Fabrika İçi Ortamlarda Görsel Nesne Sezimi

Nazlıcan Gengeç

YÜKSEK LİSANS TEZİ

Elektrik Elektronik Mühendisliği Anabilim Dalı

Aralık 2020

Visual Object Detection in Factory Environments

Nazlıcan Gengeç

MASTER OF SCIENCE THESIS

Department of Electrical Electronics Engineering

December 2020

Fabrika İçi Ortamlarda Görsel Nesne Sezimi

Nazlıcan Gengeç

Eskişehir Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Elektrik Elektronik Mühendisliği Anabilim Dalı
Telekomünikasyon - Sinyal İşleme Bilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır

Danışman: Dr. Öğr. Üyesi Hasan Serhan Yavuz

Bu tez Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TUBİTAK) tarafından 116E731 numaralı "Akıllı fabrikalar için otonom taşıyıcılar ve gerekli insan-makine ve makine-makine arayüzlerinin geliştirilmesi" başlıklı projeyle desteklenmiştir.

Aralık 2020

ETİK BEYAN

Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre, Dr. Öğr. Üyesi Hasan Serhan Yavuz danışmanlığında hazırlamış olduğum "Fabrika İçi Ortamlarda Görsel Nesne Sezimi" başlıklı YÜKSEK LİSANS tezimin özgün bir çalışma olduğunu; tez çalışmamın tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; tezimde verdiğim bilgileri, verileri akademik ve bilimsel etik ilke ve kurallara uygun olarak elde ettiğimi; tez çalışmamda yararlandığım eserlerin tümüne atıf yaptığımı ve kaynak gösterdiğimi ve bilgi, belge ve sonuçları bilimsel etik ilke ve kurallara göre sunduğumu beyan ederim. 10/12/2020

Nazlıcan Gengeç

İmza

ÖZET

Endüstri 4.0'ın endüstriye getirdiği teknolojik gelişmeler ile birlikte otonom taşıyıcı araçlar akıllı fabrikaların önemli bileşenlerinden biri haline gelmiştir. Otonom taşıyıcı araçların çevresini algılayabilmesini sağlayan algılama sistemleri, bu araçların başarılı bir şekilde çalışabilmelerinde kritik rol oynamaktadır. Bilgisayarlı görme alanındaki gelişmeler araçların çevreyi ve çevredeki nesneleri görsel olarak algılayabilmesinde büyük fayda sağlamıştır. Özellikle dış ortamlarda akıllı trafik ağları için çok sayıda görsel kamera tabanlı uygulama mevcuttur. Ancak dış ortamlara kıyasla iç mekan endüstriyel ortamlarda otonom taşıma araçları için uygulama ve veri tabanı eksikliği vardır. Akıllı fabrikalarda bazı temel güvenlik ve yön işaretleri bulunmaktadır. Bu işaretler güvenlik konularında önemlidir ve otonom araçlar tarafından doğru bir şekilde tespit edilmesi gerekmektedir.

Bu çalışmada, bir akıllı fabrika ortamının benzetildiği bir laboratuvarda fabrika içi güvenlik ve yönlendirme işaretlerinden ve fabrikalarda bulunabilecek bazı önemli nesnelerden oluşan görsel bir veri kümesi oluşturulmuştur. Gerçek zamanlı çalışan güncel derin öğrenme modelleri içerisinden Faster R-CNN, SSD, RetinaNet, YOLOv3 ve YOLOv4 derin öğrenme modelleri seçilmiştir. Seçilen bu modeller hazırlanan veri kümesi üzerinde eğitilmiştir. Daha sonra bu yöntemlerin performansları doğruluk cinsinden karşılaştırılarak hangi yöntemlerin daha başarılı olduğu gözlenmiştir. Test edilen bütün yöntemler arasında en yüksek doğruluk değeri YOLOv4 modeli ile edilmiştir.

Anahtar Kelimeler : Derin öğrenme, otonom taşıyıcı araçlar, akıllı fabrikalar, güvenlik işareti tanıma

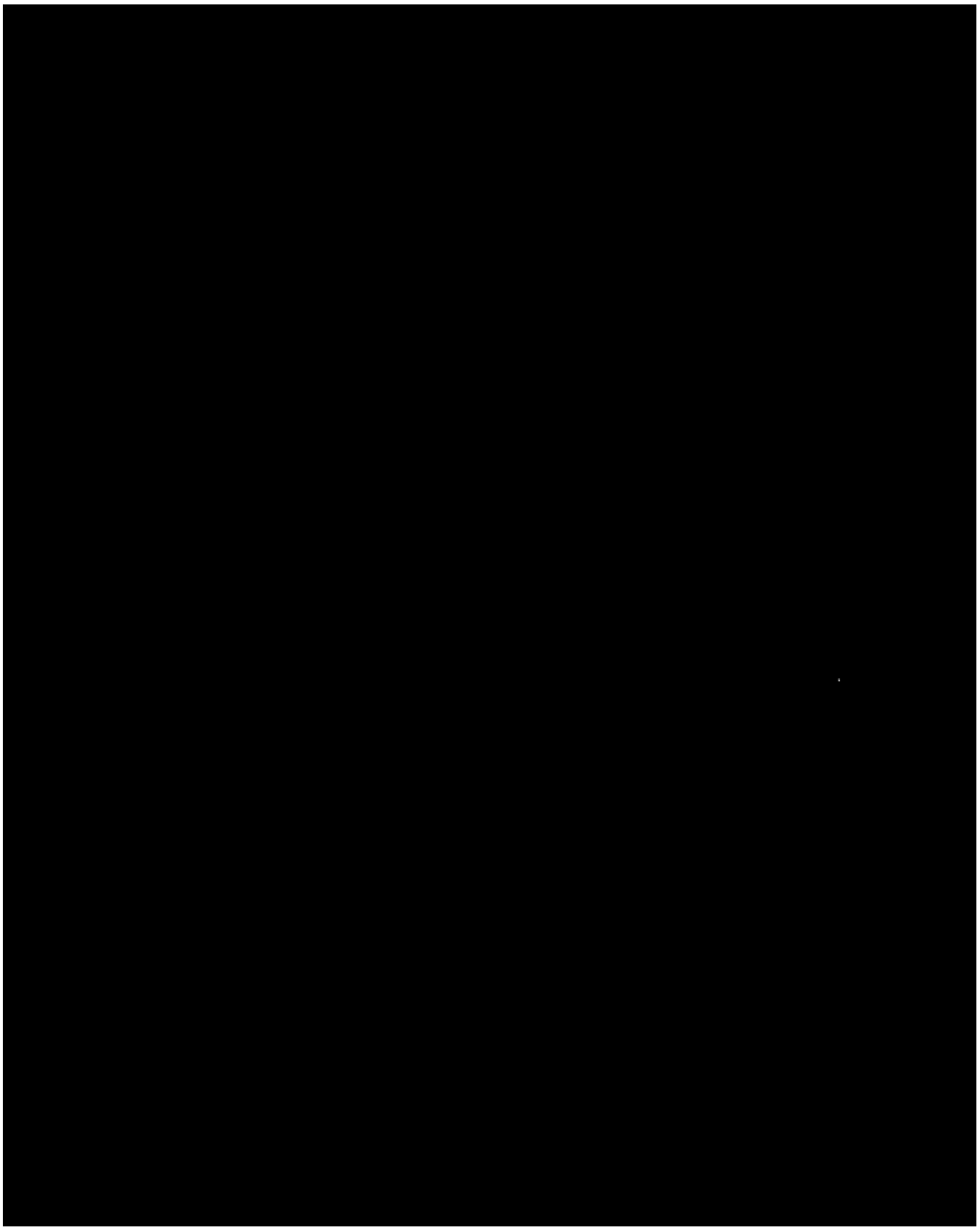
SUMMARY

With the technological developments brought by Industry 4.0 to the industry, autonomous transport vehicles have become one of the important components in smart factories. Sensing systems that enable autonomous transport vehicles to perceive their environment play a critical role in the successful operation of these vehicles. Developments in the field of computer vision studies yield the vehicles to visually perceive the environment and detect the objects in the environment. There are many applications especially for smart traffic networks in outdoor environments. However, there is a lack of application and database for autonomous transport vehicles in indoor industrial environments compared to outdoor environments. Smart factories have some basic safety and direction signs. These signs are important for safety issues and need to be correctly detected by autonomous vehicles.

In this study, a visual dataset consisting of in-factory security and orientation signs and some important objects that can be found in factories was created in a laboratory where a smart factory environment was simulated. Faster R-CNN, SSD, RetinaNet, YOLOv3 and YOLOv4 deep learning models were selected among the current deep learning models working in real time. These selected models were trained on the prepared data set. Later, by comparing the performances of these methods in terms of accuracy, it was observed which methods were more successful. The highest accuracy value among all the methods tested was with the YOLOv4 model.

Keywords : Deep learning, autonomous transport vehicles, smart factories, safety sign detection

TEŞEKKÜR



İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

ÇİZELGELER DİZİNİ

Çizelge **Sayfa**

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	**Açıklama**
L_{cls}	İkili log kaybı
L_{reg}	Yumuşatılmış L1 kaybı
ɣ	Odaklanma parametresi
N_{cls}	Sınıflandırma kaybı normalleştirme parametresi
N_{reg}	Regresyon kaybı normalleştirme parametresi
L_{conf}	Güven kaybı
L_{loc}	Yerelleştirme kaybı

Kısaltmalar	**Açıklama**
CSV	Comma Separated Values
DFT	Discrete Fourier Transform
DOT	Dominant Orientation Templates
ESA	Evrişimsel Sinir Ağları
FPN	Feature Pyramid Networks
HOG	Histogram of Oriented Gradients

SİMGELER VE KISALTMALAR DİZİNİ (Devam)

Kısaltmalar	Açıklama
HSV	Hue, Saturation, Value
LDA	Linear Discriminant Analysis
PANet	Path Aggregation Network
RELU	Rectified Linear Unit
RGB	Red, Green, Blue
RPN	Region Proposal Network
SPP	Spatial Pyramid Pooling
SVM	Support Vector Machines
XML	Extensible Markup Language
YSA	Yapay Sinir Ağları

1. GİRİŞ VE AMAÇ

Endüstri 4.0 uygulamaları ile birlikte akıllı fabrikalarda özerklik büyük ölçüde artmıştır. Bu alanda en çok kullanılan teknolojilerden biri ise otonom robotlardır. İnsan gücüne ihtiyaç duymadan fabrikalardaki taşıma işlemlerini otonom olarak gerçekleştiren araçlar olan otonom taşıyıcı araçlar (OTA), akıllı fabrikalarda önemli bir yere sahiptir. Otonom taşıyıcı araçların görevlerini başarılı bir biçimde tamamlayabilmesi için çevredeki nesneleri ve ortamı algılamaları gerekmektedir. Bu araçlar genellikle görevlerini yerine getirirken geniş, dinamik ve bilinmeyen bir fabrika ortamında gezinmek zorundadırlar. Fabrika içerisinde araçlar hareket ederken olası kazalardan kaçınmak için çevresini iyi algılayabilmeli ve insanlar gibi güvenlik kurallarına uymalıdırlar. Araçların güvenli hareketi için etraflarındaki güvenlik ve yönlendirme işaretlerini algılamaları önemlidir. OTA'ların fabrika içerisindeki güvenli hareketleri yaya geçidinde durma, park alanına park etme, yönlendirme işaretlerini algılama gibi görsel nesne tanıma uygulamaları içermektedir.

