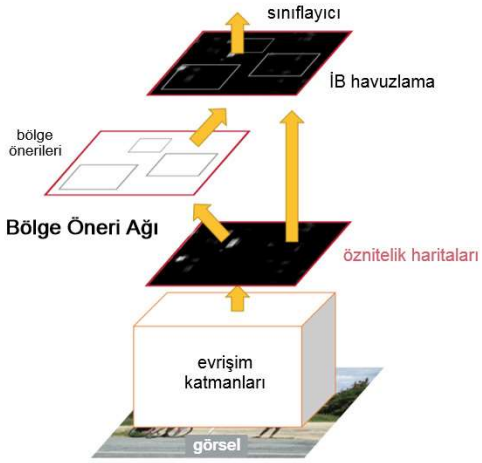
R-CNN ve Fast R-CNN modellerinin ardından, Faster R-CNN modeli ortaya çıkmıştır. Fast R-CNN modelinde seçici arama metodu kullanılarak bölge önerileri oluşturulurken, Fast R-CNN katmanları arasında ESA mimarisi kullanılmadan hesaplama yapılan tek katman olarak göze çarpmıştır. Faster R-CNN modelinde ESA kullanarak seçici arama metodunun işlevini gören bir katman oluşturulmuştur. Bölge Öneri Ağı (BÖA) adı verilen bu katman, temelde görseli girdi olarak alan ve çıktı olarak dikdörtgen şeklinde nesne önerileri veren bir ESA modelidir.




**Şekil 2.3 :** Bölge Öneri Ağı [39].

Bölge Öneri Ağı (BÖA), Faster R-CNN modelinin performansının temel yapı taşı olarak göze çarpar. Bölge önerileri oluşturmak amacıyla 3x3 boyutlarındaki bir pencere ESA öznitelik haritası üzerinde gezdirilir (Şekil 2.3). Bu gezdirme sonucunda ön plan ve arka plan için 2 adet puanlama yapılır ve her lokasyonun 4 koordinatı oluşturulur. Kaydırılan pencere sabit boyutlu olmasına rağmen, hedef

nesneler farklı boyutlarda olabilir. Bu durumda oluşabilecek hataların önüne geçmek adına, Faster R-CNN ile beraber bağlantı penceresi adı verilen bir teknik uygulanmıştır. Bağlantı pencereleri, farklı çerçeve oranlarına sahip ve farklı boyutlarda olan ancak merkezleri ortak olan pencerelerdir. Faster R-CNN modelinde her bir kaydırma penceresi için 3 farklı çerçeve oranında 3 farklı boyutta toplam k=9 tane bağlantı penceresi bulunmaktadır. Bu tekrarlı ve farklı ebatlardaki bağlantı pencereleri, öznitelik haritalarının yönden ve boyuttan bağımsız olmasını sağlamaktadır. Bağlantı pencereleri aynı öznitelik haritasının çıktılarını paylaşırlar. İşlem sonrasında sınırlayıcı pencere regresyonu, tüm görsel üzerinden değil bağlantı pencereleri üzerinden hesaplanır. Bu durum performansa olumlu etki sağlamaktadır.



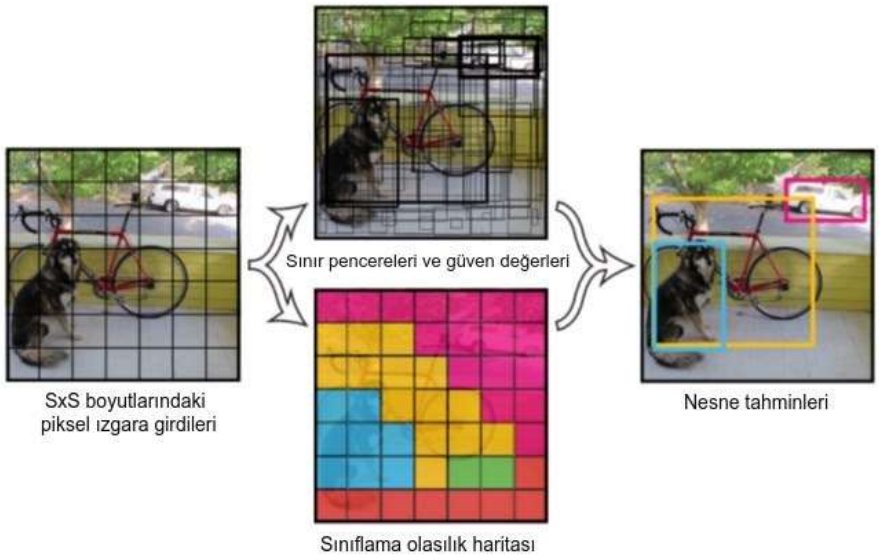

**Şekil 2.4 :** Faster R-CNN işleyiş şeması [39].

Bölge öneri ağı, önceki modellerde kullanılan seçici arama metodunun yerini almıştır (Şekil 2.4). Son tahmini gerçekleştirmek için Faster R-CNN, bir önceki model olan Fast R-CNN ile aynı mimariyi kullanır. BÖA, aynı zamanda bir öznitelik çıkarıcı ESA görevi de görmesinden dolayı direkt olarak algılayıcı ile paylaşılabilir. Böylece BÖA ve Fast R-CNN algılayıcısı kaynakları paylaşarak birlikte çalışmış olur. BÖA çıktısı olan bölge önerileri, ilişki bölgesi havuzlama katmanının girdileri olarak kullanılır. BÖA ve Fast R-CNN katmanlarının sırayla eğitildiği bir tam dolaşıma alternatif eğitim adı verilir [39].

Bu çalışmada YNR görüntüleri kullanılarak Faster R-CNN eğitilmiştir. Eğitilen model ile ilgili çıktılar sonraki bölümlerde paylaşılacaktır.

### 2.3.4 YOLO (You Only Look Once)

YOLO modeli tek bir ağ ile doğrudan nesnenin bağlı olabileceği sınıf olasılıklarını ve sınırlayıcı pencereleri tahmin eder. Basit yapısı sayesinde nesne tanıma hızı yüksektir ve gerçek zamanlı uygulamalarda kullanılmaktadır.




**Şekil 2.5 :** YOLO işleyiş şeması [40].

Her bir evrişimsel sinir ağı birden çok sınırlayıcı pencerenin tahminlemesini yapar ve sınıf olasılıklarını bulur (Şekil 2.5). YOLO tüm görsel üzerinde eğitim yapar ve algılama performansını aynı görsel üzerinde optimize eder. Bu yapı geleneksel yöntemlere göre çeşitli avantajlar sağlamaktadır.

İlk olarak, YOLO hedef algılamayı bir regresyon problemi şeklinde çözer. Bu durum karmaşık yapının oluşmasını engelleyerek zamandan kazanç sağlar. Böylece YOLO ile oluşturulan modeller oldukça hızlı bir şekilde sonuç verir.

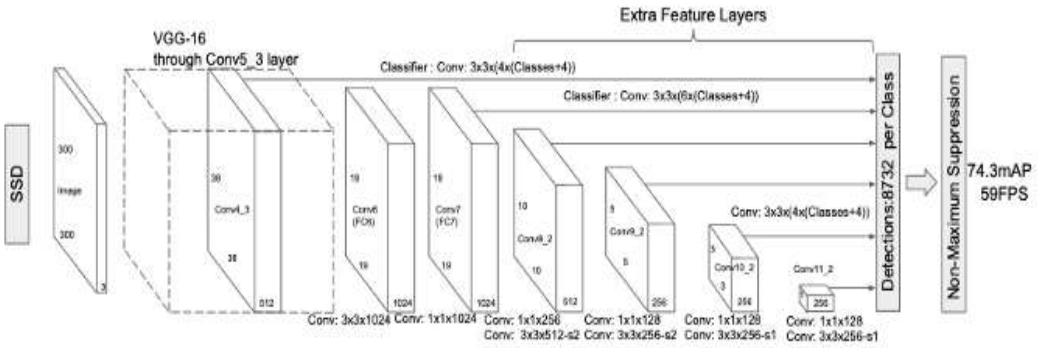
İkinci avantajı olarak, YOLO tahminleme yaparken görseli bütün olarak ele alır. Pencere kaydırma ve bölgesel tahminlemelerden farklı olarak, YOLO eğitim ve test sırasında tüm görsele göre hesaplama yapar. Böylece görüntü üzerindeki tüm sınıfları kodlayarak bağlamsal olarak ilişkili sınıfların bilgisini elde eder. Diğer bölgesel olarak görüntüyü ele alan nesne tanıma yöntemlerinin eksilerinden bir tanesi de görüntünün tamamına erişim sağlayamıyor olmasıdır. Örnek olarak Fast R-CNN, arka plandaki detaylardan etkilenerek algılama doğruluğunun azalmasına sebep olur. Çünkü içeriğin tamamını görememektedir. YOLO bütüne odaklanarak avantaj elde eder.

Üçüncü temel avantajlarından biri olarak YOLO'nun nesnenin tanımını genelleştirebilme özelliğini sayabiliriz. YOLO nesneleri tanırken genel özelliklerini çıkarmaya odaklanmasından dolayı nesneleri genelleştirme yoluna gitmektedir. Bu nedenle doğal görüntüler üzerinde nesne tanıma yaparken çok daha iyi performans sağlar. Genelleştirme özelliği sayesinde, yeni bir görüntü seti üzerinde eğitim yapılırken ya da yeni girdiler ile test edilirken yanılma payı daha azdır.

Model girdi olarak aldığı görseli SxS boyutlarında pixel ızgaralarına (grid olarak adlandırılmaktadır) böler. Izgaraların her bir bölmesi, sınırlayıcı pencereleri doğruluk oranı kullanarak tahmin eder. Elde edilen doğruluk oranları kullanılarak sınıf olasılık haritası oluşturulur. Son aşamada oluşturulan sınıf olasılık haritası doğrultusunda görsel üzerindeki olası nesneler çıktı olarak bildirilir [40].

### 2.3.5 SSD (Single Shot MultiBox Detector)

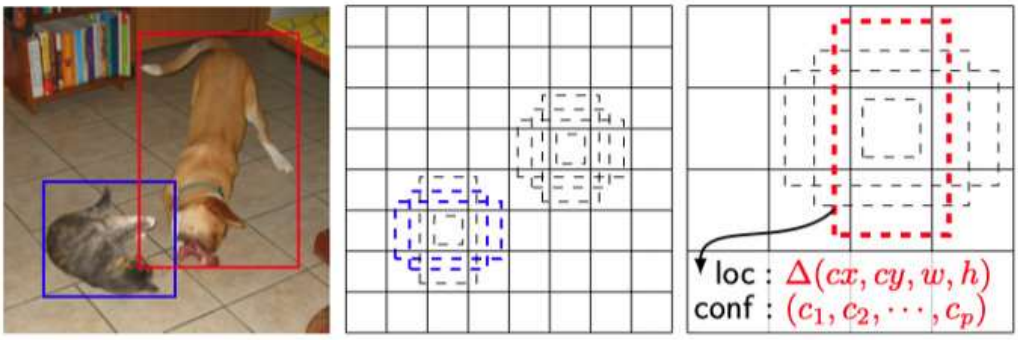
SSD modeli, YOLO gibi tek aşamada nesne algılama prensibine dayanır. Fast R-CNN gibi çok aşamada algılama yapan modellerin performans olarak daha verimli olmasından dolayı SSD modelinde YOLO'dan daha farklı bir algılama süreci izlenmiştir. YOLO modelinde aynı çerçeve içerisinde birden fazla nesne atanabilir haldeydi. Bu durum küçük nesnelerin tespitini güç hale getirirken, tek aşamalı algılayıcıların karşılaştığı ciddi problemlerden biri olarak göze çarpmaktaydı. SSD bu probleme çözüm getirmek amacıyla çeşitli metotlara başvurmuştur. Faster R-CNN tarafından da kullanılan bağlantı penceresi yöntemi bu problemi çözmek için kullanılan metotların başında gelir. Aynı alanda bulunan nesneler farklı boyutlarda olabilirler. Bağlantı pencereleri, her bir hücrede algılanan nesne sayısını artırmakla kalmayıp aynı zamanda üst üste binen küçük nesnelerinde ayrımının yapılmasına katkı sağlar.




**Şekil 2.6 :** SSD işleyiş şeması [41].

SSD modeli bağlantı pencereleri yöntemi ile beraber farklı boyutlardaki özniteliklerden faydalanılmasına olanak sağlamıştır. Tespit aşamasının hemen öncesinde çok boyutlu öznitelikler bir arada hesaplamaya dahil edilmiştir (Şekil 2.6). Çok boyutlu özniteliklerin bir araya getirilmesi farklı sınıflandırma ağlarına kolayca adapte edilebildiğinden dolayı algılayıcı ana mimari diğer modellerle kolaylıkla değiştirilebilir hale gelmiştir. Böylece performansa olan katkısına ek olarak, modelin modülerliği de artırılmıştır.

Özellikle küçük nesnelerin saptanmasını hedefleyen SSD modeli, büyük miktarda veri işlemesi artırımından yararlanmıştır. Küçük nesnelerin görüntü üzerinde algılanmasını kolaylaştırmak amacıyla, görseller rastgele şekillerde genişletilerek büyük boyutlara getirilip sonrasında rastgele kırpma işlemleri uygulanmıştır. Böylece görsel üzerinde farklı şekillerde ortaya çıkabilecek olan küçük nesnelerin algılanmasına yönelik eğitim verisi elde edilmiştir.

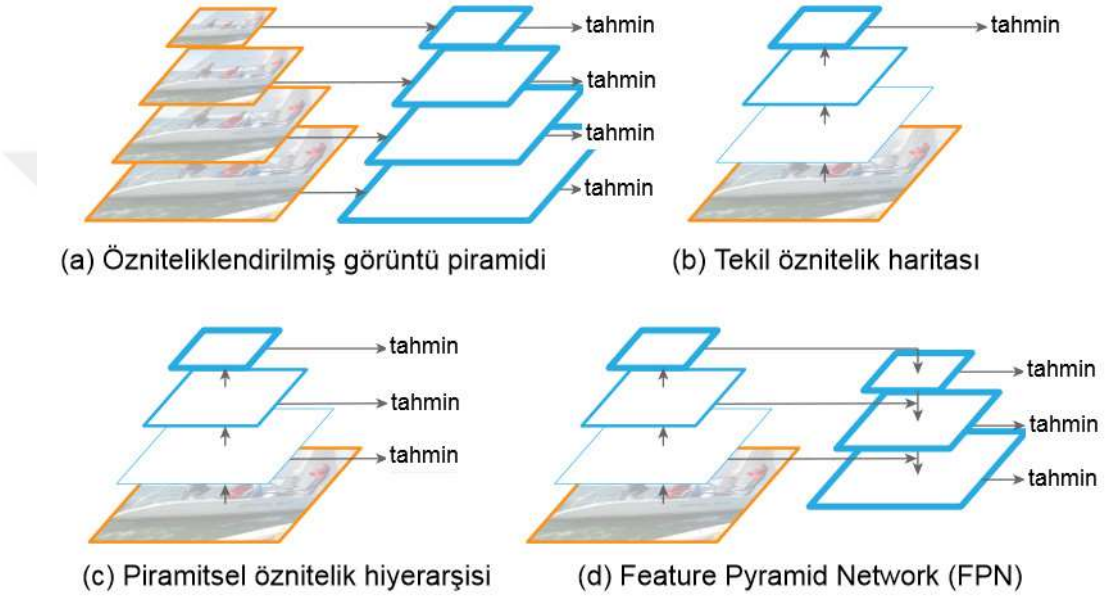



**Şekil 2.7 :** SSD hata saptama [41].

Son olarak SSD, özellikle küçük nesneleri hedefleyen büyük miktarda veri artırmadan yararlandı. Örneğin, rastgele kırpmadan önce görüntüler rastgele olarak çok daha büyük bir boyuta genişletilir ve bu da küçük nesneleri simüle etmek için eğitim verilerine uzaklaştırma efekti getirir. Ayrıca, büyük sınırlayıcı kutuları öğrenmek genellikle kolaydır. Kaybı domine eden bu kolay örneklerden kaçınmak için SSD, her bir bağlantı kutusu için en yüksek kayıplı örnekleri seçmek için sert bir negatif madencilik tekniği benimsedi. Böylelikle büyük boyutlardaki sınırlayıcı pencerelerin daha kolay öğrenebilmesi özelliğinden de faydalanılmıştır. Bu ve benzeri görsel üzerinde yapılan oynamaların toplam algılama hatasına etkisini azaltmak amacıyla SSD modeli, her bir bağlantı penceresindeki en büyük hataya sahip örnekleri filtreleme yoluna gitmiştir (Şekil 2.7).

### 2.3.6 FPN (Feature Pyramid Networks)

FPN, farklı katmanlardaki öznitelikleri kullanarak bir öznitelik piramidi oluşturur [42]. Piramit yapısı daha önce kullanılmış olsa da FPN modeli ile sunulan yapı kendine has özellikler taşımaktadır. Farklı boyutlardaki özniteliklerin çıkarımında kullanılan metotlardan biri olan piramit yapısı, desen tanıma probleminin çözümünde kullanılan efektif yollardan biridir. SSD modelinde olduğu gibi FPN modelinde de sınıflama öncesi farklı katmanlardan gelen öznitelikler beraber kullanılmıştır.




**Şekil 2.8 :** Farklı piramit ağları (Öznitelikler mavi kutucuklar ile gösterilirken anlamsal olarak güçlü öznitelikleri daha koyu ifade edilir) [42].

Şekil 2.8'de görülen yapılar:

(a) Özniteliklendirilmiş görüntü piramidi: Görüntü piramidi kullanarak öznitelik piramidi oluşturulur. Görüntünün boyutundan bağımsız olarak, her bir görüntü için öznitelikler hesaplanır. Yavaş bir süreçtir.

(b) Tekil öznitelik haritası: Daha hızlı algılama için sadece tekil boyutlu öznitelikler kullanılır.

(c) Piramitsel öznitelik hiyerarşisi: (a) ve (b) metotlarının birleşimi gibidir.

(d) FPN: (b) ve (c) kadar hızlı ancak daha yüksek doğruluğa sahiptir.

FPN yapısına sahip bir BÖA inşa etmek için farklı özellik çıktı ölçeğinde çalışan bir bölge önerisine ihtiyaç vardır. Farklı boyutlardaki objeler öznitelik piramidinin farklı katmanlarında ele alınmaktadır. Bu nedenle her bir lokasyon için farklı çerçeve

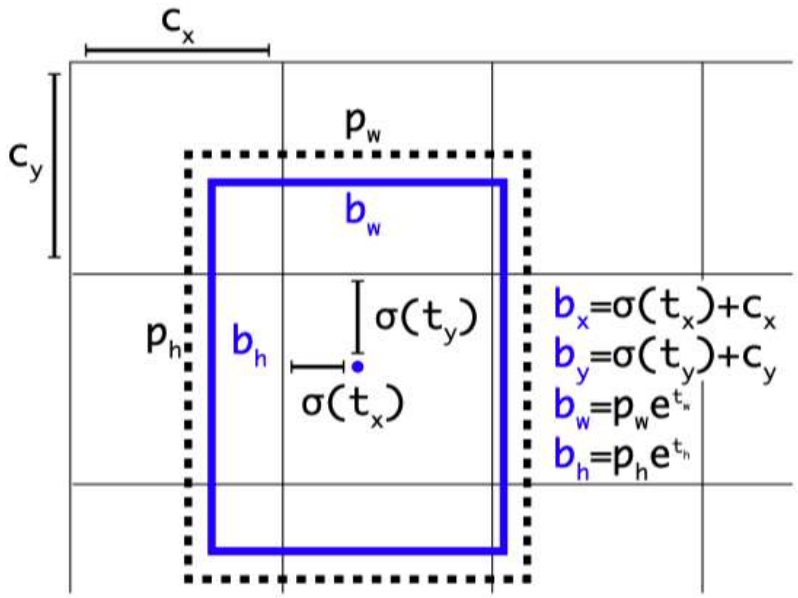
oranlarına sahip 3 tane bağlantı çerçevesi yeterlidir. FPN mimarisini Fast R-CNN algılayıcısıyla beraber kullanabilmek için farklı boyutlardaki öznitelik haritalarıyla çalışabilecek şekilde adapte edilmesi gerekmektedir. Eğer Faster R-CNN tek bir ölçekle çalışan bölge tabanlı bir algılayıcı ve BÖA çiftinden oluşuyor ise, FPN bunu farklı ölçeklerde çalışan çoklu paralel dallara dönüştürür ve sonunda tüm dallardan hesaplama sonuçlarını toplar.

### 2.3.7 YOLO v2

R-CNN modeli üzerinde yapılan geliştirmelere paralel olarak, tek aşamalı YOLO algılayıcısı için performans geliştirmeye yönelik çalışmalar devam etmiştir. YOLO modelinin ilk sürümünün birçok eksikliği ortaya çıkmıştır: ızgara tanımından kaynaklanan lokalizasyon doğruluğunun düşüklüğü göze çarpmaktadır, kullanılan regresörün küçük boyutlu nesneleri tanımayı zorlaştırdığı gözlemlenmiştir [43]. Yapılan geliştirmelerle beraber hem performans hem de hedef doğruluğu için iyileştirmeye gidilmiştir. Göze çarpan temel değişiklikler şunlardır:

- Toplu normalleştirme katmanı [30] eklenmiştir.
- SSD modeline benzer olarak, sınırlayıcı pencere regresyonu için Faster R-CNN modelinde uygulanan bağlantı pencereleri kullanılmaya başlandı (Şekil 2.9). YOLO v2 ile beraber bağlantı pencerelerinde bir takım özelleştirmelere gidildi. YOLOv2 her bir bağlantı penceresinin değerlerini hesaplamak yerine, eğitimin ilk aşamalarını stabilize etmek için ilgili ızgara hücresi içindeki nesnenin merkezi regresyonunu sınırlar.
- Öznitelik çıkarımı için omurga mimarisinde Darknet kullanılmaya başlandı.
- Küçük nesnelerin algılanmasında yaşanan problemlere çözüm olarak, ilk katmanlarda çıkarılan öznitelikleri birleştirmek için geçiş katmanı eklendi. Bu yönü ile SSD modelinin basit bir versiyonu olarak düşünülebilir.
- Farklı çözünürlükteki girdileri farklı eğitim periyotlarında dahil eden bir çok boyutlu eğitim şeması kullanıldı [43].

YOLO v2 9000 farklı sınıf ile test edilmiştir. Böylece nesne algılayıcılar için çoklu etiketli sınıflama çalışmalarının ilk adımı atılmış olarak algılanabilir.



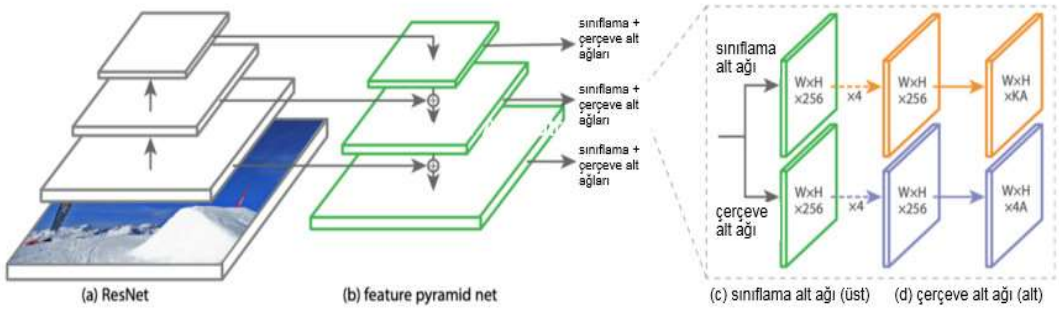

**Şekil 2.9 :** YOLOV2 bağlantı pencereleri [43].

### 2.3.8 RetinaNet

Tek aşamalı algılayıcıların çok aşamalı algılayıcılar karşısında genellikle yetersiz kaldığı gözlemlenmiştir. Bu durumun sebeplerini ortaya çıkarmak üzere RetinaNet modelinde, tek aşamalı algılayıcıların kestirimlerindeki sınıf dengesizliği üzerine çalışılmıştır. Örnek olarak YOLO modelinde, tüm olası lokasyonlarda sınıfların ve sınırlayıcı pencerelerin tahminlemesi anlık olarak yapılır. Bunun sebebi çıktıların çoğunun negatif sınıflarının eğitim sırasında belirlenmesidir. Kısaca ifade etmek gerekirse nesnenin dahil olmadığı sınıflar eğitim aşamasında belirlendiği için algılama aşamasında anında tanıma yapılabilmektedir. YOLO eğitimin ilk aşamalarında ön plan sınıflayıcı eğiterek ön plan - arka plan dengesizliğinin önüne geçmek istemiştir. Buna karşın RetinaNet, Focal Loss [44] adı verilen bir hata denklemi kullanır. Focal Loss hedef tanıma için görsel üzerinde hangi kısımların değerli olduğunu anlamada algılayıcı ağa yardımcı olur.

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{2.1}$$

Denklem 2.1'de ifade edilen Focal Loss hesaplanırken, Cross-Entropy hata denklemine odaklanma parametresi adı verilen bir üstel parametre (γ) eklemiştir. Böylelikle doğruluk oranı arttıkça hata değeri normal Cross-Entropy değerine oranla çok daha düşük olacaktır. Odaklama etkisini dengelemek adına (α) parametresi kullanılır.




**Şekil 2.10 :** RetinaNet [44].

RetinaNet modelinde FPN modeli adapte edilerek yeni bir tek aşamalı algılayıcı mimarisi oluşturulmuştur. ResNet [32] yapısı RetinaNet'in omurgasını oluşturmaktadır (Şekil 2.10). ResNet omurgasının üzerine FPN algılayıcı eklenerek farklı boyutlardaki öznitelikler kanalize edilmiş ve her bir çıktının ucuna sınıflama ve pencere regresyonu ağları yerleştirilmiştir. SSD'ye benzer olarak, çoklu boyutlarda ve çerçeve oranlarındaki hedefleri kapsayacak bağlantı pencereleri kullanılmıştır.

### 2.3.9 YOLO v3

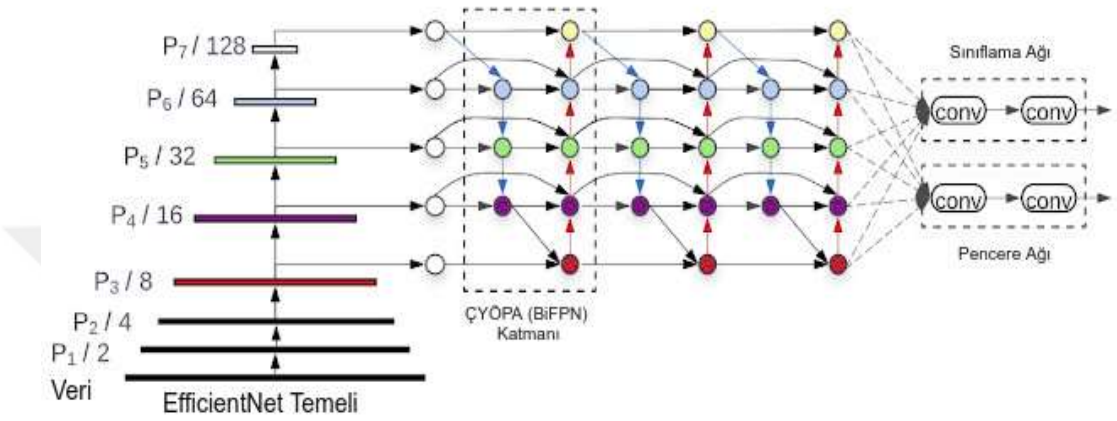
YOLO v3, YOLO model serisinin üçüncü versiyonu olarak göze çarpar. YOLO v2 modelindeki geliştirmeleri takiben yeni çalışmalarla beraber oldukça güçlü bir tek aşamalı algılayıcı ortaya konmuştur. YOLO v3 modeli ile beraber hesaplama hızı, implementasyon karmaşıklığının çözülmesi ve doğruluğun artırılması gibi konularda ilerleme kat edilmiştir. Aynı zamanda hızı ve uygulanabilirliğe beraber çeşitli sektörlerde popüler nesne algılama modellerinden biri haline gelmiştir.

YOLO v3 modelinin yüksek performansının gerekçesi olarak, çok daha güçlü öznitelik çıkarıcı mimarisi ve FPN ile entegre edilmiş RetinaNet' e benzer yapıdaki algılayıcısını gösterebiliriz. Yeni omurga ağı Darknet-53, ResNet-50 ile doğruluk anlamında benzer ancak çok daha hızlı bir algılama elde etmek için ResNet'in atlama bağlantılarından yararlanmaktadır. YOLO v3 modelinde, bir önceki v2 modelinde kullanılan geçiş katmanlarından vazgeçildi. Onun yerine FPN'nin çok ölçekli tahmin tasarımını kullanmaya başladı. Bu yapı ile beraber, YOLO v3 modelinde önceki modellerde yaşanan küçük objelerdeki algılama probleminin giderilmesi sağlandı.

2018 yılında yayınlanan çalışmada [45] belirtildiği üzere, Focal loss, lineer x,y tahminlemesi ve bağlantı penceresi x, y ofset tahminlemesi gibi yöntemler de denenmiş ancak YOLO v3 modelinin performansına olumlu etkileri olmamıştır.

### 2.3.10 EfficientDet

EfficientDet nesne tanımlama problemlerine çözüm olarak sunulmuş yüksek performans ve ölçeklenebilir bir mimari sunan bir modeldir. Daha önce ortaya konan EfficientNet [46] sinir ağı üzerine konumlandırılmıştır. Ayrıca Çift Yönlü Öznitelik Piramit Ağı yardımıyla ve yeni ölçekleme kurallarıyla yüksek doğruluklu ve daha az hesaplama gücüne ihtiyaç duyan bir model oluşturmayı hedeflemiştir.



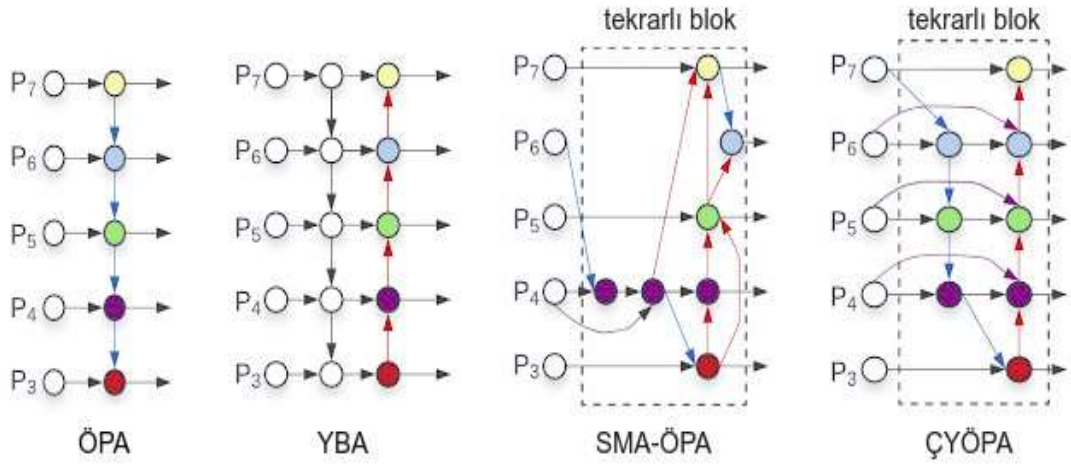

**Şekil 2.11 :** EfficientDet mimarisi [47]. EfficientDet temelinde EfficientNet ağ yapısını kullanır ve ÇYÖPA (Çift Yönlü Öznitelik Piramit Ağı) katmanı ile doğruluğu artırır.

EfficientDet, kendinden önceki nesne algılama modellerinin hesaplama verimliliğini artırmayı amaçlar. Bu amaca yönelik olarak tasarlanmıştır. Şekil 2.11'de de görüldüğü üzere EfficientDet, diğer nesne tanıma ağları gibi 3 temel yapıdan oluşur. (i) Temel yapı: girdi verisi olarak kullanılan görüntüden öznitelik çıkarımı yapar. (ii) Öznitelik Ağı: Temel Ağ tarafından çıkarılan öznitelikleri girdi olarak kullanır ve görüntünün karakteristik yapısını tanımlayan birleşik öznitelikleri çıktı olarak verir. (iii) Sınıflama/Pencere Ağı: Son olarak ortaya çıkan birleşik özniteliklerden yola çıkarak görüntü içerisindeki tüm nesnelerin sınıflarını ve lokasyonlarını tahmin eder [40].

Nesne tanıma modellerinde kullanılan öznitelik ağlarından olan aşağı yönlü Öznitelik Piramit Ağı, tahminleme yaparken semantik olarak zengin olan katmandan çözünürlüğü yüksek katmanlara geçişi sağlar [42]. Ancak ÖPA tarafından sunulan tek taraflı veri akışına alternatif olarak kullanılan Yol Birleştirme Ağı (YBA - PANet), ek olarak yukarı yönlü bir veri akışı sağlamaktadır [48]. Ancak ek olarak sağlanan bu veri akışı hesaplama sayısını artırmaktadır. Diğer nesne tanıma modellerinde kullanılan öznitelik ağlarından biri olan Sinir Mimarili Arama-

Öznitelik Piramit Ağı (SMA-ÖPA), piramitsel olarak tanımlanmış olan katmanlar arasındaki optimum yolu bulmayı amaçlar. Düzensiz yapısı nedeniyle sisteme entegre edilmesi zorlaşmaktadır [49].

Yukarıda bahsedilen öznitelik ağlarına alternatif olarak, aşağı ve yukarı yönlü olarak çalışabilen ve farklı katmanlar arasındaki optimum yolu bularak verimli bağlantılar sağlamayı amaçlayan Çift Yönlü Öznitelik Piramit Ağı (ÇYÖPA) [50], EfficientDet yapısına entegre edilmiştir. Şekil 2.12'de farklı piramit yapıları gözlenmektedir.




**Şekil 2.12 :** ÇYÖPA ile diğer öznitelik ağlarının karşılaştırılması. ÇYÖPA öznitelik verilerinin tekrarlı olarak aşağı ve yukarı yönlü (düşük çözünürlüklü P3 katmanından yüksek çözünürlüklü P7 katmanına) akışını sağlamaktadır.

Bu çalışmada EfficientDet modeli kullanılarak YNR görselleri eğitilmiştir. Eğitim sonuçları ve örnek görseller sonraki bölümlerde paylaşılacaktır.

### 2.3.11 YOLO v4

YOLO serisinin dördüncü elemanı olan YOLOv4 Nisan 2020'de ortaya çıkmıştır [51]**.** YOLO modellerinin üzerine katarak ilerleyerek hem performans hem de tahmin doğruluğu konularına odaklanılmıştır.

YOLOv3 modelinin üzerine temel olarak BoF(bag of freebies)[52] ve BoS(bag of specials)yapısı eklenerek veri işleme yapısında gelişime gidilmiştir. Modele eklenen BoF yapısı ile beraber, algılayıcının doğru tahminleme performansı, hesaplama zamanını etkilemeden yükseltilmiştir. Zamandan kayıp olmamasına rağmen hesaplama yapısının karmaşıklığı artırılmıştır.

BoF yapısına ek olarak yeni versiyonda gelen BoS ile beraber, algılama karmaşıklığı az bir miktarda artmasına rağmen modelin nesne algılama performansında önemli

ölçüde artırıma gidilmiştir. Çalışmada belirtildiği üzere YOLOv3 modeline oranla YOLO v4 modelinde ortalama doğruluk yüzde on ve yüzde on iki oranında artış göstermiştir.

### 2.3.12 YOLO v5

YOLOv5 modeli YOLO v4 modelinden 2 ay sonra ortaya çıkmıştır. YOLO v5 modelinin anlatıldığı bir literatür çalışması bulunmamakla beraber kodları açık kaynaklı bir şekilde sunulmuştur. YOLO v5 modeli önceki YOLO modellerinden farklı olarak Darknet yerine Pytorch altyapısı ile sunulmuştur. Temel değişiklikler olarak otomatik öğrenme özelliğine sahip sınırlayıcı pencereler ve mozaik şeklinde kurgulanmış veri işleme mekanizması göze çarpmaktadır.

YOLO v5 modeli ilk olarak Haziran 2020'de bir blog yazısınında sunulmuştur [53]. Çalışmalarda bahsedildiği üzere bir önceki modele oranla çok daha hafif yapıdaki mimarisi ile kod performansı artırılmıştır. Doğruluk performansı olarak YOLO v4 modeline benzer çıktılar verse de hesaplama hızı ve az yer kaplayan boyutu ile önceki modellerin önüne geçmiştir.

Bu çalışma dahilinde yer alan her bir veri seti ile birer adet YOLO v5 modeli eğitilerek sonuçları paylaşılmıştır.

# 3. YERE NÜFUZ EDEN RADAR

## 3.1 Giriş

Yere Nüfuz Eden Radar (YNR), maddelerin elektriksel özelliklerinden faydalanarak yer altındaki hedeflerin algılanmasında kullanılan araçlardan biridir [54]. Elektriksel farklılıklar sonucu oluşan radar görüntüleri nesnelerin tespiti ve sınıflandırmasında kullanılmaktadır. Her madde için farklı sinyal çıktıları elde etmesi sebebiyle maddelerin ayrımına da olanak sağlamaktadır. Kolay uygulanabilir oluşu ve yüksek çözünürlüklü çıktılar sağlaması dolayısıyla arkeolojik çalışmalar, askeri alanlar, coğrafi tanımlamalar gibi çok çeşitli konularda bir araç olarak kullanılmıştır [55]. Kullanım alanlarının yaygınlığı sebebiyle kullanımı sırasında karşılaşılan problemler de çeşitlenmektedir. Benzer elektriksel özellikler gösteren maddeler de bu problemlerden biridir. Benzer elektriksel özellik gösteren maddelerin radar görüntülerini ayrıştırmak oldukça zor olmakla beraber çevreden yayılan hedef dışı sinyaller de sonuçların doğruluğunu etkilemektedir.