Bilgisayarla görme ve derin öğrenme uygulamalarındaki ilerlemeler araçların nesneleri ve işaretleri kolayca tespit edebilmesini sağlamaktadır. Derin öğrenme öncesi klasik yöntemlerde nesne tanıma için daha çok renk ve şekil bilgilerini betimleyen düşük seviye öznitelikler kullanılmıştır. Son zamanlarda, görsel nesne ve işaret tanıma uygulamaları, özellikle derin öğrenme algoritmalarının etkisiyle, büyük gelişmeler kaydetmiştir.

Bu tez çalışmasının amacı, fabrika içi ortamlarda otonom taşıyıcı robotların görsel nesne tespit görevlerini yerine getirerek önemli bazı güvenlik ve yönlendirme işaretlerini tanımalarını gerçekleştirmektir. İlk olarak, fabrika içi ortamlarda kullanılan bazı önemli iş güvenliği ve yönlendirme işaretleriyle tezgah, robot kolu gibi bu ortamlarda bulunan bazı önemli nesnelerin görselleri elde edilmiştir. Bu görseller bir akıllı fabrika benzetimi amacıyla oluşturulan bir laboratuvar ortamından toplanmıştır. Daha sonra, bilinen en güncel gerçek zamanlı çalışan Evrişimsel Sinir Ağları (ESA) mimarileri kullanan yöntemlerle eğitilip test edilmiştir. Bu çalışmada kullanılan yöntemler Faster R-CNN (Faster Region Based Convolutional Neural Network)(Ren vd., 2017) , SSD (Single Shot MultiBox Detector) (Liu vd., 2016) ve RetinaNet (Focal Loss for Dense Object Detection) (Lin vd., 2017), YOLOv3 (You Only Look Once v3) (Redmon ve Farhadi, 2018) ve YOLOv4(You

Only Look Once v4) (Bochkovskiy vd., 2020) yöntemleridir. Bu yöntemlerin performansları doğruluk cinsinden karşılaştırılarak hangi yöntemlerin daha başarılı olduğu gözlenmiştir.

Tez çalışmasının **Literatür Araştırması** bölümünde işaret tanıma ve nesne algılamada kullanılan yöntemler geleneksel ve modern yöntemler olmak üzere iki başlık altında incelenmiştir. **Derin Öğrenme** başlığı altında, evrişimsel sinir ağlarının genel tanımı ve yapısı, evrişimsel sinir ağı mimarileri ve kullanılan hiper parametreler ele alınmıştır. **Materyal ve Yöntem**'de ise tez çalışmasında kullanılan dört adet yöntemin detaylı açıklaması verilmiştir. **Bulgular ve Tartışma** bölümünde yapılan uygulama verilmiş ve elde edilen sonuçlar tartışılmıştır. Son olarak **Sonuç ve Öneriler** bölümünde ise çalışmada ulaşılan genel sonuçlar açılanmıştır.

2.LİTERATÜR ARAŞTIRMASI

İşaret tanıma için kullanılan geleneksel yöntemler genellikle işaretlerin özniteliklerinin çıkarılmasına odaklanmaktadır. Bu yöntemler işaretlerin renk ve şekil bilgilerini kullanır. Renk bilgisinin çıkartılması için çeşitli renk uzaylarından yararlanılır. Görüntü işlemede kullanılan renk uzaylarından bazıları RGB (Red Green Blue), HSV (Hue, Saturation, Value) ve YUV (parlaklık bileşeni (Y) ve iki renk bileşeni (U ve V)) renk uzaylarıdır. En çok kullanılan renk uzayı RGB renk uzayıdır. İşaretlerin şekil bilgileri genellikle belirli şekilde (daire, üçgen, dikdörtgen, kare ve altıgen gibi) olduğundan işaret tanımada sıklıkla kullanılmaktadır. Öznitelikler elde edildikten sonra çeşitli sınıflandırma algoritmaları ile sınıflandırma işlemi yapılmaktadır. Derin öğrenme yöntemleri olarak da bilinen evrişimsel sinir ağları son yıllarda nesne tanımada önemli başarılar elde etmiştir. Bu başarının en önemli nedenlerinden biri, özellik çıkarmanın ve sınıflandırmanın aynı anda öğrenilmesi gibi uçtan uca süreçlere olanak tanıyan entegre bir yapıdır.

2.1.İşaret Tanıma ve Nesne Algılamada Kullanılan Geleneksel Yöntemler

Wang vd. (2013 a), yasaklayıcı trafik işaretlerinin algılanması için işaretlerin renk ve şekil bilgilerini kullanan bir tanıma sistemi geliştirmiştir. Sistem dört aşamadan oluşmaktadır. Öncelikle HSV renk uzayı kullanılarak görüntüdeki kırmızı pikseller çıkartılmıştır. HSV renk uzayı değişken ışık koşullarına daha dayanıklı olduğundan RGB yerine tercih edilmiştir. İkinci aşamada oluşturulan kırmızı biteşlemden bir kenar dedektörü kullanılarak kenar biteşlemi oluşturulmuştur. Kenar dedektörü olarak Sobel Kenar Dedektörü seçilmiştir. Bir sonraki aşama olan daire algılama işlemi kademeli olarak kenar piksellerini çizgilere, yaylara ve elipslere birleştirerek gerçekleşir. Daha sonra son aşama olan doğrulama aşamasında tespit edilen dairelerin yasaklayıcı trafik işaretine ait olup olmadığını tespit edilir. Bu durumda işareterin renk bilgisinden yararlanılır. Yasaklayıcı işaret olabilmesi için kırmızı, beyaz ve siyah renkteki piksel sayısının belirli bir orandan yüksek olması istenilmiştir. Böylece sistem tamamlanmıştır.

Wang vd. (2014), dairesel trafik işaretlerinin tespiti için bir elips algılama yöntemi önermiştir. Yöntem giriş görüntüsünden elde edilen doğru parçalarından oluşan kenar

biteşlemini (edge bitmap) oluşturur ve kademeli olarak elipsler ile birleştirir. Algoritma dört aşamadan oluşmaktadır. İlk olarak girişteki biteşlem görüntüsündeki kenar pikselleri doğru parçaları şeklinde birleştirilir. Bu parçaların sayısı çok fazla olduğundan birleştirilme ihtiyacı duyar. Bu yüzden önce çizgi segmentleri daha sonra da eliptik yaylar birleştirilir. Adım adım nesne sayısı azalır. Son aşama ise doğrulama aşamasıdır. Bu şekilde işaret tanımayı gerçekleştirmiş olur. Yapılan deneyler sonucunda yöntem gerçek zamanlı uygulamalar için yeterince hızlı çalışmaktadır.

Wang vd. (2013 b), GTSDB (German Traffic Sign Detection Benchmark) veri kümesinde yer alan yasaklayıcı, tehlike ve zorunlu trafik işaretlerinin tespit edilmesi için HOG ve kabadan-inceye (coarse-to-fine) kayan pencereler yöntemini kullanan bir metot önermiştir. Orjinal HOG algoritmasında kullanılan pencere boyutları, bazı görüntülerdeki trafik işaretlerini tespit etmede yetersiz kalmaktadır. Çünkü bazı görüntülelerde işaretler çok küçük olduğundan tespitinde güçlük çekilmektedir. Bu problemi çözmek için sistemde iki farklı boyutta pencere kullanılmıştır. İlk olarak küçük boyutlu pencere ile aday bölgeler kabaca çıkartılır. Daha sonra büyük boyutlu pencere ile aday bölgeler doğrulanır. Böylece HOG algoritması iki farklı boyutta pencere ile gerçeklenmiş olur. Kaba filtreleme LDA sınıflandırıcısını kullanırken, ince filtreleme daha yüksek doğruluk için SVM sınıflandırıcısını kullanmıştır.

Supreeth ve Patil (2016), videolardan gelen trafik işaretlerini algılayan renk ve şekle dayalı bir algoritma geliştirmiştir. İşaretlerin öznitelikleri renk ve morfolojik özellikler kullanılarak çıkarılmış ve sınıflandırma aşaması sinir ağı sınıflandırıcısı ile gerçekleştirilmiştir. Çalışmada kırmızı daire şeklindeki trafik işaretleri üzerinde çalışılmıştır. İlk aşamada RGB renk uzayı ile kırmızı alanlar ayırt edilmiş, daha sonra şekil filtreleme ve dairesellik özellikleri kullanılarak işaretler tespit edilmiştir.

Gode ve Khobragade (2016), nesne tanıma için renk ve şekil özelliklerini kullanarak nesneyi görüntüden ayırt etme üzerinde çalışmışlardır. Hedef nesnenin renk bilgisi kullanılarak nesne görüntüden ayrılmış ve daha sonra şekil öznitelikleri kullanılarak görütüdeki her bir nesnenin sınırları belirlenmiştir. Her bir nesnenin sınır bilgileri DFT katsayıları ile ifade edilmiştir. Daha sonra bilinen hedef nesnenin öznitelikleri ile görüntü içerisindeki nesnelerin bilgileri karşılaştırılıp nesne tespiti işlemi tamamlanmıştır.

Gupta ve Singh (2014), çalışmalarında nesnelerin şekil özelliklerini kullanarak algılanmasını önermişlerdir. Algılama işlemi sırasıyla gerekli önişlemler, kenar tespiti, öznitelik çıkarımı ve özniteliklerin eşleştirilmesi aşamalarından oluşmaktadır. Önişlemlerde giriş görüntüsünün belirli sayıda görüntüye bölünmesi ve yeniden boyutlandırma işlemleri yer almaktadır. Öznitelik olarak nesnenin kiriş, geometrik merkez ve köşegen şekil özellikleri çıkarılmıştır. Nesnelere ait öznitelik vektörleri, veritabanında var olan nesnelerin öznitelik vektörleri ile karşılaştırılır. Hedef nesnenin öznitelik vektörü karşılaştırılan nesnenin öznitelik vektörü ile eşleştiğinde nesne algılanmış olur.

Peng (2015), doku yetersizliği sebebiyle görünüm tabanlı öznitelikler bakımından zayıf kalan objelerin tespiti için bir sistem önermiştir. Bunun için Dominant Orientation Templates (DOT) yöntemini iyileştirerek uygulamıştır. Doku yetersizliği olan nesnelerde renk bilgisi nesnenin algılanmasında önemli rol oynamaktadır. Bu nedenle çalışmada DOT yöntemi değişken ışık koşullarına dayanıklı hale getirilerek iyileştirilmiştir. Nesnenin renk bilgisi DOT metotuna eklenmiştir. Oluşturulan renk şablonları ve DOT şablonları lojistik regresyon için birleştirilmiştir. Yöntem dokusuz nesnelerin karmaşık arka planda dahi sağlıklı ve hızlı bir şekilde tespit edilmesini sağlamıştır.