**Şekil 3.1 :** YNR cihaz örneği.

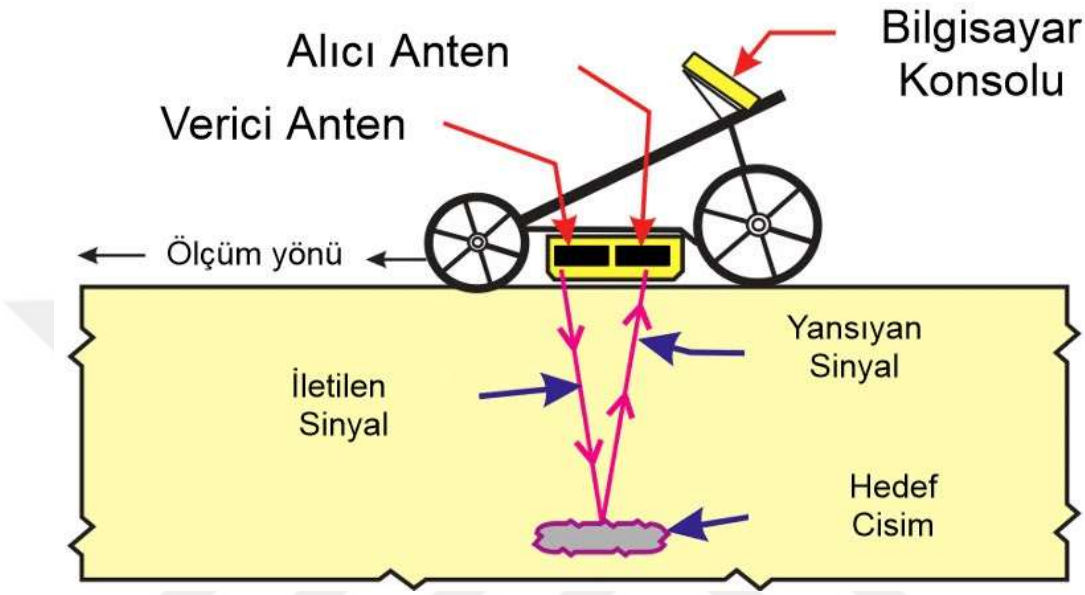
YNR gömülü nesnenin tespitinde sıklıkla kullanılmaktadır. Şekil 3.1'de YNR cihaz örneği gösterilmiştir. Kara mayınlarının tespit edilmesinde de YNR kullanılmaktadır. Sıcaklık, değişen toprak koşulları ve mayın türlerinin çeşitliliği, tespit işlemini zorlaştırmaktadır. Metal dedektörleri mayın tespitinde kullanılsa da zamanla çok az metal içeren veya hiç metal içermeyen mayınların üretilmesiyle beraber dedektörler yetersiz kalmıştır [56]. Metal dedektörlerinden ayrı olarak YNR, ortamlardaki dielektrik farklılıkları tespit eder. Bu durum metalik karakterde olmayan hedeflerin de algılanmasını sağlar. İçeriğinde metal olmayan ya da çok az miktarda metal bulunduran kara mayınların tespitinde de kullanımına olanak sağlar. YNR, yüksek derinliklerde gömülü ve farklı özelliklerdeki malzemeler ile kullanılabilir ve hızlı sonuçlar vermektedir. Bu durum YNR' nin kullanımını kolaylaştırsa da karmaşık veriler üretmesi ve üretilen verinin yorumlamasının zorluğu nedeniyle birtakım zorlukları mevcuttur. Ayrıca yanlış alarm oluşturma oranının yüksekliği göze çarpmaktadır. Bu dezavantajlarından dolayı YNR kullanarak mayın tespiti güçleşmektedir [57].

YNR sistemi dört temel parçadan oluşur:

(i) Verici anten, (ii) alıcı anten, (iii) kontrol birimi ve (iv) ekran. Verici antenden çıkan sinyaller toprağa iletilir ve iletilen sinyal toprakta hedefle buluşur. İletilen sinyallerin bir kısmı yansır ve hedefe ulaşamadan geri döner. Alıcı anten yansıyan sinyalleri toplayarak kontrol birimine iletir. Kontrol biriminde, alıcı antenden gelen sinyaller dijital sinyale dönüştürülerek ekrana yansıtılır (Şekil 3.2).

Maddenin elektriksel iletkenliğine ve manyetik geçirgenliğine bağlı olarak, elektromanyetik dalganın genliği değişkenlik gösterir. Manyetik geçirgenlik, maddeye uygulanan elektromanyetik alan sonucunda maddede oluşan mıknatıslama yeteneğidir. Manyetik geçirgenliğin yüksek olması genliğe olumsuz yönde etki eder. Manyetik geçirgenliği yüksek olan maddelerde genlik daha fazla zayıflar. Bu durum, yüksek manyetik geçirgenliğin veri kalitesini düşürmesine yol açar. Elektriksel iletkenlik ise dalgaların yayılımı ile ilgilidir. Elektriksel iletkenliği yüksek olan maddeler, elektromanyetik sinyalleri yavaşlatır. Bu sebeple, iletkenliği yüksek maddelerin YNR verileri zayıf üretilir ve sinyalin ulaşabildiği derinlik azalır [58]. Maddenin sahip olduğu özelliklerin oluşan YNR verisinin doğruluğuna olan katkısı kaçınılmazdır.

YNR verileri kullanarak hedef tespiti yaparken hem ölçüm yapılan ortamın hem de nesnenin bulunduğu ortamın elektriksel iletkenlik değerlerindeki farka göre oluşturulur. Elektriksel iletkenlik farkı hedef üzerinden yansıyan sinyallerin farklılaşmasına sebep olur. Hedefin konumu, hedefin bulunduğu ortamın yapısı, hedefin derinliği yansımaya etki eden faktörlerdendir.




**Şekil 3.2 :** YNR işleyiş şeması.

### 3.2 YNR Tarama Türleri

YNR verilerinin gösteriminde genellikle üç farklı tarama tipi kullanılır. A-tarama, B-tarama ve C-tarama olarak adlandırılan bu tarama tipleri çeşitli alanlarda kullanılırlar. A-tarama, YNR ile üretilen tek boyutlu sinyallerdir. Bitişik olan A-tarama sinyallerinin toplanması ile elde edilen iki boyutlu sinyale B-tarama adı verilir. Ve yine bitişik olan B-tarama sinyallerinin oluşturduğu üç boyutlu görüntüye C-tarama adı verilir. Tarama verileri x,y ve z eksenleri ile koordinat düzlemi üzerinde gösterilirken, x ve y eksenleri yerin yüzeyini, z ekseni derinliği ifade eder [59].

#### 3.2.1 A-Tarama

A-tarama ile elde edilen veriler, konum bilgileri bilinen ve gömülü haldeki nesnelerin tanımlanmasında veya gömülü nesne tespitinde kullanılabilir [60]. Denklem (3.1)'de görüldüğü şekilde ifade edilir. Tek boyutlu YNR veri çıktılarını ifade eder.

$$f_A(z) = f(x, y, z)|_{x=x', y=y'} \quad (3.1)$$

Sabit olarak belirlenmiş (x', y') konumundaki, monostatik ya da bistatik anten yardımı ile elde edilen tek bir sinyal ölçümünden A-tarama verisi elde edilir. Şekil 3.3' te örnek bir A-tarama verisi bulunmaktadır. Şekildeki x ekseni ile ifade edilen kısım derinliği ifade eder, y ekseni ise tarama verisinin genliğinin değişimini gösterir.

**Şekil 3.3 :** A-tarama örneği.

### 3.2.2 B-Tarama

B-tarama geniş alanlarda, yeri tam olarak bilinmeyen gömülü nesnelerin tespitini sağlar. Bu anlamda B-tarama nesne tespitinde yaygın olarak kullanılan tarama yöntemlerinden biridir [60]. B-tarama verisi, tarama yönünde ilerleyen antenden gelen, her bir noktada oluşan A-tarama verilerinin birbirine eklenmesi ile oluşur. Tek boyutlu A-tarama verileri toplanarak iki boyutlu olarak ifade edilen B-tarama veri kümesini oluşturur. Denklem 3.2'de oluşan veriler formülize edilmiştir.

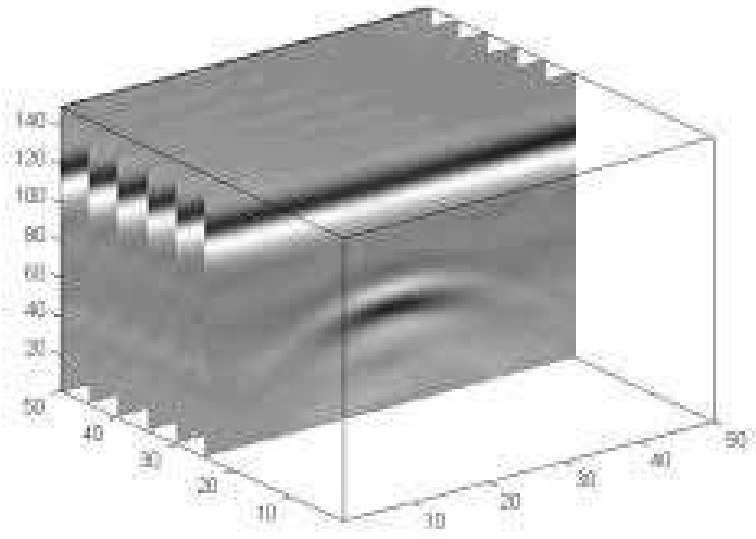
$$f_B(x, z) = f(x, y, z)|_{y=y'} \quad (3.2)$$

B-tarama verisi görselinde, her bir A-tarama verisi için gösterimin türüne göre farklı renk skalası veya gri skala şeklinde sinyalin genliği gösterilir. Şekil 3.4 ile gösterilen örnekte olduğu gibi gömülü hedefler görselde hiperbol oluştururlar.

**Şekil 3.4 :** B-tarama örneği.

B-tarama sonucu oluşan hiperboller hedef algılamada en çok kullanılan veri türlerindendir. Görsel üzerinde hiperbol yakalama yöntemi ile hedef tespiti yapılmaktadır.

### 3.2.3 C-Tarama




**Şekil 3.5 :** C-tarama örneği.

C-tarama ile oluşan veri üç boyutludur ve x, y, z eksenlerinden oluşur. Eksenlerden, x ekseni yatay konumu, y ekseni dikey konumu, z ekseni ise derinliği belirtir. C-

tarama, eş zamanlı olarak oluşturulan, paralel, iki boyutlu B-taramalarının yan yana konumlandırılması ile elde edilir. Şekil 17'de örnek mevcuttur. C- tarama verileri nesne algılamada kullanılabilen tarama türlerinden biridir. Bunun sağlanabilmesi için belirli zaman dilimlerinde toplanan B-tarama verileri içerisinde her bir derinlik seviyesi incelenerek hedef tespit edilmelidir. Ancak C-tarama sırasında, A-tarama ve B-tarama' ya oranla çok daha fazla hesaplama yapılmaktadır ve bu durum işlem süresini yukarılara çeker [60].

### 3.3 Kargaşa Giderme

YNR verisi hedeften gelen sinyallere ek olarak kargaşa(clutter) verilerini de içerir. Kargaşa verileri; direkt sinyaller, yeraltı kargaşa sinyalleri ve diğer hedef dışı gürültü sinyallerinden oluşur [61]. Hedef algılama ve sınıflandırma probleminin çözümünü zorlaştıran bu etmenlere ek olarak hedefin bulunduğu bölgenin coğrafi yapısı da gelen sinyallerin saflığına etki etmektedir [62]. Hedef dışı etmenlerin de etkilediği radar görüntüsü üzerindeki algılama ve sınıflandırma süreçlerine geçmeden önce çeşitli yöntemler uygulanarak görüntünün daha belirgin hale getirilmesi sağlanmaktadır [63]. Özellikle sığ bölgelerde hedef sinyallerini yakalamanın güç olması sebebiyle sinyal iyileştirme, kargaşa giderme gibi yöntemlere başvurulmaktadır. Bu çalışmada SVD, PCA, RPCA ve RNMF yöntemleri uygulanacaktır.

#### 3.3.1 SVD (Singular value decomposition)

SVD yöntemi matris ayrıştırmada kullanılan yöntemlerden biri [64] olmakla beraber YNR B-tarama verilerinin kargaşadan giderilmesinde kullanılır [65]. SVD ile kargaşa giderme uygulanırken, B-tarama verisi MxN boyutlarında bir matris olan *Xij* ile ifade edilir. Bu gösterim sonucunda X değerinin SVD gösterimi Denklem 3.3'teki gibi oluşur.

$$X = U\,S\,V^{T} \tag{3.3}$$

*U* ile ifade edilen MxM boyutlarındaki matrise sol tekil vektör adı verilirken, sağ tekil vektör olan *V*, NxN boyutlarında bir matristir. Aralarında yer alan *S* ise MxN boyutlarındadır. *S* matrisinin diyagonal elemanlarına tekil değerler adı verilir.

X ile ifade edilen YNR verisi 3.4'teki gibi bileşenlerine ayrılarak toplam sembolü ile ifade edilerek matris ayrıştırma işlemine hazır hale getirilir.

$$X = \sum_{i=1}^{N} \sigma_i u_i v_i^T \tag{3.4}$$

SVD uygulanan X değeri Denklem 3.5'teki gibi gösterilir. $M_k$ ile ifade edilen matris X ile aynı boyutlara sahiptir. Böylece $X$ matrisi, farklı değerlerdeki $M_k$ matrisi ile ifade edilebilir hale gelmiştir.

$$X = M_1 + M_2 + ..... + M_N \tag{3.5}$$

$X$ ile ifade edilen YNR B-tarama görselleri kargaşa ve hedef bileşenlerinden oluşur. Denklem 3.6'daki gibi $X$ verisi kargaşa ve hedef bileşenlerinin toplamı olarak ifade edilir. YNR görseli kargaşa ve hedef olarak ayrıştırılır ve $X$ verisi kargaşadan arındırılmış olur [66].

$$X = X_{kargaşa} + X_{hedef} \tag{3.6}$$

### 3.3.2 **PCA (Principal component analysis)**

PCA, kendi aralarında bağımlı olan değişken setlerini lineer olarak bağımsız değişken setlerine dönüştüren bir dikey dönüşüm yöntemidir [67]. YNR görsellerinde kargaşa giderme problemine çözüm olarak kullanılabilir. B-tarama verilerinin oluşturduğu görsel MxN boyutunda $X_{ij}$ formatında dikdörtgen bir matris olarak ifade edilir. Temel bileşen matrisi $X$ ile ifade edilir ve Denklem 3.6 elde edilir.

$$Y = A^T X \tag{3.7}$$

Denklem 3.7'de ifade edilen $A$, MxN boyutunda bir matris olup derece olarak azalan sırada öz vektörlerden oluşur. A = [$\Phi_1$, $\Phi_2$, ..., $\Phi_N$] şeklinde ifade edilirken $\Phi$ özvektörü gösterir ve $\Phi_1$ en büyük özvektörü ifade eder

Temel bileşen matrisini ifade eden S denkleme konulup Denklem 3.8 elde edilir:

$$S = A^T X \tag{3.8}$$

Denklem 3.7'nin uygulanması ile beraber, B-tarama veri görseli kargaşa verisi ve hedef verisi olarak ayrıştırılarak Denklem 3.9'da ifade edilen yapı elde edilir [68].

$$X = X_{kargaşa} + X_{hedef} = A_1^T S_1 + \sum_{i=2}^{N} A_i^T S_i \tag{3.9}$$

İlgili çalışmanın [68] veri setleri düzenlenirken PCA kullanılarak kargaşa giderme uygulanmış görsellere de yer verilmiştir. PCA uygulanan görseller eğitim verisi olarak kullanılarak, eğitimde kargaşa gidermenin etmenlerini araştırma hedeflenmiştir.

### 3.3.3 RPCA (Robust principal component analysis)

RPCA bozulmuş ölçüm verilerinden düşük boyutlu matrisler elde etmek için kullanılan yöntemlerden biridir [69] ve görüntü işleme [70] gibi bir çok alanda da veri düzenleme amaçlı kullanılmıştır. RPCA uygulanırken elde edilen YNR veri matrisi Denklem 3.10'daki şekilde ifade edilir. *X,* YNR verisini, *L* düşük boyutlu matrisi ve *S* ise seyrek matrisi ifade eder. *N* ile gösterilen değer veriye etki eden gürültüyü gösterir

$$X = L + S + N \tag{3.10}$$

D matrisi ayrıştırılarak Denklem 3.11 elde edilir [71]. Tekil değerler toplamını ifade eden operatör $\|\|_*$ ve matris girdilerinin mutlak değerlerinin toplamını ifade eden operatör $\|\|_1$ kullanılarak denklem oluşturulmuştur. Ödünleşme parametresi ise λ ile gösterilmiştir.

$$\underset{L,S}{min} \ \|L\|_* + \lambda \|S\|_1 \quad \text{s.t.} \ \|X - L - S\|_F \leq \gamma \tag{3.11}$$

Denklem 3.10'da verilen optimizasyon problemi çözülerek YNR verisi hedef ve kargaşa bileşenlerine ayrılır. Veri ayrıştırma sonrası hedef görseli S seyrek matrisi içerisinde yer alır ve kargaşa içeren düşük boyutlu matris çıkarılmış olur.

### 3.3.4 RNMF (Robust non-negative matrix factorization)

YNR görseli kargaşa verisi ve hedef verisinin toplamı ile ifade edilebilir [72]. Hedef verisi tüm görsel üzerindeki verilere oranla oldukça küçük bir yer kaplamaktaır. Bu nedenle hedef verisinin kargaşa verisinden ayrıştırılması gerekmektedir. Denklem 3.12'de olduğu gibi RNMF [73] kullanılarak YNR verisi (X), hedef verisi ($X_{hedef}$) ve kargaşa verisi ($X_{karmaşa}$) olarak ayrıştırılır.

$$X = X_{kargaşa} + X_{hedef} \tag{3.12}$$

Hedef verisi, baz matris ($W$) ve sabit matris ($H$) çarpımından elde edilir. Kargaşa verisi ise $S$ ile ifade edilerek Denklem 3.13 elde edilir.

$$X \approx WH + S \tag{3.13}$$

RNMF metodu ile kargaşa giderme işlemi şu şekilde gerçekleştirilir:

- Sürece girdi olarak şunlar verilir:
  - $X$: M x N boyutundaki YNR görseli
  - $W_0$ ve $H_0$: Baz ve sabit matris için başlangıç değerleri
  - $\lambda$: Düzenleme parametresi (Varsayılan değeri $3e^{-4}$ olarak belirlenir.)
  - $t$: Maksimum iterasyon sayısı (Varsayılan değeri 10.000 olarak belirlenir.)
  - $k$: Ayrıştırma kuvveti (Varsayılan değeri 1 olarak belirlenir.)
- $W$, $H$ ve $S$ her bir iterasyonda (3.13) ve (3.14) denklemleri kullanılarak güncellenir. İterasyon süreci, belirlenen maksimum iterasyon sayısına ulaşıldığında ya da önceden belirlenmiş olan bir eşik değerine erişildiğinde sonlanır.
- İterasyonlar sonucunda elde edilen MxN boyutlu S matrisi hedef verisini temsil eder.
- Sürecin çıktısı olarak YNR görselindeki kargaşa verisi($X_{kargaşa} = W_{M \times k}\, H_{k \times N}$) ve hedef verisi ( $X_{hedef} = S_{M \times N}$) elde edilir [74].

$$W_{ij} \leftarrow \left[ \frac{\left| \left( (S - X) H_T \right)_{ij} \right| - \left( (S - X) H_T \right)_{ij}}{2 \left( W H H^T \right)_{ij}} \right] W_{ij} \tag{3.14}$$

$$H_{ij} \leftarrow \left[ \frac{\left| \left( W^T (S - X) \right)_{ij} \right| - \left( W^T (S - X) \right)_{ij}}{2 \left( W^T W H \right)_{ij}} \right] H_{ij} \tag{3.15}$$

# 4. DERİN ÖĞRENME İLE YNR HEDEF TESPİTİ

YNR verileri üzerinde hedef tespiti problemine çözüm olarak bu çalışmada derin öğrenme modellerine odaklanılmıştır. Derin öğrenme ile nesne algılama modellerinden Faster R-CNN, EfficientDet, YOLO v3 ve YOLO v5 modelleri beş farklı veri seti ile eğitilmiştir. Derin öğrenme modellerinin doğruluğuna ek olarak farklı kargaşa giderme yöntemleri uygulanmış veri setleri kullanılarak, kargaşa gidermenin modellerin performansına etkileri araştırılmıştır. Sonraki bölümlerde kullanılan veri setleri tanıtılarak modellerin eğitim aşamaları ve eğitim sonuçlarından kesitler paylaşılacaktır.

## 4.1 Veri Setleri

Bu çalışma dahilinde kullanılan YNR tarama verileri gprMax [75] ile oluşturulmuştur. Farklı boyutlardaki nesneler ile farklı materyaller kullanılarak çeşitli senaryolar elde edilmiştir. Nesnelerin pozisyonları ve derinlikleri değiştirilerek YNR verileri simüle edilmiştir. Böylece gerçek dünyada oluşabilecek farklı senaryolarda yer alan hedeflerin algılanabilmesi için eğitim verisi çeşitlendirilmiştir.

Veri oluşturma simülasyonu esnasında 300Hz anten frekansı ve 100ns zaman ölçeği sabit tutulmuştur. Gerçek veride oluşabilecek olan gürültü bileşeni de simülasyon görsellerine eklenmiştir. Bu çalışma dahilinde 57 adet simülasyon verisi elde edilmiştir. Elde edilen simülasyon verilerine SVD, PCA, RPCA ve RNMF kargaşa giderme yöntemleri ayrı ayrı uygulanarak sonraki aşamalarda derin öğrenme model eğitiminde kullanılmıştır. Çizelge 4.1 ile eğitimde kullanılacak ham veri ve SVD, PCA, RPCA ve RNMF kargaşa giderme yöntemleri ayrı ayrı uygulanarak oluşan görsel kümelerinin yüzdelik dağılımı gösterilmektedir.

**Çizelge 4.1 :** Derin öğrenme modellerinde kullanılan görsel kümelerinin yüzde dağılımı.

| Eğitim (%) | Validasyon (%) | Test (%) |
|---|---|---|
| 70 | 20 | 10 |

Her bir görsel kümesi eğitim, validasyon ve test olmak üzere Çizelge 4.1 ile gösterildiği oranda bölümlere ayrılmıştır. Eğitim için ayrılan görseller modellerin eğitiminde kullanılırken validasyon görselleri, modeller tarafından eğitimin optimize edilmesi amacıyla kullanılmıştır. Test görselleri ise eğitime dahil edilmez ve eğitim sonrası işlemlerde modelin doğruluğunun hesaplanmasında kullanılmaktadır.

Kargaşa giderme problemi YNR hedef tanımada sıkça karşılaşılan problemlerden biridir. Hedef dışı etmenlerden kaynaklanan veriler tarama görselinde yer alır. Bu durum hedefin ön plana çıkmasının önüne geçmekle beraber kargaşanın yapısına bağlı olarak hedef verisini bozabilir. Çalışmada kullanılan ilk veri seti kargaşa giderme uygulanmamış veri setidir. Diğer veri görselleri, ham haldeki bu veri setine kargaşa giderme yöntemleri uygulanması ile oluşur.

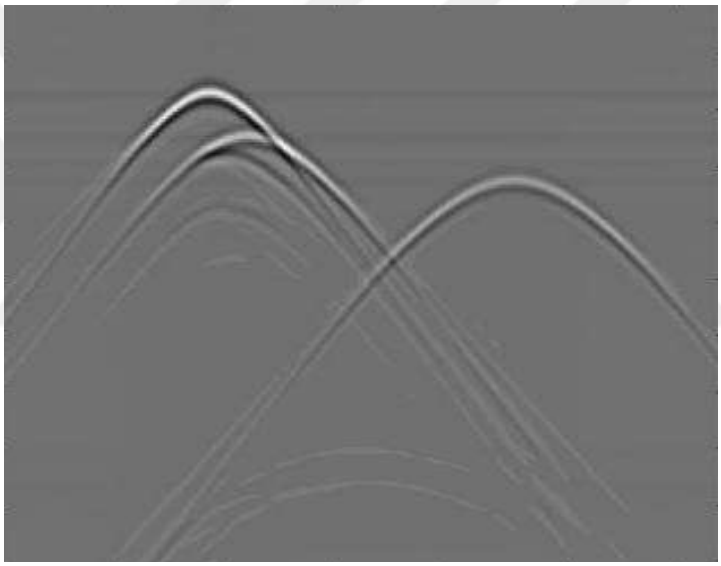
Kargaşa giderme uygulanmamış veri setindeki her görselin diğer veri setlerinde karşılığı bulunmaktadır. Bu durum kargaşa gidermenin etkilerine odaklanmaya olanak sağlar. Veri setinde yer alan hedeflerin lokasyonları değişkenlik göstermektedir. Görsellerde yer alan kargaşa bileşeninin miktarı ölçüm yapılan ortamdan ya da yüzey çeşidinden kaynaklanabilir.

Şekil 4.1'de veri setinde yer alan görsellerden biri yer almaktadır. Şeklin üst bölgesinde yüzeyden kaynaklanan kargaşa verisi göze çarpmaktadır.

Şekil 4.1 : Kargaşa giderme uygulanmamış görsel örneği.

Şekil 4.1'de yer alan hedef verilerinin kargaşadan etkilenerek sinyalin zayıfladığı göze çarpmaktadır. Gerçek hayat verilerinde Şekil 4.1'de gerçekleşen senaryo gibi veri görseli elde etme ihtimaline karşın, zayıf sinyaller içeren YNR görseller veri setine dahil edilmiştir. Böylelikle modeller zayıf sinyal üzerinden hedef tanımayı da öğrenerek benzer hedef verilerinin algılaması kolaylaşacaktır.

SVD, bu çalışma dahilinde kullanılan kargaşa giderme yöntemlerinden biridir. SVD yöntemi uygulanan ham YNR verilerinin bir araya getirilmesi ile oluşan görsel kümesi eğitimlerde kullanılmıştır. Şekil 4.2'de SVD uygulanmış örnek YNR görseli yer almaktadır.



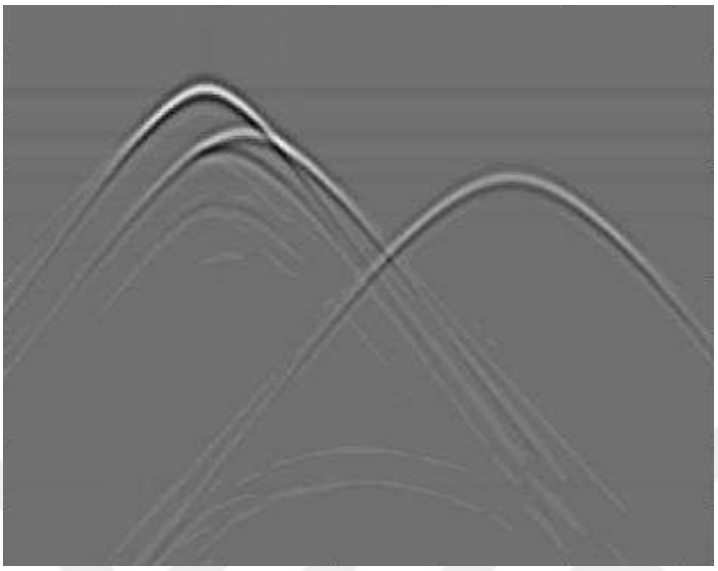
Şekil 4.2 : SVD uygulanmış örnek YNR görseli.

Şekil 4.2'de görülen görsel, Şekil 4.1'deki ham YNR verisine SVD uygulanması ile oluşmuştur. Görselin üst katmanında yer alan kargaşa verisi silinmiş ve hedef veri sinyalleri artırılarak belirgin hale getirilmiştir. Karmaşa verisinin büyük ölçüde giderildiği görülmekle beraber hedef etrafında görülen çizgi şeklindeki hedef dışı veriler göze çarpmaktadır.

SVD görsel kümesi ham veri seti ile eşit sayıda görsel içermektedir. Ham veri setinde yer alan her bir görselin SVD görsel kümesinde karşılığı bulunmaktadır.

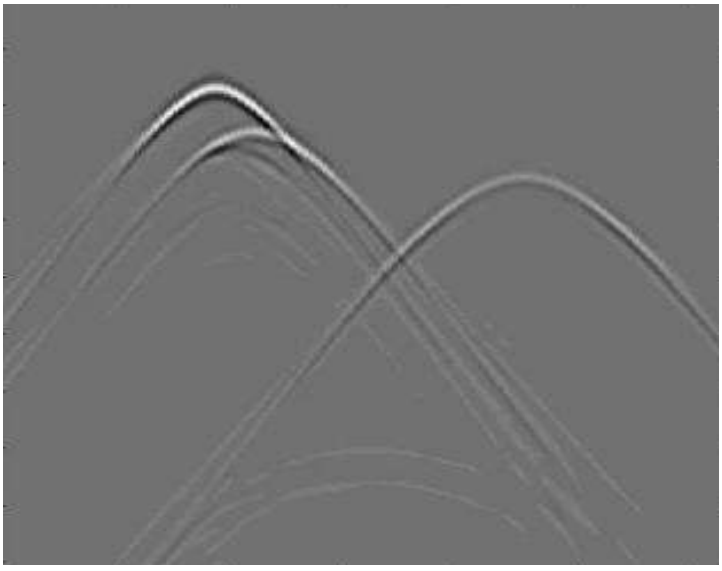
PCA bağlı değişkenlerden lineer bağımsız değişken setleri oluşturmaya yarayan bir dönüşüm metodudur. Bu yöntem ile oluşan kargaşa verisinin azaltılması

amaçlanmaktadır. Ham veri setinde yer alan görsellere PCA uygulanması sonucu oluşan görsel kümesi model eğitimlerinde kullanılmaktadır. Şekil 4.3'te yer alan görsel Şekil 4.1'deki YNR verisine PCA uygulanması sonucu ortaya çıkmıştır.



Şekil 4.3 : PCA uygulanmış örnek YNR görseli.

Şekil 4.3 ile Şekil 4.1 karşılaştırıldığında görselin üst kısmında yer alan karmaşa verisinin silindiği görülmektedir. Hedefin etrafında yer alan ve hedef dışı veri içeren çizgiler içermesi nedeniyle Şekil 4.2 ile benzerlik göstermektedir. Bu görsel özelinde PCA ve SVD yöntemlerinin benzer çıktılar verdiğini dile getirmek mümkündür.

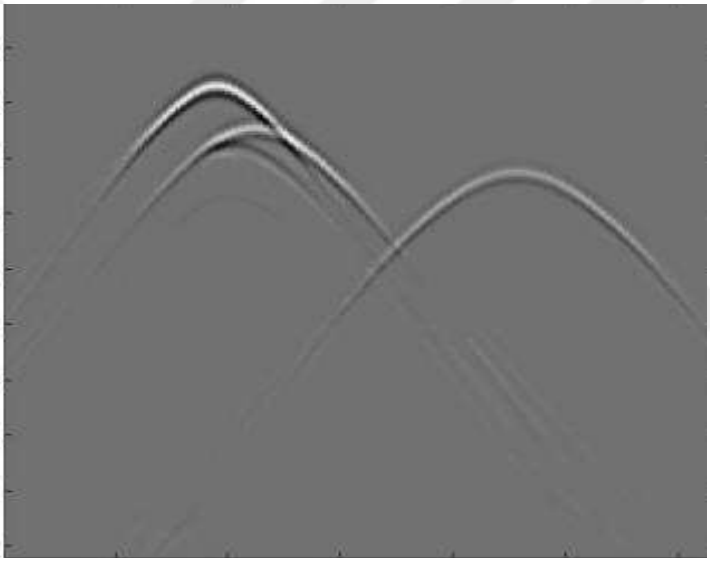


Şekil 4.4 : RPCA uygulanmış örnek YNR görseli.

Çalışmada kullanılan bir diğer kargaşa giderme yöntemi olan RPCA, ham veri setinde bulunan bulunan tüm görsellere uygulanmıştır. RPCA, kargaşa gidermede etkili yöntemlerden biridir. Şekil 4.4'te RPCA uygulanmış görsel görülmektedir.

Şekil 4.1'e RPCA uygulanması sonucu oluşan Şekil 4.4'te kargaşa verisinin giderildiği göze çarpmaktadır. Ayrıca Şekil 4.2 ve Şekil 4.3'te yer alan çizgi halindeki hedef dışı veriler de silinmiştir. Bu yönü ile RPCA yönteminin hedef dışı veri temizleme konusunda daha yüksek performanslı olduğu göze çarpmaktadır.

RNMF yöntemi, bu çalışmada kullanılan son kargaşa giderme yöntemidir. Oluşan kargaşa verilerinin bütün veriden ayrıştırılması yoluna gidilir. Belirli sayıdaki iterasyonlar ile kargaşa verisi saptanarak toplam veriden ayrıştırılır. Şekil 4.5'te örnek RNMF uygulanmış görsel yer almaktadır.



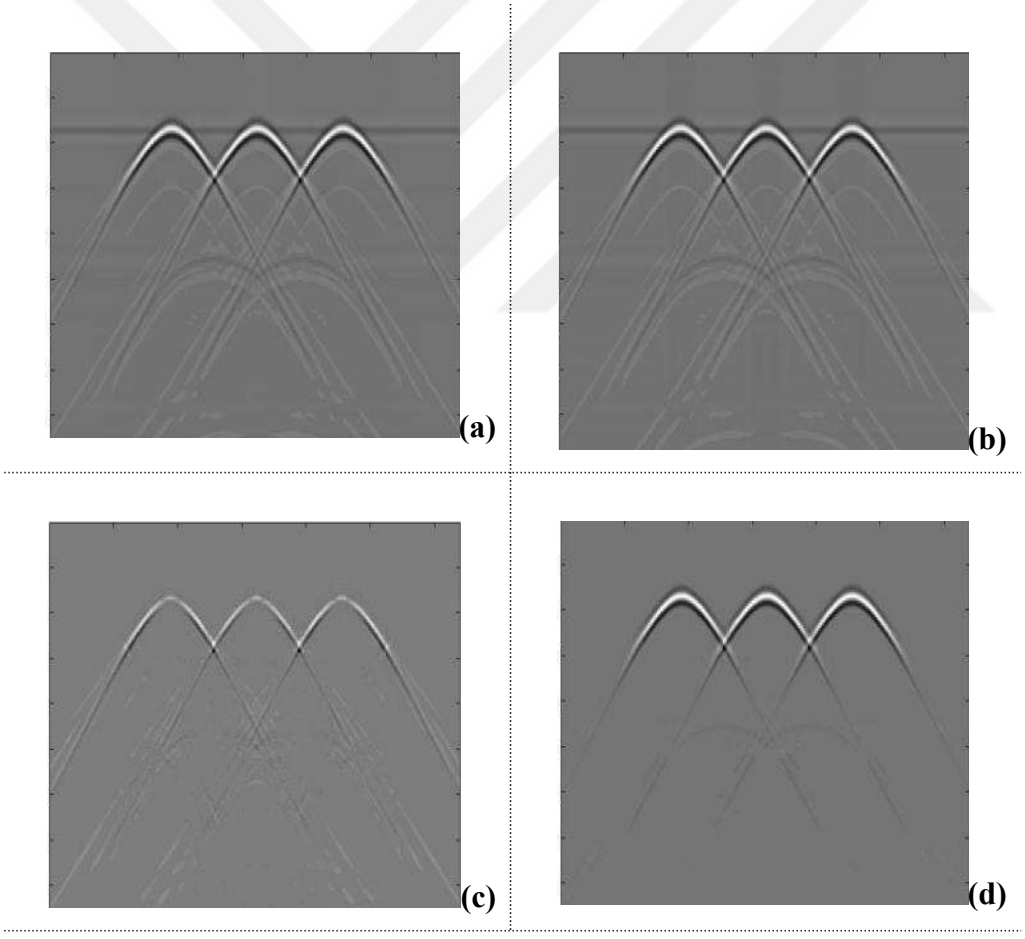
Şekil 4.5 : RNMF uygulanmış örnek YNR görseli.

Şekil 4.1'e RNMF uygulanması sonucu oluşan Şekil 4.5'te görüldüğü üzere kargaşa veri bileşeni elimine edilerek hedef veri bileşeni ön plana çıkarılmıştır. Çıktı olarak RPCA uygulanmış olan Şekil 4.4'e benzer bir yapıda olmasının yanında, bu görsel özelinde RNMF uygulanmış görselin daha az hedef dışı veri içerdiği görülmektedir.

Bu çalışma dahilinde yukarıda belirtilen görsel kümelerinin her biri eşit sayıda (57) adet YNR hedef verisi içermektedir. Bir adet ham YNR veri setine ek olarak SVD, PCA, RPCA ve RNMF uygulanmış görsel kümeleri de dahil olmak üzere toplamda beş adet görsel grubu ayrı ayrı derin öğrenme modellerinin eğitimi için bir araya getirilmiştir. Çalışma dahilinde uygulanan kargaşa giderme yöntemlerinin etkinliğini

görmek açısından Şekil 4.6'da yer alan ham YNR görseline dört farklı kargaşa giderme yöntemi uygulanması sonrası çıktılar Şekil 4.7'de gösterilmektedir.

Şekil 4.6 : Kargaşa giderme uygulanmamış örnek YNR görseli.




Şekil 4.7 : Kargaşa giderme yöntem karşılaştırması: (a)SVD. (b)PCA. (c)RPCA. (d)RNMF.

Şekil 4.6'daki ham YNR verisi ve Şekil 4.7'de yer alan kargaşa giderme yöntem çıktıları göz önüne alındığında RNMF yönteminin en az hedef dışı veri içerdiği

gözlenmektedir. SVD ve PCA uygulanmış hedef verisi yakınlarında çizgi halinde hedef dışı veri yer almaktadır. RPCA yöntemi uygulanmış görsel ise hedef dışı verilerden temizlenmiş bir görüntüye sahipken, RNMF çıktısı RPCA'e oranla daha net bir hedef çıktısı sağlamaktadır.

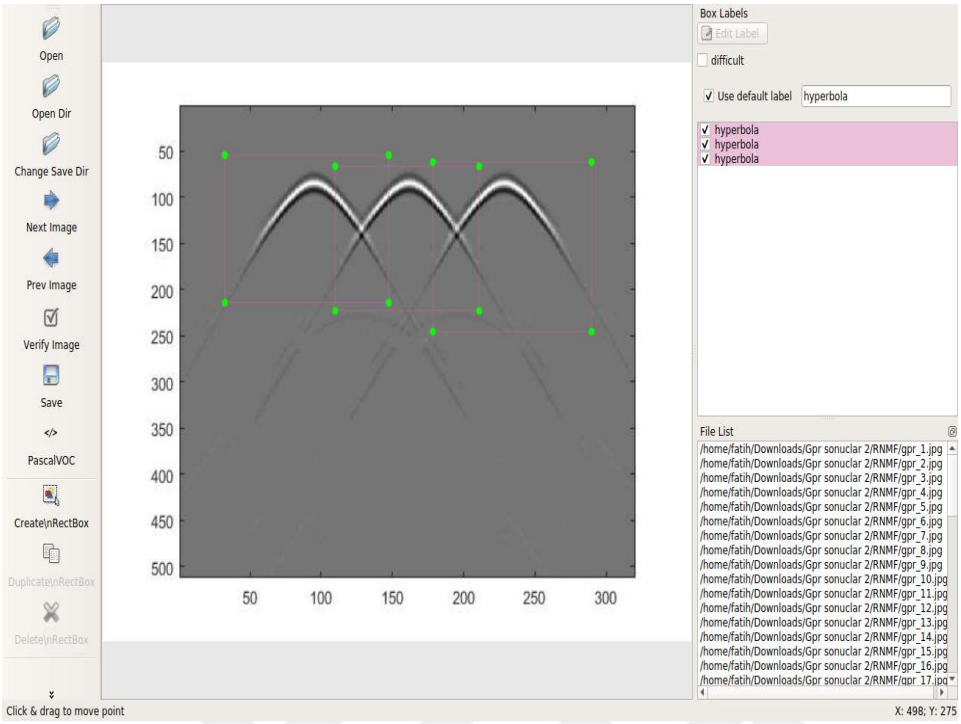
Bir adet ham veri seti ve aynı ham veri setine dört adet kargaşa giderme yöntemi uygulanması ile elde edilen görsel kümesi olmak üzere toplamda beş adet veri seti elde edilmiştir. Elde edilen her bir veri seti eğitim öncesi işlemlerden geçerek derin öğrenme model eğitimlerine hazır hale getirilir.