2.2. İşaret Tanıma ve Nesne Algılamada Kullanılan Modern Yöntemler

Krizhevsky vd. (2017), ImageNet gibi geniş bir veri kümesi üzerinde büyük bir başarı elde eden derin evrişimsel sinir ağı modeli önermiştir. Çalışmada beş evrişim katmanı ve üç tam bağlantılı katmandan oluşan sinir ağı modeli kullanılmıştır. Bu çalışma nesne tespiti için GPU' lar üzeride ESA'ları eğiten ilk çalışmadır. O zamandan beri, daha yüksek başarı oranları nedeniyle derin öğrenmeye olan ilgi artmıştır.

Girshick vd. (2014), iki aşamalı bir nesne tanıma sistemi önermiştir. R-CNN (Region Based Convolutional Neural Network) adı verilen mimaride öncelikle seçici bir arama algoritması kullanılarak yaklaşık 2000 aday bölge oluşturulur. Daha sonra, seçilen aday bölgelerin öznitelikleri bir ESA mimarisi ile çıkarılır. Öznitelik çıkarımı aşamasında her bir bölgeden 4096 boyutlu öznitelik vektörleri çıkarılmıştır. Bunun için, Krizhevsky vd. (2017) tarafından tanımlanan ESA yapısının Caffe uygulaması kullanılmıştır. Daha sonra, her sınıf için, o sınıf için eğitilmiş SVM algoritmasını kullanarak her çıkarılan öznitelik vektörü

puanlanmıştır. R-CNN modeli VOC 2012 veri kümesi üzerinde 53.3, ILSVRC 2013 veri kümesi üzerinde ise 31.4 mAP değerine ulaşmıştır.

R-CNN mimarisinden sonra, daha hızlı versiyonları olan Fast R-CNN ve Faster R-CNN mimarileri geliştirilmiştir. Fast R-CNN ‘ de R-CNN’den 10 kat daha hızlı eğitim ve 213 kat daha hızlı test süresine sahip bir algoritma geliştirilmiştir. Ayrıca doğruluk değeri de arttırılmıştır. R-CNN’deki çok aşamalı eğitim yerine, tek aşamalı eğitim süreci uygulanmıştır. Bölge üretimi işlemi hariç, tüm işlemler uçtan uca gerçekleşmektedir (Girshick, 2015). Fast R-CNN ‘de işlem hızı ve doğruluğu arttırılmış olsa da, bölge üretimi süreci hala etkili bir model için engel oluşturmaktadır. Bu probleme çözüm olarak, Faster R-CNN ’de bölge üretimi için RPN (Region Proposal Network) önerilmiştir. Bu sayede daha az maliyetle aday bölgeler üretilmiştir. RPN her konum için nesnenin sınırlarını ve skorunu tahmin eden tamamen birleşik bir yapıdır. Dahası, RPN ve Fast R-CNN birleştirilmiş ve tek bir ağ oluşturulmuştur. RPN modülü, ağın görüntüde nereye bakacağını belirten bir dikkat mekanizması olarak kullanılmıştır. Faster R-CNN ile R-CNN ve Fast R-CNN ‘den farklı olarak tamamen uçtan uca bir ağ elde edilmiştir (Ren vd., 2017).

Otonom araçların algılama sistemlerinde daha hızlı, gerçek zamanlı işlem yapabilen yöntemlere ihtiyaç duyulmaktadır. YOLO modeli tek bir değerlendirmede sınıf olasılıklarını ve bounding box koordinatlarını tahmin edebilen bir yapıdır. Önceki çalışmalara göre çok daha hızlı işlem yapabilmektedir. Bölge öneri tabanlı yapıların aksine, YOLO eğitim ve test süresi boyunca tüm görüntüyü görür. Bu nedenle sınıf ve görünüm bilgisini birlikte kodlar (Redmon vd., 2016). YOLOv2 ve YOLOv3 versiyonları ile model üzerinde geliştirmeler yapılmıştır (Redmon ve Farhadi, 2017,2018).

YOLO algoritmasının objeleri genelleyebilmek için nispeten kaba özellikler üretmesi ve görüntü içerisindeki küçük boyutlu nesneleri algılamada zorluk çekmesi problemlerine çözüm olarak SSD (Single Shot MultiBox Detector) algoritması önerilmiştir. Belirli bir özellik haritası verildiğinde, YOLO algoritmasındaki sabit ızgaralar yerine, SSD farklı en boy oranlarına ve ölçeklere sahip bir dizi bağlantı kutularından (anchor boxes) yararlanır. Farklı boyutlarda nesneleri ele almak için ağ, farklı çözünürlüklerde birden fazla özellikli haritadan gelen tahminleri birleştirir (Liu vd., 2016).

Bir başka nesne tanıma yöntemi olan RetinaNet sinir ağlarının dikkatini yanlış sınıfın tahmin edildiği örneklere odaklayan yeni bir kayıp fonksiyonu önermiştir. RetinaNet modeli veri setlerindeki sınıf dengesizliklerinin ortaya çıkardığı problemleri çözmeye odaklanmıştır. Odak kaybı, yanlış tahmin edilen örneklerin ağırlıklarını azaltmak yerine, doğru tahmin edilen örneklerin ağırlıklarını azaltır. Böylece daha yüksek doğruluk oranı hedeflenmiştir (Lin vd., 2018).

BTSRB ve GTSRB gibi büyük ölçekli trafik veri setleri erişilebilir olduğundan beri derin öğrenme metotları trafik işareti tanıma konusunda başarı göstermiştir. Zuo vd. (2017), trafik işaretlerinin algılanmasında Faster R-CNN derin öğrenme modelini kullanmıştır. Çalışmada CCF ve UISEE tarafından gerçekleştirilen bir trafik işareti tanıma yarışması için Faster R-CNN kullanılarak bir model geliştirilmiştir.

Derin öğrenme metotlarının kullanıldığı bir diğer trafik işareti algılama çalışması da YOLOv2 modeli kullanılarak yapılan çalışmadır. Zhang vd. (2017), Çin trafik işaretlerinin algılanmasında YOLOv2 modelini kullanmıştır. Chinese Traffic Sign Dataset (CTSD) veriseti gerçek zamanlı bir algılama sistemi oluşturmak için yeni görüntüler eklenerek ve görüntüler dönüştürülerek geliştirilmiştir. Küçük boyutlu işaretlerin daha kolay algılanabilmesi için modelde bazı değişiklikler yapılmıştır. Girdi görüntüleri daha yoğun ızgaralara bölünerek daha ayrıntılı öznitelik haritaları elde edilmiştir. YOLOv2 modelinden esinlenerek üç farklı ağ modeli oluşturulmuştur. Veri kümesi üzerinde performansları karşılaştırılmış ve Model -B seçilmiştir.

Abdi ve Meddeb (2017), çalışmalarında görüntü üzerinde nesne bulmak için kullanılan bir yöntem olan Haar Cascade ve derin evrişimsel sinir ağlarını birleştirerek gerçek zamanlı bir trafik işareti tanıma sistemi önermişlerdir. Çalışmada, ilk olarak Haar Cascade yöntemi kullanılarak aday bölgeler çıkartılır. Böylece hesaplama yükü azaltılmış olur. Daha sonra oluşturulan bölgelerin bir trafik işareti olup olmadığını doğrulamak için derin öğrenme sınıflandırıcısı kullanılmıştır. Son olarak arttırılmış gerçeklik eklemek amacıyla çok sınıflı bir işaret sınıflandırıcısı pozitif aday bölgeleri alır ve her biri için doğrusal bir SVM kullanarak bir üç boyutlu trafik işareti atar. Böylelikle arttırılmış gerçeklik katılmış trafik işareti algılama ve sınıflandırma sistemi tamamlanmış olur.

Iandola vd. (2016), ImageNet veri kümesi üzerinde Alexnet modeli ile ulaşılan doğruluk değerini 50 kat daha az parametre kullanarak sağlayabildikleri SqueezeNet adı verdikleri bir ağ modeli önermişlerdir. Bir nesne tanıma modelinin doğruluğunun yanı sıra, donanımsal olarak gerçekleyebilmek için kullandığı hafıza da önemli olmaktadır. Bu modelde olduğu gibi daha az parametre ile çalışan modellerin sisteme bir çok katkısı olmaktadır. Bunlardan bir tanesi paralel eğitim sırasında daha küçük modeller daha az haberleşme gerektireceğinden avantaj sağlamaktadır. Ayrıca donanımsal entegre sırasında da kolaylık sağlamaktadır. Bu sebeplerden dolayı SqueezeNet önerilmiştir. Parametreleri azaltmak için iki strateji uygulanmıştır. Bunlardan ilki evrişim katmanlarında 3×3'lük filtreler yerine 1×1'lik filtreler kullanılmıştır. İkincisi ise giriş kanallarının sayısı azaltılmıştır. Parametreleri azaltırken doğruluğu arttırmak için ise ağ içerisinde çoğunlukla geniş boyutlu aktivasyon haritaları kullanılmıştır.

Lee ve Kim (2018), önceki trafik işareti tanıma sistemlerinden farklı olarak tespit etme ve işaretin sınırlarını belirleme işlemlerini birlikte gerçekleştirmişlerdir. Sınır tespiti problemini ESA tabanlı bir poz ve şekil tespiti problemine dönüştürerek çözmüşlerdir. Önerilen yöntemde öncelikle giriş imgesinden bir ağ ile öznitelik haritaları elde edilir. Bu çalışmada ağ modeli olarak SSD yapısı tercih edilmiştir. Modelin çıktısı olarak elde edilen öznitelikler iki ayrı evrişimsel katmandan geçer. Bu katmanların isimleri poz regresyon ve şekil sınıflandırma katmanlarıdır. Bu katmanlar çıktıları iki boyutlu poz değerlerine ve şekil sınıf olasılıklarına dönüştürürler. Son olarak ise bu iki çıktı birleştirilip, işaretin sınır köşeleri belirlenir. Çalışmada kullanılan veri kümesi bir kamera ile gerçek görüntüler toplanarak elde edilmiştir.

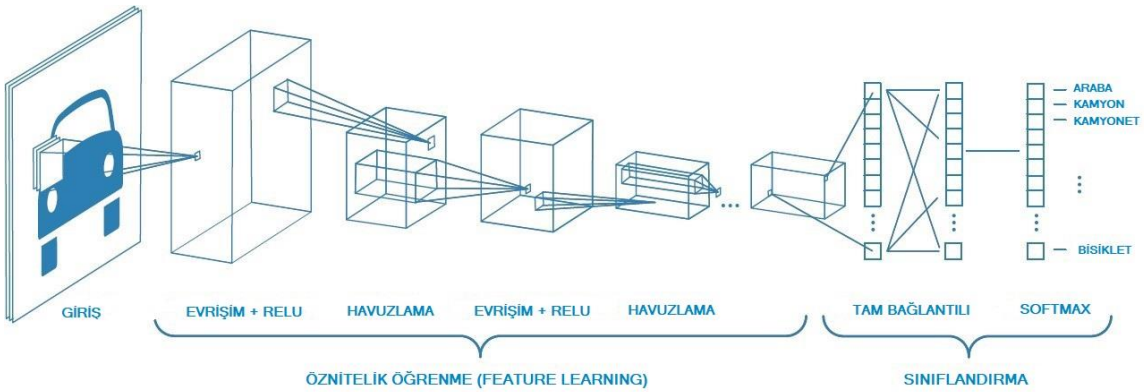
2.3. Derin Öğrenme

Derin öğrenme, yapay zekâ alanında hızlı büyüyen popüler bir makine öğrenme yaklaşımıdır. Derin öğrenme modellerinin temeli Evrişimsel Sinir Ağlarına dayanmaktadır. Bu bölümde evrisimsel sinir ağları detaylı bir şekilde anlatılıp, temel mimarilerin yapıları açıklanmıştır. Derin öğrenme modellerinin eğitim aşamasında kullanılan parametreler ve test aşamasında kullanılan değerlendirme ölçütleri anlatılmıştır.