Bu çalışma kapsamında Faster R-CNN, YOLO v3, EfficientDet ve YOLO v5 modelleri YNR hedef tanıma problemine çözüm olarak oluşturulan beş farklı veri seti ile eğitilmek üzere eğitim öncesi işlemler adımına geçirilmiştir. Kargaşa giderme yöntemlerinin derin öğrenme modellerinin performanslarına etkilerini gözlemlemek amacıyla, veri setleri aynı sayıda ve aynı çözünürlükte verilerden oluşmaktadır. Bu durum karmaşa giderme yöntemi dışında etki edebilecek yöntemlerin önüne geçmeyi amaçlamaktadır. Eğitim öncesi işlemlere tabi tutulan veri setleri sonrasında eğitim aşamasında kullanılmaktadır.

### 4.2 Eğitim Öncesi İşlemler

Model eğitimlerinin hemen öncesinde eğitime katılacak olan veri setleri bir takım işlemlerden geçirilir. İşlemler sonrasında eğitime hazır hale gelen veri setleri bölümlere ayrılarak eğitime dahil edilir. Eğitim öncesi süreçler eğitimin sağlıklı olması açısından oldukça önemlidir. Bu nedenle model eğitiminin öncesinde eldeki veriler işlenerek ham veri hazır veri haline getirilir. Çalışmalarda kullandığımız SVD , PCA, RPCA ve RNMF veri setlerine bire bir aynı işlemler uygulanmıştır.

Veri setlerinin eğitimde yer alabilmesi için öncelikle her bir görsel tek tek taranarak üzerinde yer alan hiperbol şeklindeki YNR hedefleri işaretlenmelidir. Bu işlemi gerçekleştirmek için **labelImg** adı verilen bir program kullanılmıştır. **LabelImg** ile görsel üzerindeki hedeflerin işaretlenebilmesi için işaretlenecek olan hedeflere bir etiket atanmalıdır. Bu çalışmada her bir hedef hiperbolü "hyperbola" etiketi ile işaretlenmiştir.




Şekil 4.8 : LabelImg ile hedef hiperbolleri işaretlenen görsel örneği.

Her bir görsel üzerindeki hiperbol şeklinde yer alan hedefler tek tek işaretlenmiştir. İşaretleme sırasında dikdörtgen şeklinde bir çerçeve kullanılarak, hedef hiperbolü ortalanacak şekilde çizilmiştir. Dikdörtgen ile belirlenen hedef pozisyonları **txt** formatında, görsel ile aynı isimde bir dosyada saklanmaktadır. Böylelikle, ilgili görselin üzerinde yer alan hedeflerin lokasyonları kaydedilmiş olur. Örnek olarak Şekil 4.8'de işaretlenen YNR verisi için oluşan hedef bilgisi Şekil 4.9'da gösterilmiştir.

Şekil 4.8'de **labelImg** ile işaretlenen hiperbollerin oluşturduğu Şekil 4.9'da yer alan ilk parametre sınıf bilgisini içerir. Sınıf değerleri listesinin numaralandırılmasına sıfırdan başlanır ve çalışmada tek bir sınıf (hiperbol) yer almasından dolayı hedeflerin sınıfı sıfır (0) ile ifade edilir. Sınıflama bilgisinin dışında yer alan değerler hedefi çevreleyen dikdörtgen çerçevenin, dört köşesinin koordinatlarını içerir.

```
0 0.353571 0.240476 0.164286 0.114286
0 0.522321 0.236905 0.151786 0.102381
0 0.688393 0.240476 0.155357 0.109524
```

Şekil 4.9 : Hiperbollerin  işaretlenmesi ile oluşan dosya içeriği.

Veri setlerinde yer alan tüm görsellerdeki hedef hiperbolleri tek tek işaretlenir ve koordinat bilgileri örnekte olduğu gibi kaydedilir. Sonrasında veri setlerinin eğitim, validasyon ve test aşamalarında kullanılacak olan görseller parçalara ayrılır. Bu çalışmada her bir veri seti %70 eğitim, %20 validasyon ve %10 test olmak üzere üç parçaya ayrılmıştır. Veri setlerinin üçe bölünmesi sonucu oluşan görsel toplulukları, hangi amaçla kullanılacağına bağlı olarak farklı klasörlerde saklanır. Bu durum eğitim, validasyon ve test süreçlerinde kullanılacak görsellere ulaşılmasını kolaylaştırır.

Eğitim için ayrılan görsel grubu, hem Faster RCNN, YOLO v3, EfficientDet ve YOLO v5 modellerinin eğitim aşamasında kullanılır. Validasyon grubu ise eğitimin model tarafından optimum hale getirilmesinde kullanılır. Test grubu eğitim tamamlandıktan sonra hedef tespit doğruluğunun testinin yapılmasında kullanılacak olan görsellerden oluşur. Sonuç bölümünde eğitim verileri ile beraber test grubunda yer alan görsellerin hedef algılama sonuçlarına yer verilecektir.

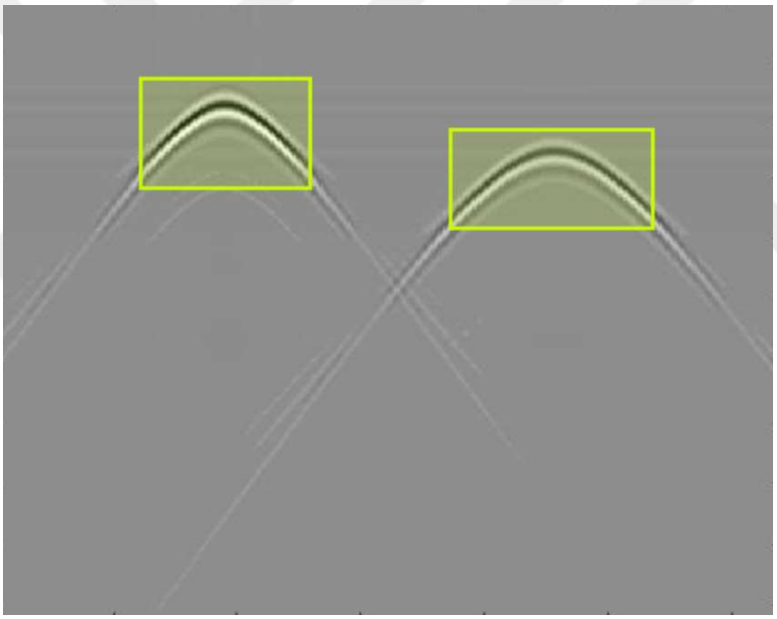
Veri setlerinde bulunan her bir görsel eğitim öncesinde yeniden boyutlandırılarak ortak bir çözünürlüğe sahip olması sağlanır. Çalışmamız dahilinde eğitim, validasyon ve test aşamalarında kullanılan her bir görsel 416x416 çözünürlüğüne getirilir. Eğitim öncesi işlemler sonucu hazırlanan görsellerden oluşan veri setleri eğitim aşamasına aktarılır.

### 4.3 Model Eğitimleri

Çalışma dahilinde kullanılan Faster R-CNN, YOLO v3, EfficientDet ve YOLO v5 modelleri, SVD, PCA, RPCA ve RNMF uygulanmış veri setleri ve kargaşa giderme uygulanmamış veri seti ile eğitime tabi tutulmuştur. Bu nedenle toplamda her bir model beş farklı veri seti ile eğitilmek üzere toplamda yirmi farklı eğitim gerçekleştirilmiştir. Her bir eğitimde, önceki aşamada eğitime hazır hale getirilen veri setlerinde değişim yapılmadan kullanılmıştır. Faster R-CNN, EfficientDet, YOLO v3 ve YOLO v5 modelleri aynı görseller ile ayrı ayrı eğitilerek YNR hedef algılayıcılar oluşturulmuştur. Eğitimlerin tamamı aynı özellikteki makinelerde GPU kullanılarak gerçekleştirilmiştir. Aynı özellikteki makinelerde gerçekleştirilen eğitim ile beraber modellerin toplam eğitim süreleri kaydedilerek sonraki bölümlerde paylaşılmıştır. Eğitim ve test ortamı olarak Google Colab kullanılmıştır.

### 4.3.1 **Faster R-CNN model eğitimi**

Bu çalışma dahilinde YNR hedef tanıma problemine çözüm olarak test edilen Faster R-CNN, diğer yöntemlerden daha eski olmasına rağmen hedef tanıma için en çok tercih edilen yöntemlerden biridir. Tüm veri setleriyle ayrı ayrı olmak üzere beş ayrı Faster R-CNN modeli eğitilmiştir. Her eğitim aynı konfigürasyonlar ve aynı özellikteki makinelerde gerçekleştirilmiştir. Eğitimler **python** programlama dilinde yazılmış kodlar üzerinden yapılırken **pytorch** kütüphanesi kullanılmıştır. Her bir veri setinin eğitim, validasyon ve test için ayrılan parçaları üzerinden eğitim tamamlanırken, testlere diğer gruplarda yer alan görseller dahil edilmemiştir. Eğitim öncesi işlemlerde hedefleri işaretlenen görseller eğitime dahil edilmiştir (Şekil 4.10). Mevcut işaretli hedefler yardımıyla Faster R-CNN modelinin hedef özniteliklerini çıkararak anlamlı veri elde etmesi sağlanmıştır.



Şekil 4.10 : Hedefleri işaretlenmiş YNR görüntü örneği

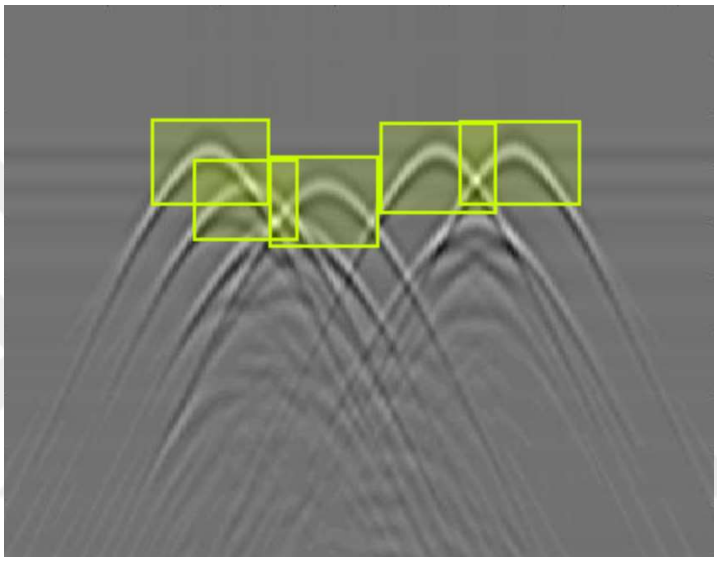
Eğitimlerde maksimum iterasyon sayısı 1500 olarak ayarlanmıştır. Eğitim sonucu oluşan model verileri kaydedilerek, test verilerinin değerlendirilmesinde kullanılmıştır.

### 4.3.2 EfficientDet model eğitimi

EfficientDet eğitimleri sırasında, Faster R-CNN model eğitimlerinde olduğu gibi **python** dili ve **pytorch** kütüphanesi kullanılmıştır. Eğitimlerde yine tüm veri setleri kullanılırken tek değişkenin derin öğrenme modeli olmasına özen gösterilmiştir.

Eğitim öncesi işlemlerden çıkan veriler model eğitimine sunulmuştur. Eğitimlerde özellikle farklı pozisyonlarda YNR hedefi içeren görsellere ağırlık verilmesinden dolayı, çeşitli şekillerde işaretlenmiş hedef verisi kullanılmıştır. Örnek olarak üst üste ve karmaşık bir yapıda olan işaretlenmiş YNR verisi Şekil 4.11'te görülebilir.

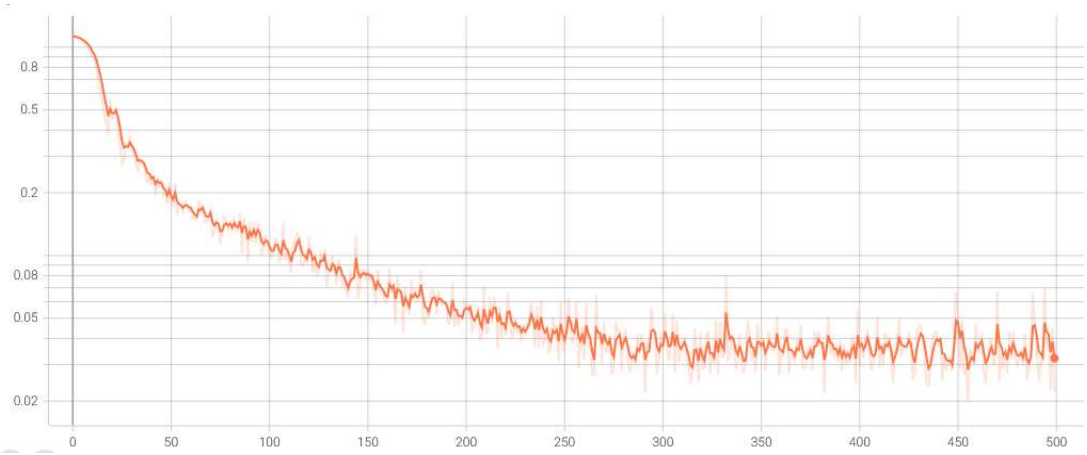
İlk izlenimler sonucunda EfficientDet model eğitimlerinin Faster R-CNN model eğitimlerine oranla daha kısa sürdüğü gözlenmiştir. Bu durumun hedef algılama ve doğruluk oranlarına yansıması sonraki bölümde tartışılacaktır.



Şekil 4.11 : Karmaşık hedefler içeren işaretlenmiş YNR görsel örneği

EfficientDet modeli Faster R-CNN'den farklı yapısı sebebiyle eğitim süreçleri de farklılık göstermektedir. Örnek olarak Faster R-CNN 1500 iterasyon sonucunda optimum kayıp değerine erişirken EfficientDet modelinde iterasyon sayısı daha az görülmüştür. Model eğitimleri süresince kayıp değerlerinin değişimi izlenir. Kayıp değeri, hedefin asıl pozisyonu ile model tarafından tahminlenen pozisyonu arasındaki farkı ifade eder. Kaybın belirli bir iterasyon sonunda sabit kalması ya da belirli değer arasında gidip gelmesi durumunda eğitim sonlandırılabilir. EfficientDet model eğitimlerinde 500 iterasyon sonucunda kayıp değerlerine bakılarak eğitim sonuçlandırılmıştır. İterasyon sayısı Faster R-CNN'e göre daha düşük olsa da kaybın optimize edilmesi sağlanmıştır. Şekil 4.12'de kayıp değerlerinin iterasyona bağlı değişimi gösterilir. Zamanla kayıp değerinin minimize edilerek belirli değerler

arasında kaldığı görülmektedir. Her iki model için de ortalama doğruluk ve kayıp değerleri karşılaştırması sonraki bölümde sunulmuştur.




Şekil 4.12 : EfficientDet model eğitiminde kayıp değerinin iterasyon sayısına göre değişimi

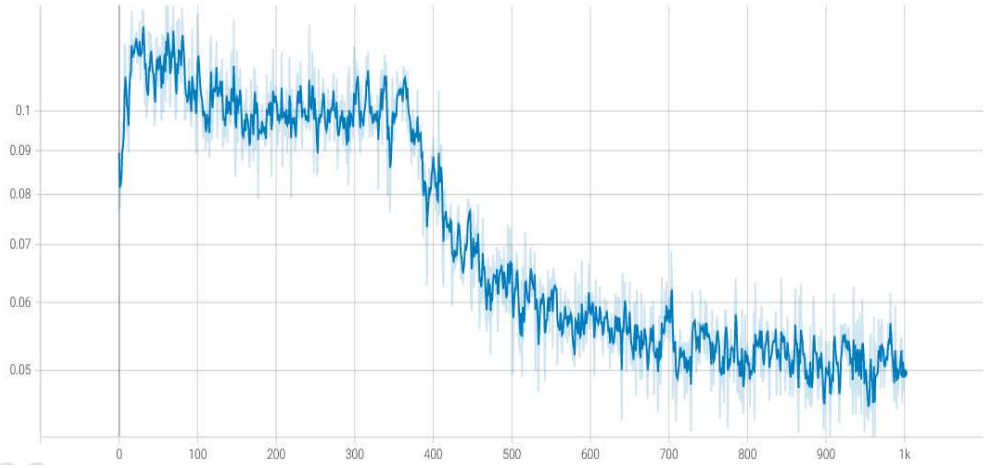
### 4.3.3 YOLO v3 model eğitimi

YOLO v3 eğitimlerine tüm veri setleri dahil edilmiştir. Veri setlerinin her biri **python** progamlama dili ve **pytorch** kullanılarak eğitilmiştir. Tekrarlı eğitim sayısını ifade eden epoch değeri 300 olarak ayarlanarak eğitildiğinde, optimum değerlere erişildiği görülmüştür.

### 4.3.4 YOLO v5 model eğitimi

Bu çalışmada yer verilen son derin öğrenme modeli olan YOLO v5 tüm veri setleri ile eğitilerek beş farklı YNR hedef yakalayıcı elde edilmiştir. YOLO v5 python programlama dili ve pytorch kullanılarak eğitilmiş olup diğer modeller ile aynı süreçlere dahil olmuştur. Epoch değeri 1000 olarak ayarlanarak eğitim bu sayıdaki tekrarlı hesaplamalar sonucunda tamamlanmıştır.

Şekil 4.13'te ham veri ile eğitlmiş YOLO v5 modelinin kayıp değerinin iterasyona göre değişimi gösterilmektedir. YOLO v5 modellerinde genel olarak düşük kayıp değeri göze çarpmaktadır. Kargaşa giderme yöntemlerinden bağımsız olarak modelin bir özelliği olduğu görülmektedir.