2.3.1. Evrişimsel sinir ağları

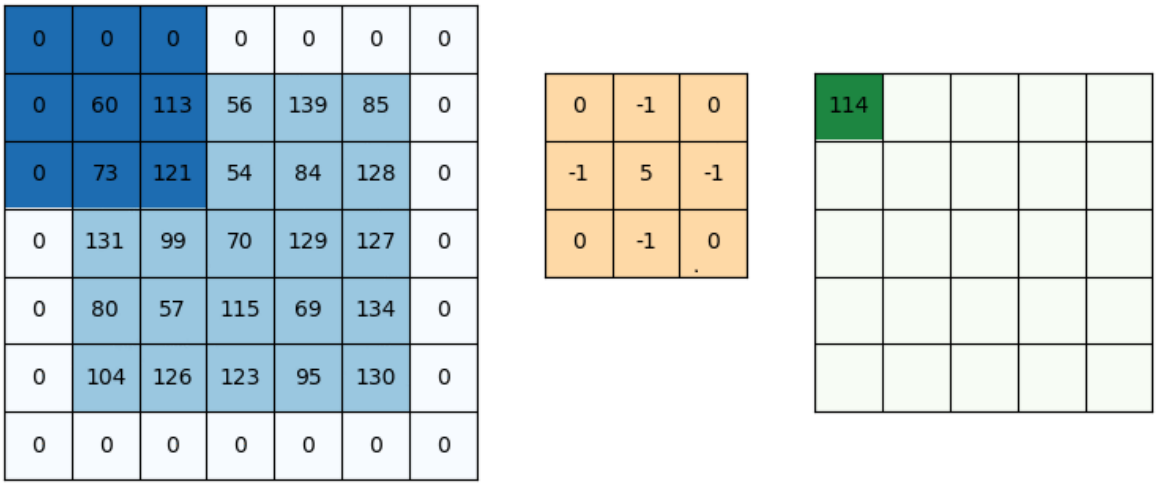
Evrişimsel sinir ağları derin öğrenme mimarisinin temelini oluşturmaktadır. ESA yapısı ilk olarak Lecun tarafından 1989 yılında tanıtılmıştır (LeCun vd.,1989). Bir evrişimsel sinir ağı, girdi olarak alınan görüntüde var olan çeşitli yönleri ya da nesneleri birbirinden ayırt edebilen bir derin öğrenme mimarisidir. Son dönemlerde, evrişimsel sinir ağları özellikle nesne tanıma çalışmalarında büyük başarı göstermiştir. Bu başarının en önemli nedenlerinden biri öznitelik çıkarma ve sınıflandırma gibi işlemleri uçtan uca birleştiren bütünleşik bir yapının olmasıdır. ESA'lar geleneksel Yapay Sinir Ağları (YSA) yapısı gibi nöronlardan oluşmaktadır. ESA ve YSA yapısı arasındaki en göze çarpan fark ESA yapısının özellikle bir görüntüdeki örüntü tanıma görevinde kullanılıyor olmasıdır. ESA yapısı görüntüye özgü özelliklerin mimariye kodlayarak ağı görüntü odaklı görevlere uygun olarak oluşturur (Shea ve Nash, 2015).

Evrişimsel Sinir Ağları mimarisi temel olarak öznitelik çıkarımı ve sınıflandırma adı verilen iki bileşenden oluşmaktadır. Bu iki bileşen en genel haliyle üç tip katman içermektedir. Bu katmanlardan evrişim katmanı ve havuzlama katmanı öznitelik çıkarımında kullanılırken, tam bağlı katman sınıflandırma işleminde kullanılmaktadır. Bu üç katmanın yanında bazı ara katmanlar da mevcuttur. Hepsi birlikte ESA yapısı, evrişim katmanı (Convolution Layer), aktivasyon katmanı (Activation Layer), havuzlama katmanı (Pooling Layer), tam bağlantılı katman (Fully-Connected Layer), seyretme katmanı (Dropout Layer) ve sınıflandırma (Classification) aşamalarından meydana gelmektedir (Şekil 2.1).



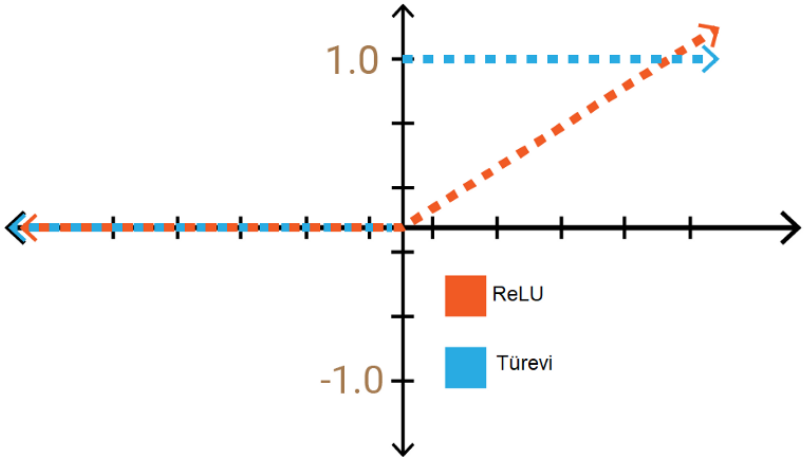
Şekil 2.1. Evrişimsel sinir ağları katmanları (Saha,2018)

Evrişim katmanı, ESA yapısının temel katmanı evrişim katmanıdır. Evrişim katmanı, giriş görüntülerini bir dizi filtre işleminden geçirir. Bir modeldeki ilk evrişim katmanı kenarlar, çizgiler ve köşeler gibi düşük seviyeli öznitelikleri çıkartır. Daha yüksek seviyeli katmanlar ise daha yüksek seviyeli öznitelikleri çıkartır (Hijazi vd., 2015). Katman sayısı modele veya kullanılacağı göreve göre farklılık gösterebilir. İlk evrişim katmanında yer alan tüm nöronlar giriş görüntüsündeki alıcı alandaki piksellere bağlanır. İkinci evrişim katmanında ise, her bir nöron birinci katmandaki küçük bir dikdörtgen içinde bulunan nöronlara bağlanır. Bu mimari sayesinde düşük seviyeliden yüksek seviyelere doğru görüntüden öznitelikler elde edilir. Katmanın isminden de anlaşılacağı üzere, giriş görüntü matrisinin ve filtre matrisinin konvolüsyonu öznitelik haritasını verir. Evrişim katmanını oluştururken bazı parametrelerin belirlenmesine ihtiyaç vardır. Bunlar filtre boyutu, sıfır ekleme/doldurma ve adım aralığıdır. Filtre giriş görüntüsüne tam olarak uymadığında sıfır ekleme/doldurma kullanılır. Giriş görüntüsüne sığdırmak için görüntü matrisi sıfırlarla doldurulur. Giriş görüntü matrisi genellikle filtre matrisinden daha büyüktür. Bu nedenle, filtrenin görüntü matrisinin üzerinde kaydırılması gerekir. Giriş matrisine kaydırılan piksel sayısına adım aralığı adı verilir. Evrişim katmanının çıktısı olarak oluşan öznitelik haritasının boyutu $O = \frac{I-F+2P}{S} + 1$ formülü ile hesaplanır. Burada O öznitelik haritasının boyutunu, I giriş görüntüsünün boyutunu, F filtre boyutunu, P sıfır ekleme miktarını ve S ise adım sayısını ifade etmektedir.



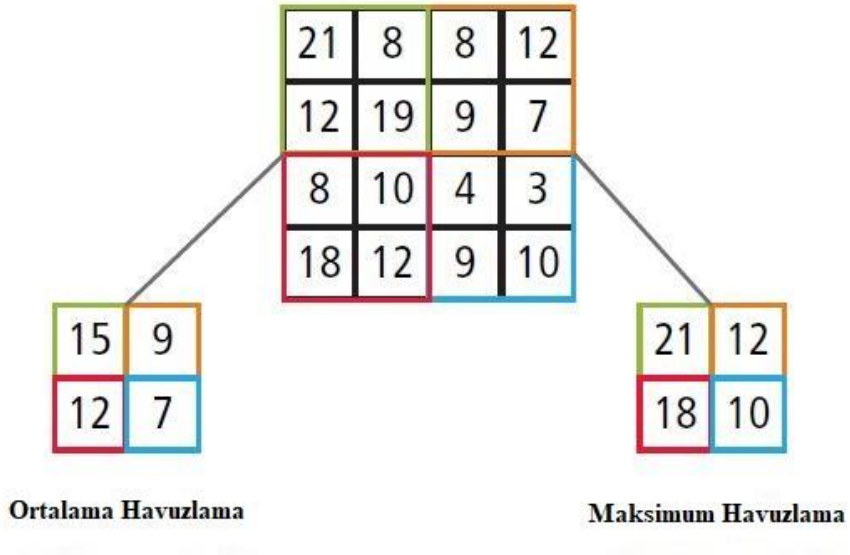
Şekil 2.2. İki boyutlu bir evrişim işlemi örneği

Şekil 2.2'de gösterildiği gibi 5×5' lik bir giriş görüntüsüne P=1 ve S=1 olamk üzere 3×3'lük bir filtre uygulanırsa, çıktı 5×5'lik bir öznitelik haritası olmaktadır. Her evrişim katmanından sonra modeli doğrusallıktan kurtarmak için bir aktivasyon fonksiyonu eklenir. Aktivasyon fonksiyonuna modele doğrusal olmayan gerçek dünya özelliklerini tanıtmak için ihtiyaç duyarız. Bir yapay sinir ağı modelinde aktivasyon fonksiyonu kullanılmadan görüntü, video, yazı ve ses gibi karmaşık gerçek dünya bilgilerini içeren verilerin sağlıklı bir şekilde öğretilmesi çok güçtür. Bir çok aktivasyon fonksiyonu mevcuttur. Bunlardan bazıları basamak fonksiyonu, sigmoid fonksiyonu, tanh fonksiyonu, ReLU fonksiyonu ve softmax fonksiyonudur. Bunlardan en çok tercih edileni ReLU (Rectified Linear Unit) fonksiyonudur (Şekil 2.3).



Şekil 2.3. ReLU fonksiyonu

Havuzlama katmanı, evrişim katmanlarından sonra uygulanır. Bu katman alt örnekleme olarak da adlandırılmaktadır. Katmana gelen giriş görüntüsünü örnekleyerek uzamsal boyutunu düşürür. Modelin parametre sayısını azaltarak hesaplama yükünü ve bellek kullanımını azaltır. Bir havuzlama katmanında, bir filtre içinde bulunan her nöron, evrişimsel tabakaya benzer şekilde bir önceki tabakadaki nöronların çıkışlarına bağlanır. Evrişim katmanında olduğu gibi bu katmanda da giriş görüntüsüne belli boyutlarda filtreler uygulanır. Modellerde iki tür havuzlama türü kullanılmaktadır. Maksimum havuzlamada, giriş matrisi üzerinde dolaştırılan filtredeki her bir kare için en yüksek değer çıkış matrisine yazılır. Ortalama havuzlamada ise her karenin ortalamasının alınıp çıkış matrisine yazılır (Şekil 2.4).



Şekil 2.4. Ortalama ve maksimum havuzlama

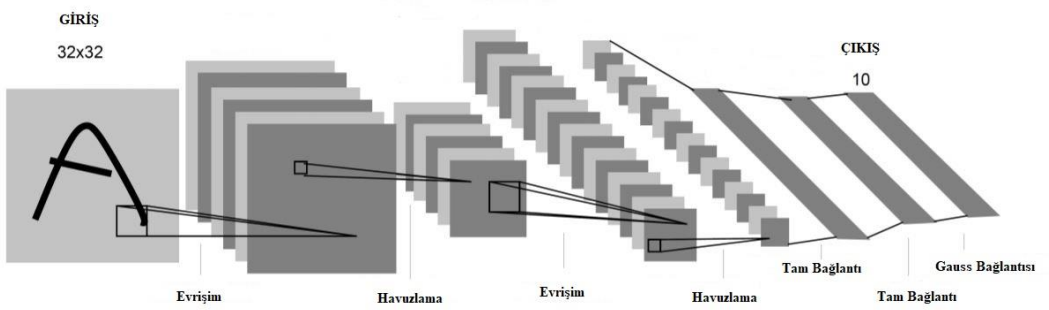
Tam bağlantılı katman, kendinden önceki katmandaki tüm nöronlarla birbirine bağlı olduğu için bu katmana tam bağlantılı katman adı verilmektedir. Bu katmanda öznitelik haritası bir vektör haline getirilmektedir. Bu katmana kadar öznitelik çıkarımı yapılırken tam bağlantılı katmanda sınıflandırma işlemleri başlar. Tam bağlantılı katmanda, kendinden önceki katmandan almış olduğu verileri çıkışta vermesi gereken sınıf sayısına dönüştürme işlemi gerçekleşir.

2.3.2. Evrişimsel sinir ağları mimarileri

Özellikle ImageNet gibi büyük ölçekli veri setleri erişilebilir olduktan sonra evrişimsel sinir ağlarına olan ilgi artmış ve birçok ESA modeli önerilmiştir. Bu modellerin bazıları ImageNet Large Scale Visual Recognition Challenge (ILSVRC) gibi yarışmalarda başarı göstermiştir. Bu modellerin başarısı verilen veri kümesindeki hata oranı ile ölçülmektedir. LeNet-5, AlexNet, VGGNet ve ResNet mimarileri kısaca tanıtılmıştır.

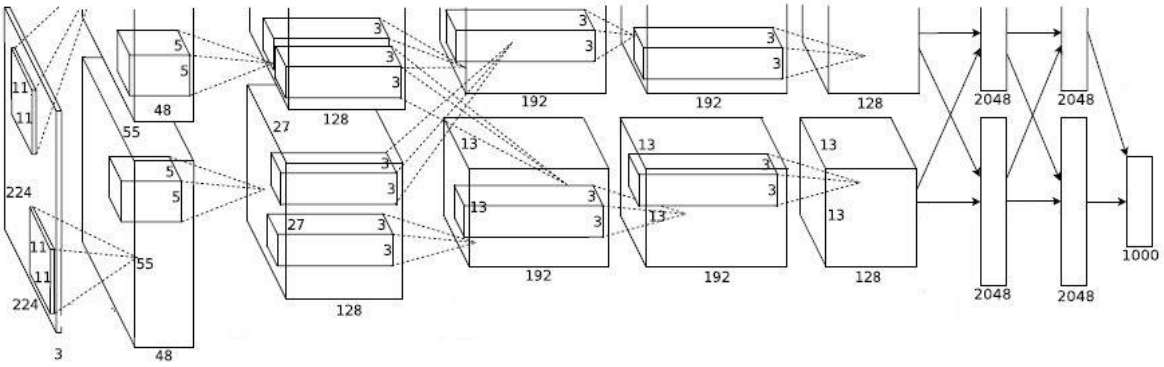
LeNet-5, özellikle MNIST veri kümesi gibi el yazması rakamları algılamak için geliştirilmiş en çok bilinen ESA mimarisidir (Lecun vd., 1998). MNIST veri kümesi 32×32 boyutlu el yazması rakamların görüntülerini içeren 10 sınıflı bir veri kümesidir. LeNet-5 mimarisi günümüzdeki ESA yapılarına kıyasla çok basit bir yapıya sahiptir. Toplam yedi tane katmana sahiptir. Bu katmanların üç tanesi evrişim katmanı, iki tanesi havuzlama

katmanı, bir tanesi tam bağlantılı katman ve sonuncusu ise çıkış katmanıdır. Evrişim katmanında 5×5 filtreler ile işlem yapılır. Havuzlama katmanında ise 2×2'lik ortalama havuzlama kullanılmıştır. Bütün ağda aktivasyon fonksiyonu olarak tanh sigmoid fonksiyonu kullanılmıştır. LeNet-5 mimarisi Şekil 2.5'de gösterilmiştir.



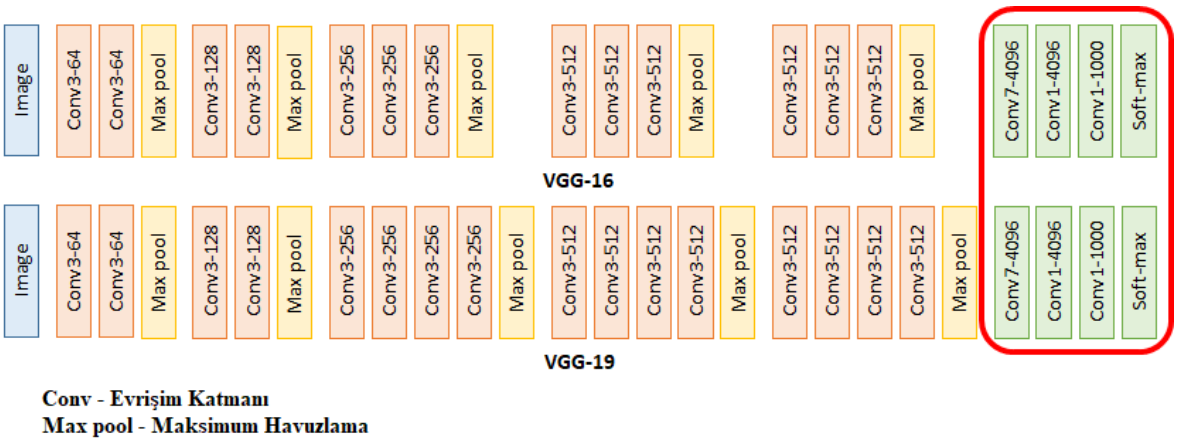
Şekil 2.5. LeNet-5 mimarisi (Lecun vd., 1998)

AlexNet, 2012 yılında ILSVRC yarışmasını kazanan bir ESA mimarisidir (Krizhevsky vd., 2017). Yarışmada test verileri üzerinde % 37.5 ve % 17.0 ilk-1 ve ilk-5 hata oranlarını elde etmiştir. AlexNet yapısı ile 1000 sınıflı 1,2 milyon yüksek çözünürlüklü görüntü sınıflandırılmıştır. AlexNet toplam sekiz katmandan oluşmaktadır. Bu katmanlardan beş tanesi bazılarını maksimum havuzlama katmanı takip eden evrişim katmanlarından, iki tanesi tam bağlantılı katmandan ve bir tanesi ise tam bağlantılı çıkış katmanından meydana gelmektedir. AlexNet, LeNet mimarisinden farklı olarak aktivasyon fonksiyonu olarak tanh yerine ReLU fonksiyonunu kullanmıştır. Mimaride eğitilen görüntü sayısı, kullanılan bir adet GTX 580 GPU tarafından eğitilemeyecek kadar fazladır. Bu sebeple model iki GPU tarafından eğitilmiştir. AlexNet mimarisi Şekil 2.6'da gösterilmiştir.



Şekil 2.6. AlexNet Mimarisi (Krizhevsky vd., 2017)

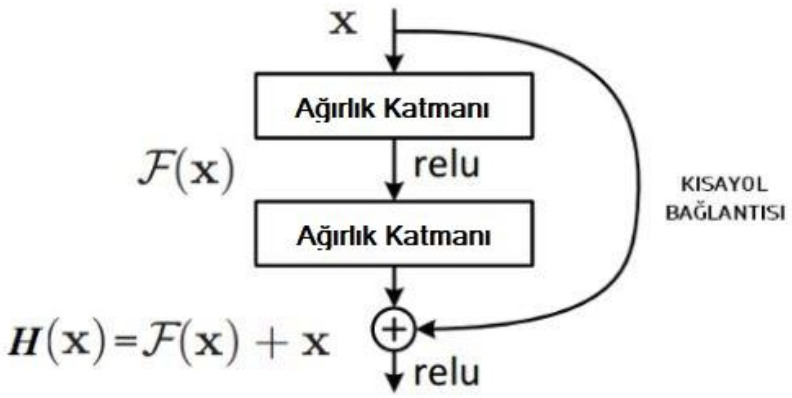
VGGNet, AlexNet temeline dayanan bir sinir ağı modelidir (Simonyan ve Zisserman, 2015). AlexNet'e göre daha küçük boyutlu filtreler kullanılmış fakat model daha derin hale getirilmiştir. AlexNet modelindeki 11×11 boyutlu filtreler yerine VGGNet tüm evrişim katmanlarında 3×3 boyutlu filtreler kullanmıştır. Filtre boyutu küçültülmüş fakat modelin derinliği arttırılarak, derinliğin algılamadaki doğruluğu nasıl etkilediği incelenmiştir. VGGNet 16 ve 19 tane ağırlık katmanına sahip VGG16 ve VGG19 olarak adlandırılan iki versiyona sahiptir. VGG16 ve VGG19 modellerinin mimarisi Şekil 2.7' de gösterilmiştir. VGGNet, AlexNet'ten daha iyi performans göstermesine ve daha basit bir tasarıma sahip olmasına rağmen, daha fazla hesaplama gerektiren parametreye sahiptir. Model, 1000 sınıfa ait 14 milyondan fazla görüntünün veri kümesi olan ImageNet'te %92,7 ilk beş test doğruluğunu elde etmiştir (ILSVRC 2014).