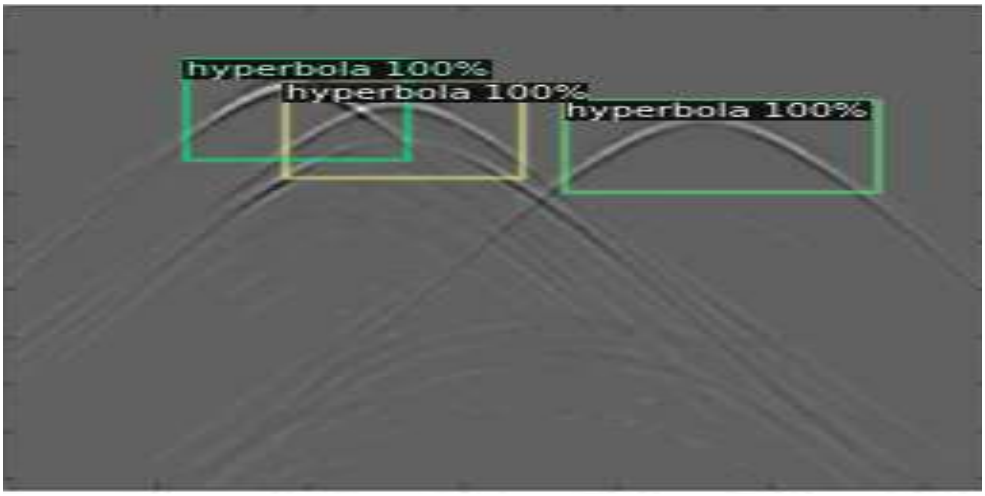



Şekil 4.13 : Ham veri ile eğitilen YOLO v5 kayıp grafiği

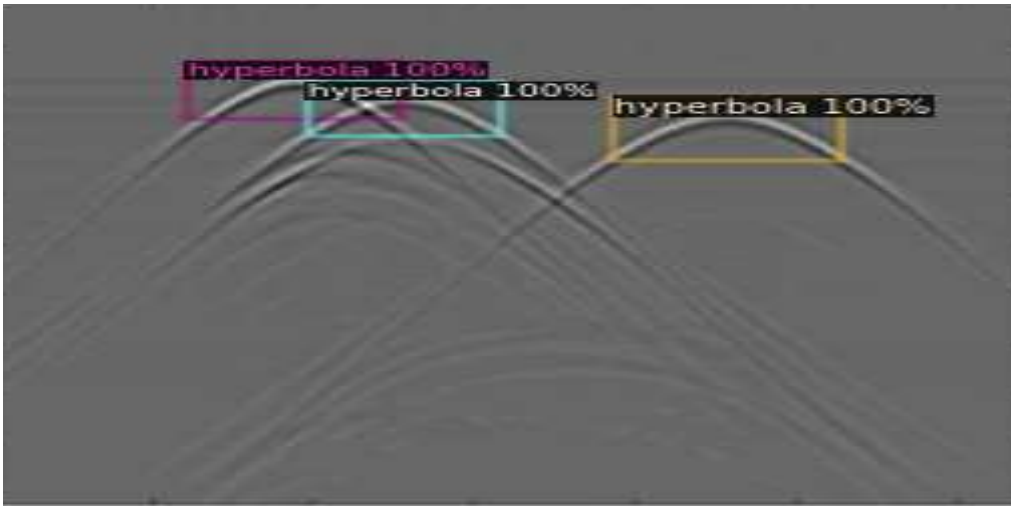
### 4.4 Eğitim Sonuçları

Faster R-CNN, EfficientDet, YOLO v3 ve YOLO v5 kullanılarak beş farklı veri seti ile eğitilen modellerin eğitiminden sonra ilgili değerler karşılaştırılmıştır. Modellerin eğitim sonuçları yine aynı boyuttaki görseller ile oluşturulmuştur.

Test sonuçlarında hedef hiperbolleri etrafında oluşturulan dikdörtgen çerçeve COCO [76] değerlendirmelerinde kullanılmaktadır. Çerçevenin hedef etrafındaki lokasyonu modelin doğruluğunu etkilemektedir. Örnek olarak aynı model ile eğitilen ve aynı hedefe ait olan Şekil 4.14 ve Şekil 4.15'deki görseller etrafındaki çerçevenin köşe noktaları farklılık göstermektedir. Hedef lokalizyon farklılıklarına sebep olan bu detaylar COCO değerlendirme sonuçlarına etki etmektedir.



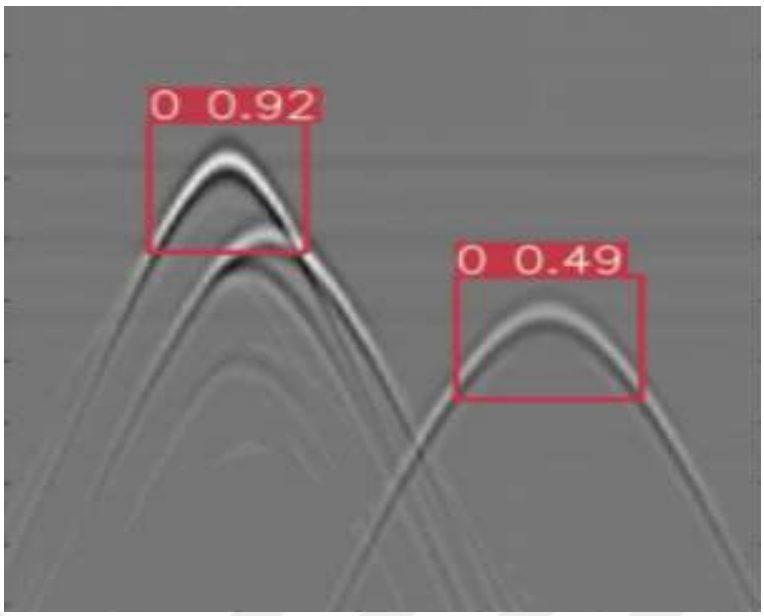

Şekil 4.14 : Faster R-CNN modeli ve RNMF veri seti ile eğitilmiş test sonuç örneği.




Şekil 4.15 : Faster R-CNN modeli ve PCA veri seti ile eğitilmiş test sonuç örneği.

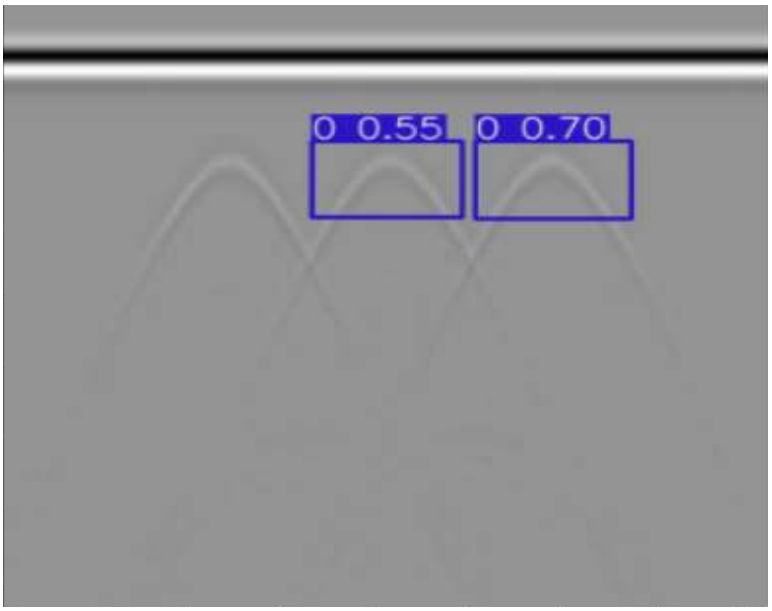
Modeller göz önüne alındığında kargaşa giderme uygulanmamış ham veri ile eğitilen YOLO v3 modellerinin karmaşık hedeflerde başarısız olduğu göze çarpmaktadır. Örnek olarak Şekil 4.16'da kargaşa giderme uygulanmamış verideki hedefler algılanamamıştır. Ancak SVD ile eğitilen model, daha belirgin olan hedefin bir kısmını algılamayı başarmıştır (Şekil 4.17).

Şekil 4.16 : Ham veri ile  eğitilmiş YOLO v3 model çıktısı.

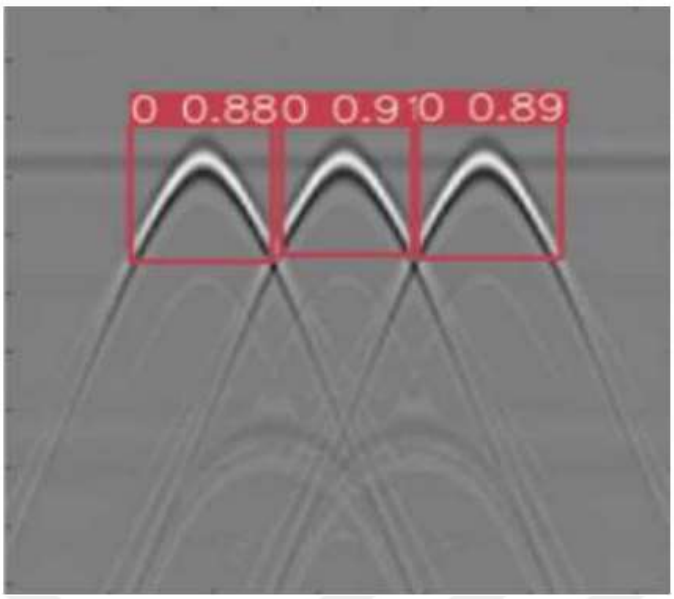



Şekil 4.17 : SVD veriseti ile hedef algılamış YOLO v3 model çıktısı.

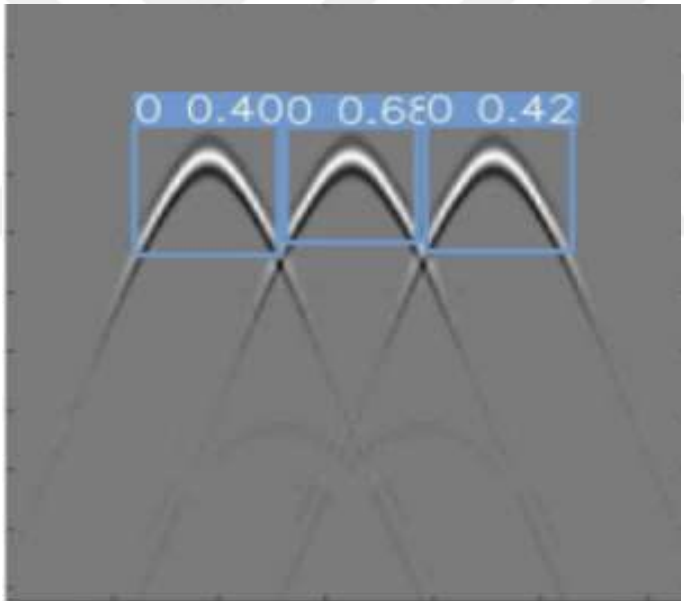
Şekil 4.18'de yer alan görsel kargaşa giderme uygulanmamış veri seti ile eğitilmiş modeldir. Görüldüğü üzere hedeflerden birini algılayamamıştır. Ancak Şekil 4.19 ve Şekil 4.20'de yer alan SVD ve RNMF veri setleri ile eğitilmiş modeller hedeflerin tamamını bulmuştur. Bu durumun sebebi olarak SVD ve RNMF hedeflerinin sınır çizgilerinin belirginliği gösterilebilir.




Şekil 4.18 : Ham veri seti ile eğitilen YOLO v3 çıktı örneği.



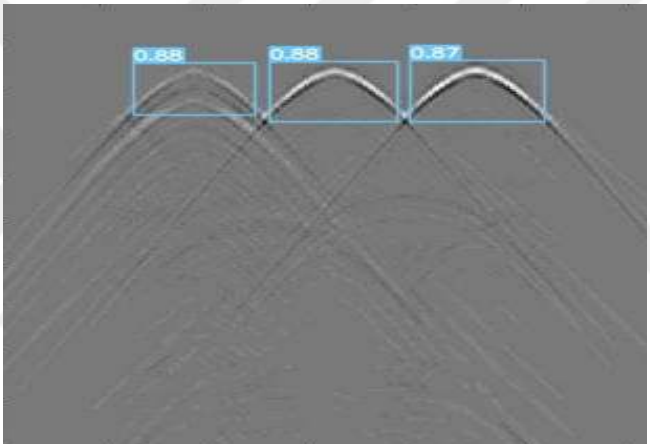

Şekil 4.19 : SVD ile eğitilen YOLO v3 çıktı örneği



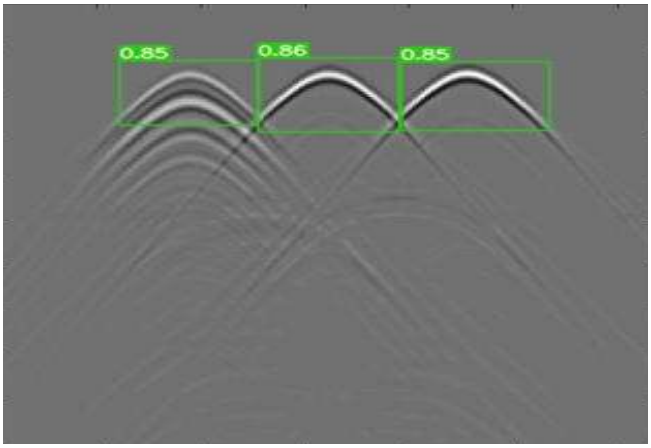

Şekil 4.20 : SVD ile eğitilen YOLO v3 çıktı örneği

YOLO v5 modeli ile eğitilen hedef algılayıcıların tamamı yüksek performans göstermektedir. Ham veriler ile eğitilen model hedef algılamada başarılı olmakla beraber aradaki performans farkını belirleyen hedeflerin güven skoru olmaktadır. Şekil 4.21'de ham veri ile eğitilen modelin tüm hedefleri bulduğu gözlenmektedir. Ancak güven skoru RPCA (Şekil 4.22) ve RNMF(Şekil 4.23) veri setlerine oranla daha düşüktür.


Şekil 4.21 : Ham veri ile eğitilen YOLO v5 çıktı örneği



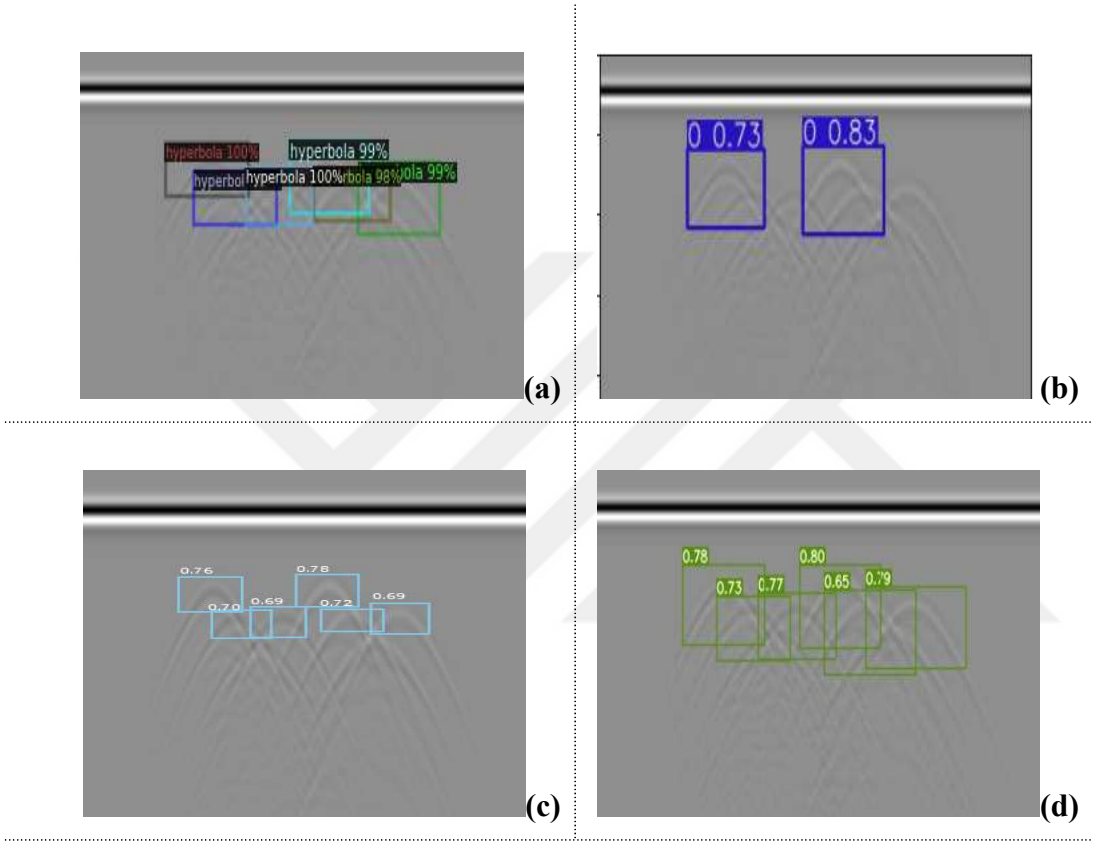

Şekil 4.22 : RPCA verisi ile eğitilen YOLO v5 çıktı örneği




Şekil 4.23 : RNMF verisi ile eğitilen YOLO v5 çıktı örneği

Genel bağlamda kargaşa giderme uygulanmamış veri seti ile eğitilen modellerin performansı diğerlerine göre daha düşük kaldığı görülmektedir. Kargaşa giderme uygulanan ile kargaşa giderme uygulanmamış veri setleri karşılaştırıldığında kargaşa giderme uygulananların sonuçları olumlu olduğu görülmektedir.

Modellerin aynı veri setlerindeki performansı karşılaştırılırken ortak verilerle oluşturulan çıktılar göz önüne alınmalıdır.

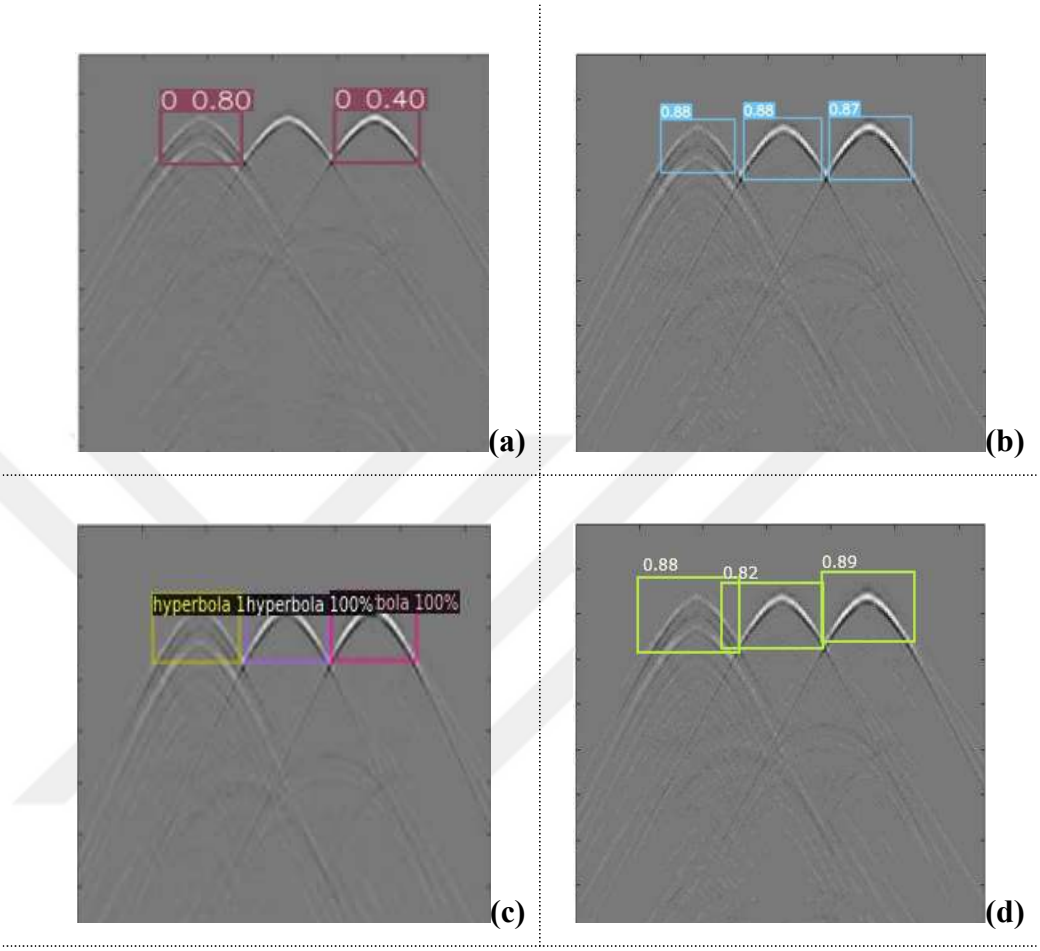



Şekil 4.24 : Ham veri ile eğitilmiş model karşılaştırması: (a) Faster R-CNN. (b)YOLO v3. (c)EfficientDet. (d)YOLO v5.