Şekil 2.7. VGG16 ve VGG19 ağları

He vd. (2016), % 3,6'lık hata oranı ile 2015 yılında ILSVRC yarışmasını kazanan ResNet modelini önermişlerdir. ResNet modelinin çıkış noktası başarı ile katman sayısı arasındaki ilişki olmuştur. ResNet öncesi modellerde katman sayısı arttıkça başarının arttığı gözlemlense de, çalışmada kaybolan gradyan (vanishing gradient) probleminden dolayı ağ derinleştikçe hata oranının arttığı anlaşılmıştır. Gradyanların kaybolması katman sayısının fazlalılığından dolayı, çarpanların ilk katmana geri dönene kadar etkisi yok olması problemidir. Çalışmada bu problemin önüne geçmek ve daha derin ağlar oluşturabilmek için kısayol bağlantıları içeren yeni bir model önerilmiştir. Bu kısayol bağlantısı Şekil 2.8'de gösterilmiştir. Bu bağlantı sayesinde çarpanlar katmanlardan geçerken kaybolsa dahi,

tekrardan ağa girdi olarak eklenmektedir. Böylece çarpanlar kaybolmadan 152 katmanlı derin bir sinir ağı oluşturmuşlardır.



Şekil 2.8. ResNet mimarisinde kullanılan bloklar (He vd., 2016)

2.3.3. Hiper parametreler

Bir evrişimsel sinir ağı modelinin en yüksek başarı ile çalışabilmesi için kullanılan parametrelerin en doğru şekilde seçilmesi gerekmektedir. Derin öğrenme modellerinde olduğu gibi kullanılan veri kümesine, modele ve amaca göre değişen parametrelere hiper parametreler denilmektedir. En sık kullanılan hiper parametrelerden bazıları mini-batch boyutu, öğrenme hızı, momentum, eğitim tur sayısı, kayıp fonksiyonu ve kullanılan optimizasyon algoritması olarak sayılabilir.

Mini-batch boyutu, çok büyük boyutlu veri kümeleri ile çalışırken tüm verilerinin aynı anda işleme alınması büyük boyutlarda hafıza gerektirmektedir. Ayrıca, tüm verilerin işlenmesi ciddi anlamda zaman almaktadır. Hafıza ve zaman maliyetinden dolayı, mini-batch adı verilen bir biçimde veri kümesi parçalar halinde işlenmektedir. Mini-batch parametresi olarak belirlenen değer, modelin aynı anda kaç veriyi işleyeceği anlamına gelmektedir. Modelin başarısı için bu değerin en uygun şekilde belirlenmesi gerekmektedir.

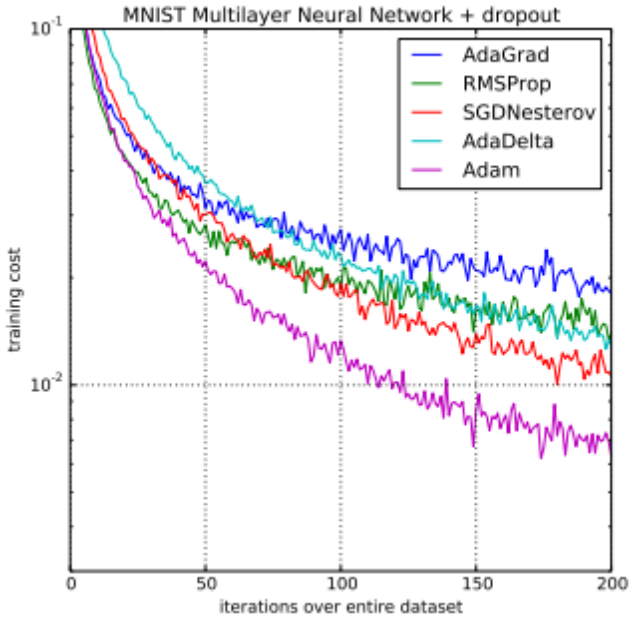
Öğrenme hızı sinir ağlarını eğitirken kullanılan en önemli hiper parametrelerden bir tanesidir. Öğrenme hızı, model ağırlıkları her güncellendiğinde tahmin edilen hataya yanıt olarak modelin ne kadar değiştirileceğini kontrol eden bir hiper parametredir (Brownlee,

2019). Bu değer sabit olarak belirlenebildiği gibi, sürekli artan bir şekilde de ayarlanabilir. Bu değer çok büyük olduğunda salınıma sebep olduğu gibi, eğer çok küçük seçilirse de öğrenim çok uzun sürebilmektedir. Model eğitilirken genellikle öğrenim hızı başlangıçta yüksek tutulup, zamanla azaltılmaktadır. Öğrenme hızı mevcut durumun bir sonraki adımı ne kadar etkileyeceğini belirlerken, momentum değeri geçmiş adımların sonraki adımları ne oranda etkileceğini belirlemektedir. Momentum parametresi için uygun aralık 0,8 ve 0,99 iken, genellikle 0,9 kullanılmaktadır.

Eğitim tur sayısı (Epoch) bir derin öğrenme modelinin eğitim sürecinde, veri kümesinin tamamının geçiş sayısını belirtmektedir. Başka bir ifadeyle, bir tur sayısı veri kümesinin bir kez sinir ağı üzerinden ileri ve geri aktarılmasını ifade eder. Ağırlık değerleri her bir tur sayısında adım adım hesaplanarak güncellendiği için genellikle ilk eğitim turlarında başarım düşük olmaktadır. Tur sayısı arttırıldığında başarımda artacak fakat belirli bir tur sayısından sonra başarım artış hızı çok azalacaktır (Çarkacı, 2018). Eğitim tur sayısının ne olması gerektiği hakkında doğru bir cevap bulunmamaktadır. Başarım artış hızının gözle görülür şekilde azaldığı noktada modelin eğitimi sonlandırılabilir.

Kayıp fonksiyonu eğitim örneğinin çıktısı ile istenilen referans çıktı arasındaki farkı (hatayı) belirlemek için kullanılan bir parametredir. Bu fonksiyonun çıktısına göre sonraki adımlarda ağırlıklıklar güncellenerek hata azaltılmaya çalışılmaktadır. Ağ tüm eğitim verisinden her geçiş yaptığında ağırlıkların bağlantısına göre hesaplanır. En sık kullanılan kayıp fonksiyonları Sigmoid, Softmax, Kare Öklid Mesafesi ve Cross-Entropy gibi fonksiyonlardır. Veri kümesi ve model bilgilerine göre en uygun kayıp fonksiyonu seçilerek hata en aza indirilmeye çalışılmalıdır.

Optimizasyon algoritmaları ağırlıkları güncelleyerek ağın kayıp fonksiyonunu en aza indirmek için kullanılırlar. Optimizasyon algoritmasının seçimi doğru sonuçlar üreten bir modelin eğitilmesinde önemli rol oynamaktadır. En çok kullanılan algoritmalardan bazıları Gradient Descent, Stochastic Gradient Descent, Adam, AdaGrad ve RmsProp olmaktadır. Her algoritmanın başarı oranı ve hızı birbirinden farklıdır. Optimizasyon algoritmalarının MNIST veri kümesi üzerindeki çalışma zamanı grafiği Şekil 2.9'da gösterilmektedir.



Şekil 2.9. Bazı optimizasyon algoritmalarının çalışma zamanı grafiği (Çarkacı, 2018)

2.3.4. Performans değerlendirme ölçütleri

Sınıflandırma problemlerinde sinir ağı modellerinin performansları karşılaştırılırken en çok kullanılan ölçüt Ortalama Kesinlik (Average Precision – AP) ölçütüdür. Ortalama kesinliğin nasıl hesaplandığını anlayabilmek için öncelikle karışıklık matrisini, kesinlik (precision) ve duyarlılık (recall) birimlerini açıklamak gerekmektedir. Literatürde sınıflandırma işlemlerinin performanslarını değerlendirebilmek için karışıklık matrisleri (confusion matrix) sıklıkla kullanılmaktadır. Karışıklık matrisleri tahminlerin doğruluğu hakkında bilgiler veren birer ölçüm tablosudur. Şekil 3.10'da gösterildiği gibi tabloda dört ayrı değer mevcuttur. Gerçek pozitif değeri gerçekte pozitif olan ve pozitif olarak tahmin edilen örnek sayısını, sahte negatif gerçekte pozitif olan ve negatif tahmin edilen örnek sayısını, sahte pozitif gerçekte negatif olan ve pozitif tahmin edilen örnek sayısını, gerçek negatif ise gerçekte negatif olan ve negatif tahmin eden örnek sayısını ifade etmektedir.

Matris üzerinde oluşan bu değerlerden yararlanılarak doğruluk (accuracy), kesinlik (precision) ve duyarlılık (recall) gibi değerler hesaplanabilmektedir.

		Tahmin Edilen Sınıf	
		Pozitif (P)	**Negatif (N)**
Gerçek Sınıf	**Pozitif (P)**	Gerçek Pozitif (True Positive - TP)	Sahte Negatif (False Negative - FN)
	Negatif (N)	Sahte Pozitif (False Positive - FP)	Gerçek Negatif (True Negative - TN)

Şekil 3.10. Karışıklık Matrisi

Doğruluk (Accuracy) : Doğruluk (Accuracy), doğru olarak yapılan tahminlerin tüm tahminlere oranını ifade etmektedir. Doğruluk değeri Denklem 3.1' e göre hesaplanmaktadır.

$$Doğruluk = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.1}$$

Kesinlik (Precision) : Yapılan tahminlerin ne kadar doğru olduğunu açıklamaktadır ve Denklem 3.2'deki gibi hesaplanmaktadır.

$$Kesinlik = \frac{TP}{TP+FP} \tag{3.2}$$

Duyarlılık (Recall) : Doğru yapılan tahminlerin yüzdesini vermektedir ve Denklem 3.3 'deki gibi hesaplanmaktadır.

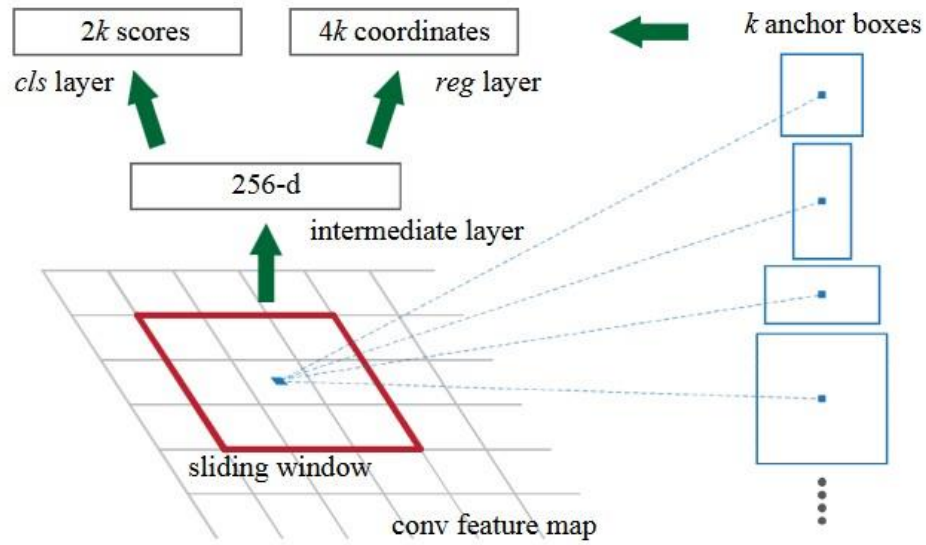
$$Duyarlılık = \frac{TP}{TP+FN} \tag{3.3}$$

3.MATERYAL VE YÖNTEM

Bu çalışmada CNN tabanlı yöntemlerden Faster R-CNN, SSD, RetinaNet, YOLOv3 ve YOLOv4 modelleri eğitilip test edilmiştir. Test edilen bu yöntemlerden Faster R-CNN iki aşamalı bir yöntem olarak kabul edilirken, geri kalanlar tek aşamalı dedektörlerdir.