Şekil 4.24'te ham veri ile eğitilen modellerin aynı test datasında YNR hedef tahminleme sonuçları verilmiştir. YOLO v3 modelinin yalnızca iki hedef yakaladığı gözlenmektedir. Faster R-CNN ve EfficientDet modelleri başarılı bir şekilde hedef yakalasa da, tüm hedefleri doğru tahminleyen ve hedef lokalizasyonunu daha doğru gerçekleştiren YOLO v5 olarak göze çarpmaktadır.

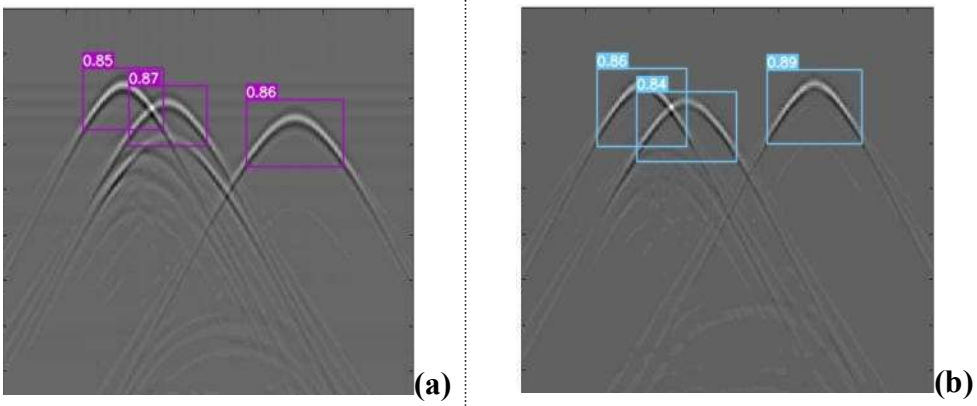
Bir sonraki karşılaştırmada (Şekil 4.25), YOLO v3 modelinin bir kez daha hedef yakalama konusunda zayıf kaldığı görülmektedir. Diğer modeller hedeflerin tamamını yakalarken hedef etrafındaki çerçevenin lokasyonu ve güven skoru modele göre değişkenlik göstermektedir. Test veri setlerinin tamamında YOLO v3 dışındaki

modeller benzer hedef algılama performansları sergilemektedir. Bu durumda modellerin karşılaştırmaları görsel çıktılar yerine sayısal ölçümlerle belirlenebilmektedir.




Şekil 4.25 : RPCA verisi ile eğitilmiş model karşılaştırması:
(a)YOLO v3. (b)YOLO v5. (c)Faster R-CNN. (d)EfficientDet.

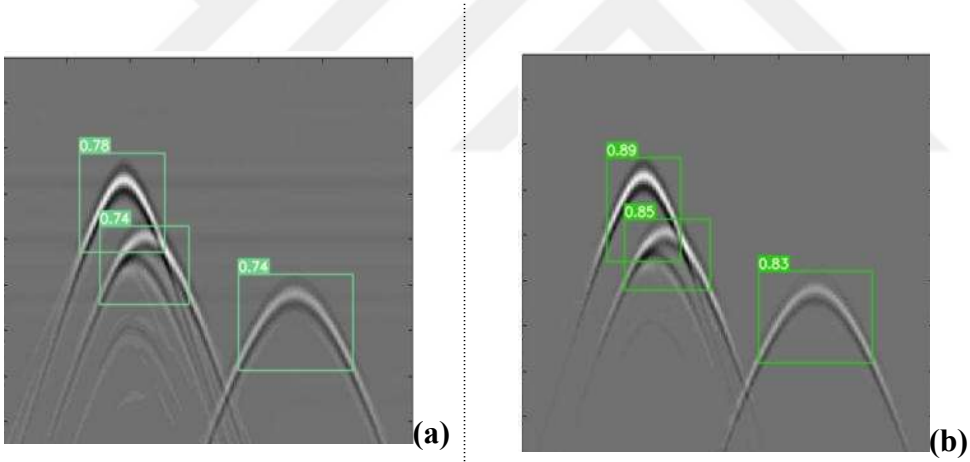
Aynı model üzerinde kargaşa giderme verilerinin karşılaştırılması yapılırken benzer test verileri göz önüne alınmalıdır. Veri setlerinin test gruplarında yer alan benzer görseller üzerinde kargaşa giderme yöntemlerinin etkileri karşılaştırılmalıdır. Şekil 4.26'da YOLO v5 modelinin PCA ve RPCA görselleri ile eğitilen versiyonlarının çıktıları karşılaştırılmıştır. PCA ile RPCA çıktıları arasında keskin bir fark olmamakla beraber iki hedef algılayıcı da başarı göstermiştir.




(a) (b)

Şekil 4.26 : YOLO v5 modelinde kullanılan kargaşa giderme yöntemleri karşılaştırması: (a)PCA. (b)RPCA.

Şekil 4.27'de bu kez SVD ile RNMF veri seti ile eğitilen YOLO v5 hedef algılayıcıları karşılaştırılmıştır. Her iki algılayıcı da başarılı olmakla beraber RNMF ile eğitilen modelin güven skoru küçük bir fark ile daha üstündür.




(a) (b)

Şekil 4.27 : YOLO v5 modelinde kullanılan kargaşa giderme yöntemleri karşılaştırması: (a)SVD. (b)RNMF.

Kargaşa giderme yöntemleri arasında belirgin farklara rastlanmamıştır.

Tüm YNR hedef algılayıcı modeller sayısal veriler yardımı ile karşılaştırılmak üzere Çizelge 4.2 oluşturulmuştur. Çizelge 4.2 ile eğitim sonuçları karşılaştırmalı olarak verilmiştir. Çizelgede yer alan sütunlar COCO [76] model değerlendirme kriterlerini içermektedir. Hedef tanıma modelinin ortalama doğruluğu mAP ile ifade edilir. Doğru tahminlenen hedeflerin toplam tahmin sayısına bulunması ile hesaplanır. COCO AP ile model değerlendirmesi yaparken hedefin lokalizasyon tahminlemesini

dikkate alır. Hedef nesne etrafında çizilen tahmini çerçeve ile asıl çerçeve arasındaki benzerlikleri hesaplar. KBO (kesişim birleşim oranı) değeri model tarafından tahmin edilen hedef lokasyonu ile asıl hedef lokasyonuna göre hesaplanır. Modelin tahminleme sırasında hedefin etrafında çizdiği çerçeve ile hedefin gerçekte yer aldığı çerçevenin kesiştiği alanın, iki çerçevenin birleşimi ile oluşan alana bölümü ile hesaplanır (Şekil 4.27). Bu bağlamda eşik değerleri tanımlayarak model performansını bu kriterlere göre değerlendirir. $AP_{50}$: hedef tahmin çerçevesi ile asıl çerçeve arasında %50 ve daha fazla kesişim olma olasılığını belirtir (KBO > 0.5). $AP_m$ orta boyutlu ($32^2$ ve $96^2$ piksel arası yer kaplayan) nesnelerin doğru tahminlenme doğruluğunu gösterir. Veri setlerinde küçük boyutlu ($32^2$ pikselden az) ve büyük boyutlu ($96^2$ pikselden fazla) hedefler yer almadığından ilgili $AP_s$ ve $AP_l$ kriterlerine yer verilmemiştir.


Şekil 4.28 : KBO gösterimi

Çizelge 4.2 ile eğitilen tüm modellerin karşılaştırmalı sonuçları paylaşılmıştır. “-” ile ifade edilenler ilgili değerin hesaplanamadığı anlamına gelir. Kalın ile işaretli olanlar en iyi performans gösterenlerdir.

Çizelgede yer alan eğitim süresi bilgisayar tarafından belirtilen süredir. Eğitimler Google Colab ile yapılmaktadır. Google Colab makinesinin özellikleri şunlardır:

- GPU: NVIDIA Tesla K80, 12 GB GDDR5 VRAM
- CPU: Tek çekrdekli Intel Xeon, 2.3Ghz
- RAM: 12.6 GB

Tüm testlerin aynı ortamda yapıldığı göz önüne alındığında bilgisayarın hesaplama kapasitesi sabit olarak kabul edilebilir. Bu nedenle eğitim süresinin düşüklüğü, olumlu bir etmendir.

**Çizelge 4.2 :** Eğitim Sonuçları

| Model | Veri Seti | Eğitim Süresi(dk) | mAP | $AP_{50}$ | $AP_m$ |
|---|---|---|---|---|---|
| YOLO v3 | Ham Veri | 72 | - | 0.895 | - |
| YOLO v3 | PCA | 69 | - | 0.976 | - |
| YOLO v3 | RNMF | 71 | - | 0.995 | - |
| YOLO v3 | SVD | 78 | - | 0.945 | - |
| YOLO v3 | RPCA | 72 | - | 0.965 | - |
| Faster R-CNN | Ham Veri | 102 | 0.513 | 0.902 | 0.513 |
| Faster R-CNN | PCA | 48 | 0.555 | 0.918 | 0.570 |
| Faster R-CNN | RNMF | 50 | 0.452 | 0.771 | 0.461 |
| Faster R-CNN | SVD | 51 | 0.364 | 0.618 | 0.364 |
| Faster R-CNN | RPCA | 56 | 0.412 | 0.604 | 0.412 |
| YOLO v5 | Ham Veri | 26 | 0.560 | 0.994 | 0.560 |
| YOLO v5 | PCA | 26 | 0.551 | 0.966 | 0.551 |
| YOLO v5 | RNMF | 26 | 0.657 | **0.996** | 0.650 |
| YOLO v5 | SVD | 26 | 0.623 | 0.996 | 0.623 |
| YOLO v5 | RPCA | 26 | 0.532 | 0.965 | 0.532 |
| EfficientDet | Ham Veri | 22 | 0.589 | 0.880 | 0.582 |
| EfficientDet | PCA | 20 | 0.663 | 0.976 | 0.663 |
| EfficientDet | RNMF | **18** | **0.669** | 0.988 | **0.669** |
| EfficientDet | SVD | 21 | 0.663 | 0.971 | 0.663 |
| EfficientDet | RPCA | 19 | 0.592 | 0.880 | 0.592 |

Eğitim sonuçlarını modele göre ve kargaşa giderme yöntemine göre olmak üzere iki farklı pencerede inceleyebiliriz. Çizelge 4.2 incelendiğinde şu sonuçlara varılmaktadır:

- COCO değerlendirme sonuçları baz alındığında en başarılı model olarak YOLO v5 ve EfficientDet göze çarpmaktadır. Her iki modelin de RNMF veri seti ile eğitilmiş olan hedef algılayıcıları en iyi iki performansa sahiptir. Bu anlamda RNMF yönteminin olumlu etkisinden bahsedilebilmektedir.
- Eğitim süreleri göz önüne alındığında yine YOLO v5 ve EfficientDet ön plana çıkmaktadır. EfficientDet daha hızlı eğitilebilmektedir. RNMF ile eğitilen hedef algılayıcının en hızlı şekilde eğitimi tamamladığı görülmektedir. Görece olarak daha yeni modeller olan EfficientDet ve YOLO

v5 modellerinin mimarilerinde gerçekleştirilen optimizasyon çalışmaları bu duruma sebep olarak gösterilebilir.

- Kargaşa giderme yöntemlerinin etkisi göz önüne alındığında, ham veri seti ile eğitilen modellerin düşük performansları göze çarpmaktadır. EfficientDet ve YOLO v5 modelleri arasında en düşük performanslar ham veri seti ile eğitilen modellerde gözlenmektedir. EfficientDet, YOLO v5 ve Faster R-CNN modelleri arasında en düşük $AP_{50}$ değerleri ham veri ile eğitilen modellerden gelmektedir. Bu durumda ham verinin düşük performansı dikkat çekicidir.
- Faster R-CNN modelinin eğitim süreleri karşılaştırıldığında en uzun süren eğitim ham veri seti ile eğitilen modele aittir. Bu durumun sebebi olarak, kargaşa giderme uygulanmamış görselde yer alan hedef dışı kargaşa verilerinin ilave hesaplamaya sebep olarak eğitimin süresine olumsuz etki etmesi gösterilebilir.
- Kargaşa giderme yöntemleri arasında çeşitli performans farklılıkları görülse de genel değerler açısından RNMF yöntemi göze çarpmaktadır. Çeşitli performans parametrelerinde üst sıralarda olan modellerin RNMF veri seti ile eğitildiği dikkate alınmalıdır.
- Kargaşa giderme yöntemlerinin etkileri görsellerde dikkat çekici olsa da derin öğrenme modelleri arasında belirgin farklar oluşmamıştır. Bu durumun sebebi veri setinde yer alan hedeflerin yüzeye uzak olması ve dolayısıyla kargaşa verisinden uzakta daha rahat tespit edilebiliyor oluşu gösterilebilir. Yüzeye daha yakın olan hedeflerde daha belirgin farklar göstermesi öngörülmektedir. Bu amaçla yüzeye yakın hedefler içeren yer altı mayın tespitinde, kargaşa giderme yöntemlerinden faydalanılması derin öğrenme modelinin performansına çok daha olumlu etki edecektir.

Modellerin eğitim sonuçları dikkate alındığında kargaşa giderme uygulamanın modellerin performansına olan olumlu etkisi gösterilmiştir. Genel anlamda kargaşa giderme uygulanmamış veri seti ile eğitilen modellerin sonuçları ortalamaların üzerine çıkmakta zorlanmaktadır. Diğer bir deyişle kargaşa giderme uygulanmış verilerin performansı, kargaşa giderme uygulanmamış modellere oranla daha yüksektir.

Derin öğrenme modelleri arasında ise EfficientDet ve YOLO v5 modelleri diğerlerinin önüne geçmektedir. Modeller özelinde karşılaşılan olağan dışı durumlar haricinde kargaşa gidermenin derin öğrenme modellerinin performansına olan etkisi olumlu görünmektedir. Bu çalışma dahilinde uygulanan kargaşa giderme yöntemlerinden RNMF, diğer yöntemlere göre ön plana çıkmaktadır.

Bu bölüme kadar paylaşılan sonuçlar simülasyon verisi üzerinde hedef algılama çıktılarını içermekteydi. Bu sonuçlara ek olarak modellerin reel verideki performanslarını görmek amacıyla, eğitim veri setlerine 10 adet reel veri de eklenerek yeniden eğitim yaptırılmıştır. Eğitim sonrası elde edilen hedef algılama modelleri reel veri üzerinde test edilerek sonuçlar gözlenmiştir.

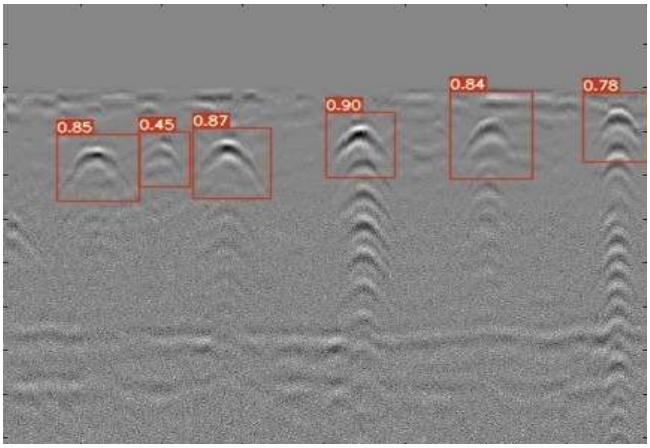
Reel verilerin elde edilme koşulları şu şekildedir:

- 1.5 Ghz antenli SIR3000 cihazı kullanılmıştır.
- Hedefler arası yatay mesafe 0.8m‘dir.
- Hedeflerin zemine dikey uzaklığı 5’er cm ara ile azalmaktadır.

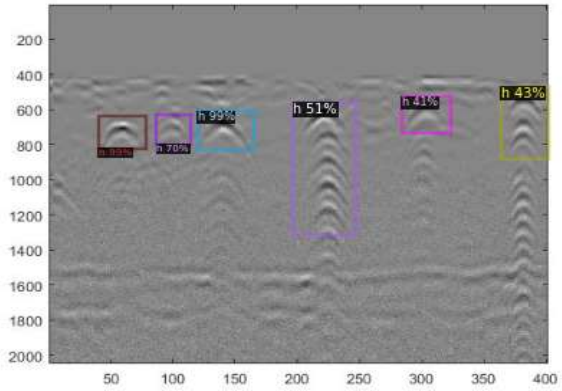
Reel veri ile yapılan testler dikkate alındığında Faster R-CNN modelinin tüm hedefleri tespit ettiği gözlenmektedir (Şekil 4.29). Test verisinde yer alan 6 hedeften 2 tanesini %99 güven skoru ile bulmayı başarmıştır. Ancak hedeflerden bir kısmının güven skorlarının %41’lere kadar gerilediği saptanmıştır. Bu nedenle Faster R-CNN modelinin hedef bulma konusunda başarılı ancak güven skoru anlamında değişken bir yapıya sahip olduğu gözlemlenmiştir.

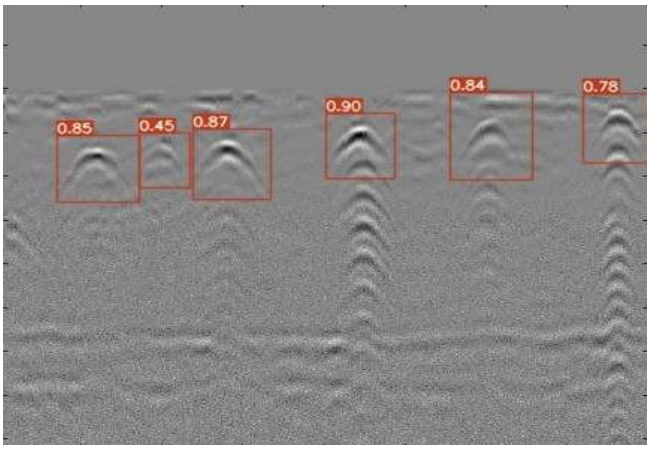
Şekil 4.30’da yer alan EfficientDet model çıktısında, daha zayıf olarak gözlemlenen hedef %45 güven skoru ile saptanırken diğer hedefler %78 - %87 arasında güven skorlarına sahiptir. Bu anlamda EfficientDet modelinin Faster R-CNN gibi tüm hedefleri saptadığı gözlemlenirken, güven skorlarının daha tutarlı olarak dağıldığı görülmüştür.

Şekil 4.31’de yer alan YOLO v5 çıktısı diğer modeller gibi tüm hedefleri saptamayı başarırken tüm güven skorları da %84 - %89 arasında değerler alarak diğer modellere oranla daha tutarlı sonuçlar ortaya koymuştur. Görece olarak daha zayıf olan hedefleri diğer modeller düşük güven skoru ile saptarken, YOLO v5 modeli yüksek güven skorları ile en iyi performansı göstermiştir.