3.1. Faster Region Based Convolutional Neural Network – (Faster R-CNN)

Faster R-CNN (Ren vd., 2017), RPN ve Fast Region Based Convolutional Neural Network (Fast R-CNN) dedektöründen oluşan iki aşamalı bir nesne algılama ağıdır. RPN yapısı önerdiği her bölge için dikdörtgen bir bölge ve bir puanlama üretmek üzere görüntüyü girdi olarak alan bir evrişimsel sinir ağıdır. RPN daha önceki bölge öneri yöntemlerine göre bölge öneri süresini kısaltarak, Faster R-CNN yönteminin gerçek zamanlı olmasını sağlamıştır. RPN mimarisi Şekil 3.1' de gösterilmektedir.



Şekil 3.1. Bölge öneri ağı mimarisi

Faster R-CNN daha önceki R-CNN yöntemlerinden farklı olarak, farklı ölçeklerde anchor kutularını (belirli bir yükseklik ve genişlikte önceden tanımlanmış sınırlayıcı kutular kümesi) kullanmaktadır. Bu sayede farklı en boy oranlarına sahip bölge önerileri daha kolay bir şekilde oluşturulmaktadır. Ağ, öznitelik haritası üzerinde 3×3 boyutlu bir pencere

gezdirip, k tane anchor kutusu üretmektedir. Her noktada, üç farklı en boy oranı ve ölçeğe sahip dokuz tane anchor kutusu elde edilmektedir. Son olarak öznitelik vektörü de her bir kayan pencereden elde edilmektedir. Daha sonra da sınıflandırma işlemi gerçekleşmektedir. Faster R-CNN modelinde kullanılan kayıp fonksiyonu, Fast R-CNN modelinde kullanılan ile aynı olup Denklem 3.1 'de verilmiştir.

$$L(p_i, t_i) = \frac{1}{N\text{cls}}\left(\sum_i L_{\text{cls}}(p_i, p_i*) + \lambda \frac{1}{N_{reg}} \sum_i p_i * L_{reg}(t_i, t_i*)\right) \quad (3.1)$$

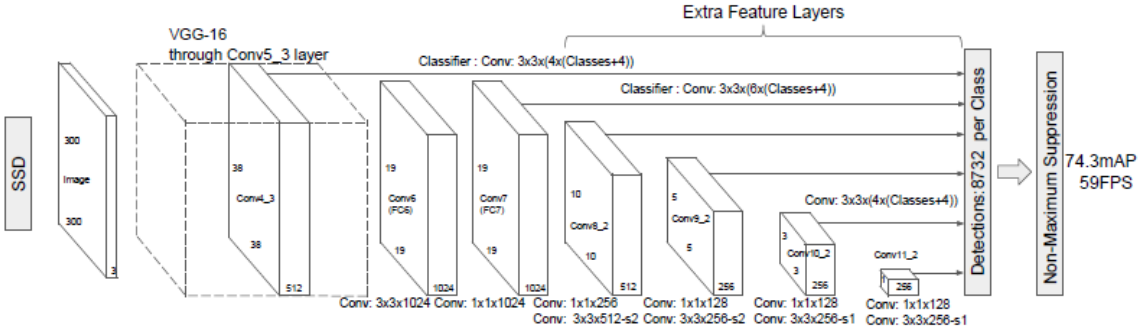
Bu denklemde, L_{cls} ikili log kaybını, L_{reg} yumuşatılmış L1 kaybını, p_i i.anchor kutusunun nesne olma olasılığını ve p_i* ise referans (ground truth) etiket bilgisini ifade etmektedir. t_i ve t_i* terimleri de sırasıyla tahmin edilen sınırlayıcı kutuyu (bounding box) ve referans sınırlayıcı kutuyu temsil etmektedir. Regresyon ve sınıflandırıcı kaybı N_{cls} ve N_{reg} ile normalleştirilmektedir. Faster R-CNN yüksek doğruluk ve hız sağlamaktadır ancak sınıf dengesizliği ve küçük nesneleri doğru algılayamama sorunlarına sahiptir.

3.2.Single Shot MultiBox Detector (SSD)

Single Shot MultiBox Detector, YOLO modeli gibi tek aşamalı bir dedektördür. YOLO modelindeki gibi, SSD modelinde de nesne sınıfı puanları ve sınırlayıcı kutu değerleri doğrudan tahmin edilmektedir. SSD temel model olarak VGG ağını kullanmaktadır. Modelde farklı en boy oranlarına ve ölçeklere sahip nesneleri işlemek için çok sayıda öznitelik haritalarından gelen tahminler birleştirilmektedir. Birden fazla çözünürlükte tahminler yapabilmek için VGG ağının sonuna ek evrişimsel katmanlar eklenmiştir. Bu katmanlar sınırlayıcı kutuları ve bunların karşılık gelen nesne puanlarını farklı en boy oranları ve ölçeklerde tahmin etmektedir. Ağda öznitelik katmanı başına bir ölçek kullanılır ve bu ölçek parametresine Denklem 3.2'deki denklem kullanılarak karar verilir.

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m-1}(k-1), \quad k \in [1, m] \quad (3.2)$$

Denklem 3.2'de s_{min} 0,2 , s_{max} 0,9 ve m değeri öznitelik katmanı sayılarını ifade etmektedir. Her bir ölçek için, toplamda altı tane anchor kutusu kullanılır ve bunlar 1,2,3,1/2,1/3 en boy oranlarına sahiptir. SSD modelinin yapısı Şekil 3.2' de verilmiştir.



Şekil 3.2. SSD Mimarisi

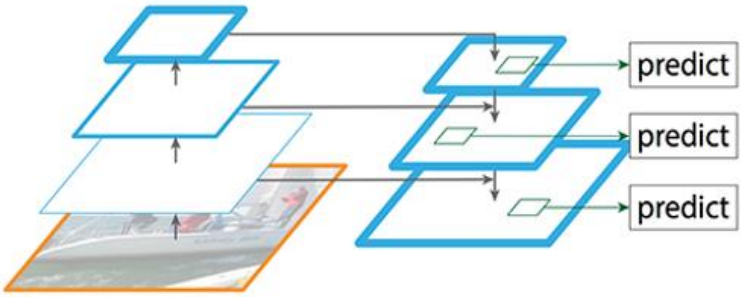
SSD modelinde konum kaybını hesaplamak için yumuşatılmış L1 kaybı, güven kaybını hesaplamak için ise kategorik çapraz entropi kullanılmaktadır. Kayıp fonksiyonu Denklem 3.3 'deki gibidir.

$$L(x, c, l, g) = \frac{1}{N} \left(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\right) \tag{3.3}$$

Denklem 4.3'de *l* terimi tahmin edilen sınırlayıcı kutuyu, g terimi referans kutuyu, c terimi güven değerini, N terimi pozitif eşleşmeleri ve α terimi ise konumlandırma kaybı için ağırlığı temsil etmektedir.

3.3.Focal Loss for Dense Object Detection (RetinaNet)

RetinaNet, ResNet öznitelik çıkarıcısının üstünde Feature Pyramid Networks (FPN) kullanan tek aşamalı bir nesne algılama ağıdır. FPN, anlamsal (semantik) olarak çok ölçekli ve güçlü öznitelik haritaları oluşturan hızlı bir mimaridir. FPN' nin mimarisi Şekil 3.3'de verilmiştir.



Şekil 3.3. Öznitelik Piramit Ağı

Üç en boy oranına (1:2, 1:1, 2:1) ve üç ölçeğe ($2^0, 2^{1/3}$, $2^{2/3}$) sahip toplam dokuz bağlantı kutusu (anchor box) , FPN'nin her piramit seviyesinden elde edilmektedir. Tahmin edilen sınırlayıcı kutular ve sınıflandırma hedefleri sırasıyla tamamen evrişimsel ağlar olan kutu regresyonu ve sınıflandırma alt ağını beslemektedir. Sınıflandırma alt ağı, nesne varlığı olasılığını tahmin eder ve kutu regresyon alt ağı, her bağlantı (anchor) kutusu ve nesne sınıfları için her bir konumda yer doğruluk nesnesine (varsa) olan ofsetleri azaltır. Tek aşamalı nesne dedektörlerinde ön plan- arka plan sınıf dengesizliği sorunu ağların performansını azaltmaktadır. RetinaNet odak kaybı kullanarak sınıf dengesizliği problemini çözmeye çalışmıştır. Odak kaybı iyi sınıflandırılmış veya kolay örneklerle elde edilen kaybı azaltır. Bu nedenle, dedektörü iyi sınıflandırılmış veya kolay örneklerle boğmak yerine, odak kaybı eğitim sırasında zor örneklere daha fazla odaklanır. Odak kaybı fonksiyonu Denklem 3.4'de verilmiştir.

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{3.4}$$

Denklemde γ odaklanma ve α ise dengeleme parametresini ifade etmektedir. Odaklanma parametresi iyi sınıflandırılmış örneklerin ağırlıklarını düşürür ve zor örneklere daha fazla ağırlık verir. Dengeleme parametresi ise sınıftaki örnek sayısının tersini ifade eder. Belirli bir sınıftaki örnek sayısı fazla ise daha az ağırlık verilir. RetinaNet, tek aşamalı dedektörlerin hızından yararlanır ve pozitif-negatif sınıf dengesizliğinin üstesinden gelir.

3.4.You Only Look Once Version 3 (YOLOv3)

YOLOv3 modeli son derece hızlı tek aşamalı bir nesne algılama ağıdır. YOLO (Redmon vd., 2016) ve YOLOv2 (Redmon ve Farhadi, 2017) modellerinden sonra geliştirilmiştir. YOLO girişteki görüntüyü S×S ızgaraya böler. Her ızgara hücresi sabit sayıda sınırlayıcı kutu üretir ve her sınırlayıcı kutu beş parametreden oluşmaktadır (*tx, ty, tw, th, güven puanı*). Modelin çıkış tensör boyutu N × N × (3 . (4 + 1 + C)) kadardır. Burada, N×N ızgara hücre sayısını, 3 rakamı üç ölçekte tespit yapıldığını, dört tane sınırlayıcı kutu değerlerini, 1 nesnenin varlığının tahminini ve C ise sınıf sayısını ifade etmektedir. YOLOv3 modelinde kullanılan kayıp fonksiyonu önceki YOLO versiyonlarında kullanılan ile aynıdır. Kayıp fonksiyonunda, sınıf tahminleri için çok etiketli bir yaklaşım kullanılarak karmaşık veri setlerinde daha iyi modelleme yapmak için ikili çapraz entropi kaybı seçilmiştir. YOLOV3' de kullanılan kayıp fonksiyonu Denklem 3.5'de verilmiştir.