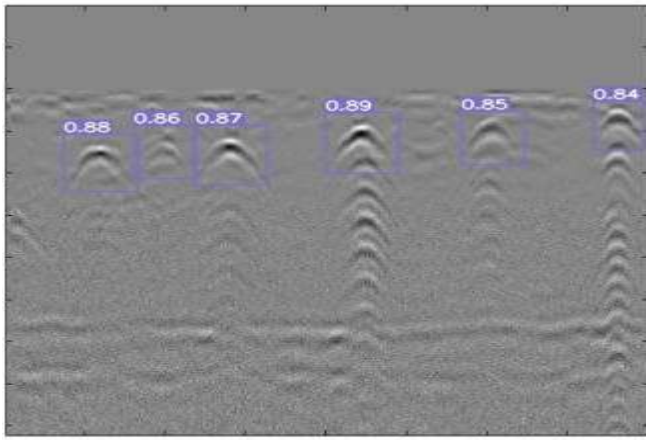



Şekil 4.29 : Faster R-CNN modelinin reel veri test örneği




Şekil 4.30 : EfficientDet modelinin reel veri test örneği




Şekil 4.31 : YOLO v5 modelinin reel veri test örneği

Reel veri testleri (Şekil 4.29, Şekil 4.30, Şekil 4.31) dikkate alındığında, tüm modellerin hedef tespit etme konusunda başarılı olduğu gözlemlenirken modeller arasındaki performans farklılıkları güven skorları ile ortaya konmuştur. Görece olarak daha eski bir nesne algılama modeli olan Faster R-CNN, hedeflerin tespitinde başarılı olurken güven skorlarında dalgalanmalar gözlemlenmiştir. EfficientDet ve YOLO v5 modellerinin Faster R-CNN'e oranla daha iyi sonuçlar verdiği izlenmiştir. Buna karşın, YOLO v5 modeli reel veri testinde hem hedef tespiti hem de güven skorları bakımından derin öğrenme modelleri arasında en iyi performansa sahip olan model olarak göze çarpmaktadır.

Simülasyon verileri ile yapılan eğitim sonuçlarına ek olarak reel veride yapılan testler ile beraber derin öğrenme modellerinin YNR verilerinde hedef tespitinde oldukça başarılı oldukları gözlemlenmektedir. Simülasyon verilerinde yapılan karşılaştırmalar sonucunda kargaşa giderme yöntemlerinin modelin başarısına olan etkileri gösterilmiştir. Kargaşa giderme yöntemleri arasında en başarılı olan yöntemin RNMF olduğu göze çarpmıştır. Simülasyon verilerinde EfficientDet ve YOLO v5 en başarılı hedef tespit modelleri olarak gözlemlenirken reel veride yapılan testlerde YOLO v5 modelinin daha başarılı olduğu saptanmıştır.

# 5. SONUÇ VE ÖNERİLER

Derin öğrenme modelleri bir çok alandaki problemlere geleneksel yöntemlere alternatif olarak çözüm olmaktadır. Bu tez kapsamında derin öğrenme modellerinden faydalanılarak YNR hedef tanıma çalışması yapılmıştır. YNR çalışmalarının sıkıntılarından biri olan kargaşa gürültülerine ve kargaşa giderme yöntemlerine değinilerek bu yöntemlerin derin öğrenme modellerine olan katkısı da araştırılmıştır.

YNR gömülü hedef tespiti, engel arkası nesne algılama vb bir çok alanda kullanılmaktadır. Nesnelerin elektriksel farklılıklarından faydalanarak hedef tanıma yoluna gider. Ölçümün yapıldığı çevre, yüzey alanının yapısı ve hedef dışı nesnelerden de etkilenebilir bir yapıdadır. Bu nedenle YNR verisi yalnızca hedef verisini yansıtmaz. Bu durum YNR hedef tanıma problemini daha da güçleştirmektedir. YNR hedef tanıma öncesinde, tarama verisinde yer alan hedef dışı çıktıların ayrıştırılması adına kargaşa giderme yöntemleri uygulanmaktadır. Hem derin öğrenme modelleri ile YNR hedef tanıma gerçekleştirmek hem de kargaşa giderme yöntemlerinin model performansına olan etkisini araştırmak adına ham YNR görselleri içeren veri setine SVD, PCA, RPCA ve RNMF kargaşa giderme yöntemleri uygulanmış ve toplamda beş farklı veri seti oluşturulmuştur.

Derin öğrenmenin çok çeşitli alanlarda kullanılmasından dolayı modellerin yapıları değişkenlik göstermektedir. Bu çalışma nesne tanıma modellerinden olan YOLO v3, Faster R-CNN, EfficientDet ve YOLO v5 kullanarak YNR hedeflerini tanımaya odaklanmıştır. Eğitim sonuçları dikkate alındığında simülasyon verilerinde EfficientDet ve YOLO v5 modellerinin performanslarının diğerlerine oranla daha iyi olduğu görülmektedir. Reel veride yapılan testlerde ise YOLO v5 modelinin derin öğrenme modelleri arasında en iyi performansı sergilediği görülmektedir. Kargaşa giderme yöntemlerinden RNMF, genel anlamda derin öğrenme modellerine en olumlu katkıyı veren yöntem olarak dikkat çekmiştir. Ancak bazı modellere özgü durumlarda farklı yöntemlerde istisnai olarak ön plana çıkmaktadır. Bu durumun oluşmasında modellerin ağ yapılarının farklılıkları etkili olabilmektedir.

Bu çalışma dahilinde derin öğrenme ile toplamda yirmi farklı YNR hedef tarama modeli oluşturulmuş olup nesne tanıma modellerinden Faster R-CNN, YOLO v3, EfficientDet ve YOLO v5 karşılaştırıldığında EfficientDet ve YOLO v5 modellerinin diğer modellerden daha yüksek performans sergiledikleri gözlenmiştir. Bu yüksek performansa ek olarak diğer modellere oranla daha kısa sürede eğitimleri tamamlanmıştır. Hem hedef yakalama performansı hem de işlem süresi göz önüne alındığında EfficientDet ve YOLO v5 bu çalışmada öne çıkmaktadır. Ek olarak kargaşa giderme uygulanmış veri setleri ile eğitilen modellerin kargaşa giderme uygulanmamış verilerle eğitilen modellerden daha performanslı olduğu saptanmıştır. Özellikle RNMF yöntemi diğerlerinin önüne geçmektedir. Bu bağlamda kargaşa gidermenin derin öğrenme modellerinin performansına olumlu yönde etki ettiği ortaya konulmuştur. Sonuç olarak bu çalışma ile YNR hedef tanıma probleminin çözümünde YOLO v5 ve EfficientDet modellerinin daha başarılı performansları olduğu görülmüş ve kargaşa gidermenin derin öğrenme modellerine olumlu yönde etki ettiği saptanmıştır.

Bu çalışmanın devamında, eğitim veri setindeki görsel sayısı artırılarak modellerin performansı artırılabilir. Derin öğrenme modellerinin veri miktarı arttıkça daha yüksek performans gösterdikleri göz önüne alınarak veri sayısındaki artış olumlu etkiye sebep olacaktır. Ayrıca bu çalışmada kullanılan kargaşa giderme yöntemlerine ek olarak, farklı kargaşa giderme yöntemleri de denenerek modellere olan etkisi gözlemlenebilir. Veri setinin kısıtlı olduğu durumlarda veri seti çeşitlendirilerek, farklı durum ve pozisyonlardaki hedeflerin tespiti kolaylaştırılabilir. Yine veri setindeki görsel sayısının yetersiz olduğu durumlarda mevcut görseller döndürme, kaydırma gibi işlemlerle veya Generative Adversarial Network (GAN) kullanılarak veri seti genişletilebilir. Bunlara ek olarak, simülasyon verilerinden oluşan veri seti gerçek verilerle zenginleştirilerek reel veriler üzerinde hedef tespit performansı artırılabilir. Gerçek verilerin yapısının simülasyon verilerine göre farklılık göstermesinden dolayı hedef yakalama performansları simülasyon verilerinde değişkenlik gösterebilmektedir. Gerçek radar görsellerinin veri setinde yer alması modelin farklı verileri tanıma kapasitesi artırılabilir.

## 6. KAYNAKLAR


[1] **Williams, R. J., & Zipser, D.** (1989). A learning algorithm for continually running fully recurrent neural networks. Neural computation, 1(2), 270-280.

[2] **Sun, X., Wu, P., & Hoi, S. C.** (2018). Face detection using deep learning: An improved faster RCNN approach. Neurocomputing, 299, 42-50.

[3] **GRM, Klemen, et al.** Strengths and weaknesses of deep learning models for face recognition against image degradations. Iet Biometrics, 2017, 7.1: 81-89.

[4] **Maqueda, A. I., Loquercio, A., Gallego, G., García, N., & Scaramuzza, D.** (2018). Event-based vision meets deep learning on steering prediction for self-driving cars. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5419-5427).

[5] **Pathak, A. R., Pandey, M., & Rautaray, S.** (2018). Application of deep learning for object detection. Procedia computer science, 132, 1706-1717.

[6] **Annan, A. P.** (2005). Ground-penetrating radar. In Near-surface geophysics (pp. 357-438). Society of Exploration Geophysicists.

[7] **Socher, R., Huval, B., Bath, B., Manning, C. D., & Ng, A.** (2012). Convolutional-recursive deep learning for 3d object classification. Advances in neural information processing systems, 25, 656-664.

[8] **Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D.** (2009). Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence, 32(9), 1627-1645..

[9] **Rowley, H. A., Baluja, S., & Kanade, T.** (1998). Neural network-based face detection. IEEE Transactions on pattern analysis and machine intelligence, 20(1), 23-38.

[10] **Parkhi, O. M., Vedaldi, A., & Zisserman, A.** (2015). Deep face recognition.

[11] **Dollar, P., Wojek, C., Schiele, B., & Perona, P.** (2011). Pedestrian detection: An evaluation of the state of the art. IEEE transactions on pattern analysis and machine intelligence, 34(4), 743-761..

[12] **Chen, X., Ma, H., Wan, J., Li, B., & Xia, T.** (2017). Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1907-1915).

[13] **Chen, C., Seff, A., Kornhauser, A., & Xiao, J.** (2015). Deepdriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the IEEE international conference on computer vision (pp. 2722-2730).

[14] **Krizhevsky, A., Sutskever, I., & Hinton, G. E.** (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

[15] **Valstar, M. F., Jiang, B., Mehu, M., Pantic, M., & Scherer, K.** (2011, March). The first facial expression recognition and analysis challenge. In Face and Gesture 2011 (pp. 921-926). IEEE.

[16] **Khan, S. H., Hayat, M., Bennamoun, M., Sohel, F. A., & Togneri, R.** (2017). Cost-sensitive learning of deep feature representations from imbalanced data. IEEE transactions on neural networks and learning systems, 29(8), 3573-3587.

[17] **Stuhlsatz, A., Lippel, J., & Zielke, T. (2012).** Feature extraction with deep neural networks by a generalized discriminant analysis. IEEE transactions on neural networks and learning systems, 23(4), 596-608.

[18] **Lienhart, R., & Maydt, J.** (2002, September). An extended set of haar-like features for rapid object detection. In Proceedings. international conference on image processing (Vol. 1, pp. I-I). IEEE.

[19] **Lowe, D. G.** (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

[20] **Freund, Y., & Schapire, R. E.** (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139.

[21] **Cortes, C., & Vapnik, V.** (1995). Support vector machine. Machine learning, 20(3), 273-297.

[22] **Ren, S., He, K., Girshick, R., Zhang, X., & Sun, J.** (2016). Object detection networks on convolutional feature maps. IEEE transactions on pattern analysis and machine intelligence, 39(7), 1476-1481.

[23] **Pitts, W., & McCulloch, W. S.** (1947). How we know universals the perception of auditory and visual forms. The Bulletin of mathematical biophysics, 9(3), 127-147.

[24] **Rumelhart, D. E., Hinton, G. E., & Williams, R. J.** (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

[25] **Hinton, G. E., & Salakhutdinov, R. R.** (2006). Reducing the dimensionality of data with neural networks. science, 313(5786), 504-507.

[26] **Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L.** (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[27] **Dahl, G., Ranzato, M. A., Mohamed, A. R., & Hinton, G. E.** (2010). Phone recognition with the mean-covariance restricted Boltzmann machine. Advances in neural information processing systems, 23, 469-477.

[28] **Deng, L., Seltzer, M. L., Yu, D., Acero, A., Mohamed, A. R., & Hinton, G.** (2010). Binary coding of speech spectrograms using a deep auto-encoder. In Eleventh Annual Conference of the International Speech Communication Association.

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

[30] **Ioffe, S., & Szegedy, C.** (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

[31] **Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A.** (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[32] **He, K., Zhang, X., Ren, S., & Sun, J.** (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[33] **Simonyan, K., & Zisserman, A.** (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[34] **Girshick, R., Donahue, J., Darrell, T., & Malik, J.** (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[35] **Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W.** (2013). Selective search for object recognition. International journal of computer vision, 104(2), 154-171.

[36] **Zhou, X., Zhuo, J., & Krahenbuhl, P.** (2019). Bottom-up object detection by grouping extreme and center points. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 850-859).

[37] **He, K., Zhang, X., Ren, S., & Sun, J.** (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904-1916.

[38] **Girshick, R.,** (2015) "Fast r-cnn," inICCV.

[39] **Ren, S., He, K., Girshick, R., & Sun, J.** (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497.

[40] **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).

[41] **Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C.** (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

[42] **Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S.** (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

[43] **Redmon, J., & Farhadi, A.** (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

[44] **Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P.** (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

[45] **Redmon, J., & Farhadi, A.** (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[46] **Tan, M., & Le, Q.** (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

[47] **Tan, M., Pang, R., & Le, Q. V.** (2020). Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10781-10790).

[48] **Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J.** (2018). Path aggregation network for instance segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8759-8768).

[49] **Ghiasi, G., Lin, T. Y., & Le, Q. V.** (2019). Nas-fpn: Learning scalable feature pyramid architecture for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7036-7045).

[50] **Zhu, L., Deng, Z., Hu, X., Fu, C. W., Xu, X., Qin, J., & Heng, P. A.** (2018). Bidirectional feature pyramid network with recurrent attention residual modules for shadow detection. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 121-136).

[51] **Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M.** (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[52] **Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., & Li, M.** (2019). Bag of freebies for training object detection neural networks. *arXiv preprint arXiv:1902.04103*.

[53]<https://blog.roboflow.com/yolov5-is-here/>, erişim tarihi: 18.01.2021

[54] **Daniels, D. J.,** (2004) Surface-Penetrating Radar, 2nd edition, IEEE Press

[55] **Jol, H. M. (Ed.).** (2008). Ground penetrating radar theory and applications. elsevier.

[56] **Gader, P. D., Mystkowski, M., & Zhao, Y.** (2001). Landmine detection with ground penetrating radar using hidden Markov models. IEEE Transactions on Geoscience and Remote Sensing, 39(6), 1231-1244.

[57] **Zoubir, A. M., Chant, I. J., Brown, C. L., Barkat, B., & Abeynayake, C.** (2002). Signal processing techniques for landmine detection using impulse ground penetrating radar. IEEE sensors journal, 2(1), 41-51.

[58] **Baker, G. S., Jordan, T. E., & Pardy, J.** (2007). An introduction to ground penetrating radar (GPR). Special Papers-Geological Society of America, 432, 1.

[59] **Kumlu, D.**(2018). New Clutter Removal Methods For Through Obstacle Target Detection(Doktora Tezi). İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul

[60] **Chomdee, P., Boonpoonga, A., & Prayote, A.** (2014, October). Fast and efficient detection of buried object for GPR image. In The 20th Asia-Pacific Conference on Communication (APCC2014) (pp. 350-355). IEEE.

[61] **Chen, C. S., & Jeng, Y.** (2011). Nonlinear data processing method for the signal enhancement of GPR data. Journal of Applied Geophysics, 75(1), 113-123.

[62] **Soldovieri, F., Catapano, I., Barone, P. M., Lauro, S. E., Mattei, E., Pettinelli, E., ... & Galli, A.** (2013). GPR estimation of the geometrical features of buried metallic targets in testing conditions. Progress In Electromagnetics Research, 49, 339-362.

[63] **Yavuz, M. E., Fouda, A. E., & Teixeira, F. L.** (2014). GPR signal enhancement using sliding-window space-frequency matrices. Progress In Electromagnetics Research, 145, 1-10.

[64] **Abujarad, F., Jostingmeier, A., & Omar, A. S.** (2004, June). Clutter removal for landmine using different signal processing techniques. In Proceedings of the Tenth International Conference on Grounds Penetrating Radar, 2004. GPR 2004. (pp. 697-700). IEEE.

[65] **Tjora, S., Eide, E., & Lundheim, L.** (2004, June). Evaluation of methods for ground bounce removal in GPR utility mapping. In Proceedings of the Tenth International Conference on Grounds Penetrating Radar, 2004. GPR 2004. (Vol. 1, pp. 379-382). IEEE.

[66] **Kumlu, D., & Erer, I.** (2018). Clutter removal in GPR images using non-negative matrix factorization. Journal of Electromagnetic Waves and Applications, 32(16), 2055-2066.

[67] **Zhang, L., Chen, Z., Zheng, M., & He, X.** (2011). Robust non-negative matrix factorization. Frontiers of Electrical and Electronic Engineering in China, 6(2), 192-200.

[68] **Kumlu, D., & Erer, I.** (2019). Improved clutter removal in GPR by robust nonnegative matrix factorization. IEEE Geoscience and Remote Sensing Letters, 17(6), 958-962.

[69] **Wall, M. E., Rechtsteiner, A., & Rocha, L. M.** (2003). Singular value decomposition and principal component analysis. In A practical approach to microarray data analysis (pp. 91-109). Springer, Boston, MA.

[70] **Garcia-Fernandez, M., Alvarez-Lopez, Y., Arboleya-Arboleya, A., Las-Heras, F., Rodriguez-Vaqueiro, Y., Gonzalez-Valdes, B., & Pino-Garcia, A.** (2017, July). SVD-based clutter removal technique for GPR. In 2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting (pp. 2369-2370). IEEE.

[71] **Verma, P. K., Gaikwad, A. N., Singh, D., & Nigam, M. J.** (2009). Analysis of clutter reduction techniques for through wall imaging in UWB range. Progress In Electromagnetics Research, 17, 29-48.

[72] **Kalika, D., Knox, M. T., Collins, L. M., Torrione, P. A., & Morton Jr, K. D.** (2015, May). Leveraging robust principal component analysis to detect buried explosive threats in handheld ground-penetrating radar data. In Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XX (Vol. 9454, p. 94541D). International Society for Optics and Photonics.

[73] **Zhou, Z., Li, X., Wright, J., Candes, E., & Ma, Y.** (2010, June). Stable principal component pursuit. In 2010 IEEE international symposium on information theory (pp. 1518-1522). IEEE.

[74] **Song, X., Xiang, D., Zhou, K., & Su, Y.** (2017). Improving RPCA-based clutter suppression in GPR detection of antipersonnel mines. IEEE Geoscience and Remote Sensing Letters, 14(8), 1338-1342.

[75] **Warren, C., Giannopoulos, A., & Giannakis, I.** (2016). gprMax: Open source software to simulate electromagnetic wave propagation for Ground Penetrating Radar. Computer Physics Communications, 209, 163-170.

[76] <https://cocodataset.org/#detection-eval>, erişim tarihi: 18.01.2021

## ÖZGEÇMİŞ

**Ad-Soyad** **:** Fatih Köprücü

## ÖĞRENİM DURUMU:

- **Lisans** **:** 2015, Bahçeşehir Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği

## MESLEKİ DENEYİM VE ÖDÜLLER:

- 2015 Bahçeşehir Universitesi Mekatronik Mühendisliği Bölüm Birinciliği
- 2016 Serasist, Arge Mühendisi
- 2020 Sekom Yazılım Geliştirme Uzmanı
- 2021 PatikaGlobal Yazılım Geliştirme Uzmanı