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]$$
$$+\ \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$
$$+\ \sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2$$
$$+\ \sum_{i=0}^{S^2}1_{i}^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2 \qquad (3.5)$$

Kayıp fonksiyonunda ilk terim yanlış hücre merkezi tahminlerini, ikinci terim ise karekök işlemi nedeniyle daha büyük sınırlama kutularına kıyasla cezalandırılan hatalı daha küçük sınırlama kutularının genişliğini ve yüksekliğini cezalandırır. Eğer bir nesne mevcutsa, güven puanı üçüncü terim ile en üst değere çıkartılır. Nesne yoksa bu değer dördüncü terim ile en aza indirilir ve beşinci terim ise sınıflandırma kaybıdır. YOLOv2' nin kullandığı öznitelik çıkarıcı Darknet-19 yapısı yerine, YOLOv3'de daha sağlam ve derin bir mimari olan Darknet-53 kullanılmıştır. Darknet-53 yapısı ResNet (He vd., 2016) yapısına

benzer atlama bağlantıları kullanan 53 evrişimsel kıvrımdan oluşmaktadır. YOLOv3 modeli ağ sonundaki katmanların anlamsal bilgisini ve daha önceki katmanların ayrıntılı özelliklerini daha iyi kullanmak için üç farklı öznitelik haritasında tahmin yapmaktadır. Ayrıca, bu üç farklı öznitelik haritası ile görüntüdeki daha küçük nesneler daha başarılı bir şekilde algılanmaktadır.

3.5.You Only Look Once Version 4 (YOLOv4)

YOLOv4 (Bochkovskiy vd., 2020) en son geliştirilen YOLO modelidir. Bu çalışmada yazarlar tek bir Grafik İşleme Biriminde (GPU) üzerinde eğitilebilen, daha hızlı ve daha doğru sonuçlar veren bir model oluşturmayı amaçlamışlardır. En yüksek doğruluğu elde etmek için Bag-of-Freebies ve Bag-of-Specials methodlarının etkilerini araştırıp, en iyi çalışanları modelde kullanmışlardır.

En genel haliyle dedektör modelleri giriş görüntüsü, omurga (backbone) ağı, boyun (neck) ve baş (head) kısımlarından oluşmaktadır. YOLOv3 modelinde omurga ağı olarak Darknet-53 kullanılırken, YOLOv4 modelinde CSPDarknet53 (Wang vd., 2020) sinir ağı kullanılmıştır. CSPDarknet53 ağı öznitelik haritasını CSPNet stratejisi ile iki kısma ayırır ve ve daha sonra birleştirir. Öznitelik haritasını bölme ve birleştirme işlemleri ile ağdan daha fazla gradyan akışı sağlanır. Modelin boyun kısmında SPP (Spatial Pyramid Pooling) (He vd., 2015) ve PANet (Liu ve Qi, 2018) kullanılmıştır. SPP modülü CSPDarknet53 ağına eklenerek ağın alıcı alanı arttırılmıştır. Omurga ağının farklı katmanlarındaki parametrelerin bir araya getirilmesi için YOLOv3 modelinde FPN kullanılırken, YOLOv4 modelinde PANet tercih edilmiştir. Modelin son kısmı olan baş kısmında ise YOLOv3 kullanılmıştır. YOLOv3 modeli ile karşılaştırıldığında, YOLOv4 modeli doğruluğu %10 ve hızı %12 oranında geliştirmiştir.

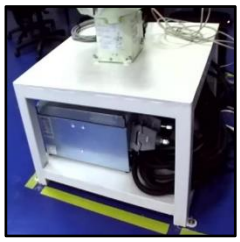
4.BULGULAR VE TARTIŞMA

4.1. ESOGÜ IFARLAB Görsel Veri Kümesi

Fabrika ortamlarındaki işaretleri tespit etmek için, gerçek ortamdan kaydedilen görüntülere ihtiyaç vardır. Bu amaçla Eskişehir Osmangazi Üniversitesi'nde bir akıllı fabrika ortamının simule edildiği bir laboratuvarda fabrika içi güvenlik ve yönlendirme işaretlerinden oluşan görsel bir veri kümesi oluşturulmuştur. Laboratuvar ortamı, fabrikadaki birçok standart tipik nesneyi içermektedir. Otonom taşıyıcı araçların fabrika içerisindeki güvenli hareketi için ortamdaki nesneleri doğru algılayabilmesi kritik öneme sahiptir. Veri kümesi içinde hem araç sürücüsü hem de insan davranışları için bilgilendirici olan nesne ve işaretler bulunmaktadır. 16 adet sınıf içeren veri kümesinin oluşturulduğu fabrika test ortamı Şekil 4.1'de gösterilmektedir.

Şekil 4.1. Çalışmada kullanılan fabrika test ortamı

Veri kümesi 16 sınıftan oluşan fabrika ortamında bulunabilecek nesne ve işaretlerden oluşmaktadır. Sınıflara ait nesne ve işaret görselleri Şekil 4.2'de gösterilmiştir.

			
Sağ Yön	İleri Sağ Yön	Hız Limiti	Tezgah
			
Forklift Uyarı	Giriş Yasaktır	Acil Çıkış	Dur
			
Yaya Yolu	Yaya Yolu Uyarı	Park Alanı	Raf
			
Yangın Tüpü	OTA	Robot Kolu	Robot Tezgahı

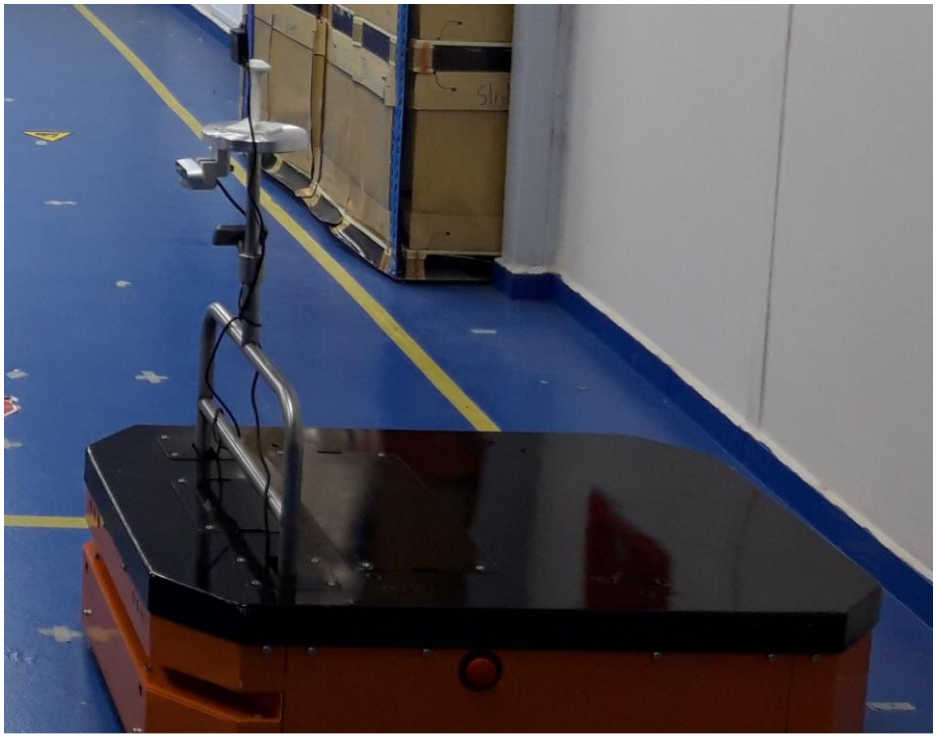
Şekil 4.2. Sınıflara ait nesne ve işaret görselleri

Test ortamından görüntülerin elde edilmesi için ZED Stereo kamera Şekil 4.3'daki OTA'nın üzerine yerleştirilmiş ve ortamda gezdirilerek video kayıtları alınmıştır. Veri çeşitliliği ve aracın farklı ışık koşullarında da sağlıklı algılama yapabilmesi için veriler dört farklı aydınlatma koşulu altında oluşturulmuştur. İlk durumda, laboratuvar ortamı sadece gün ışığı ile aydınlatılmıştır. İkinci senaryoda, ortam sadece floresan ışıkla aydınlatılır. Üçüncü senaryoda, ortam sadece led aydınlatma ile aydınlatılır. Son senaryoda, ortam

floresan ışığı ve led ışığı birlikte kullanarak aydınlatılmıştır. Ayrıca, kameranın farklı çekim açıları ve nesne ve işaretlerin farklı konumları ile video kayıtları alınmıştır. Videolar stereo kamera ile çekildiğinden her bir karede sağ ve sol kameradan gelen iki görüntü oluşmaktadır.Bu görüntüler 2560 × 720 çözünürlükte elde edilmiştir. Daha sonra, video kayıtlarından görüntüler elde edilip, görüntüler sağ ve sol görüntü olarak ikiye bölünmüştür. Tüm görüntüler yeniden boyutlandırılarak her bir görüntü 1040×640 çözünürlükte ayarlanmıştır. Tüm işlemler tamamlandıktan sonra, toplam 1040×640 çözünürlükte 8370 görüntü elde edilmiştir. Bu görüntülerin 7570 tanesi eğitim, 800 tanesi ise test için kullanılmıştır. Her bir görüntü birden fazla işaret ve nesne içermektedir. Eğitim ve test görüntülerinde hangi işaret ve nesneden kaç tane olduğu Çizelge 4.1' de gösterilmiştir. Veri kümesindeki 8370 görüntüden toplamda 25919 nesne elde edilmiştir.

Çizelge 4.1. Veri kümesindeki toplam nesne sayıları

Sınıf İsmi	**Eğitim**	**Test**	**Toplam**
Sağ Yön	2155	218	2373
İleri Sağ Yön	1182	99	1281
Hız Limiti	524	88	612
Tezgah	1667	178	1845
Forklift Uyarı	1305	94	1399
Giriş Yasaktır	827	93	920
Acil Çıkış	1018	55	1073
Dur	2141	286	2427
Yaya Yolu	3138	350	3488
Yaya Yolu Uyarı	1907	168	2075
Park Alanı	326	61	387
Raf	1405	166	1571
Yangın Tüpü	453	100	553
OTA	1180	74	1254
Robot Kolu	2481	188	2669
Robot Tezgahı	1843	149	1992
		Toplam Nesne Sayısı :	**25919**



Şekil 4.3. Çalışmada kullanılan OTA

Çalışmada kullanılan derin öğrenme modellerini hazırlanan veri kümesi ile eğitimini gerçekleştirmek için her bir görüntüye ait etiket dosyalarının hazırlanması gerekmektedir. Bu işlem veri kümesindeki her bir görüntüde yer alan nesnelerin sınıf bilgisini ve bu nesnelerin konumlarını içeren dosyalardır. Toplanan görüntüler manuel olarak bir program aracılığıyla etiketlenmiştir. Her bir görüntüye ait nesne ve işaretlerin sınıf ve konum bilgilerini içeren etiket dosyaları oluşturulmuştur. Bu süreç yaklaşık olarak üç ayda iki kişi tarafından gerçekleştirilmiştir. Etiket dosyaları her derin öğrenme modeline uygun şekilde hazırlanmıştır. YOLOv3 ve YOLOv4 modelleri txt formatında etiket dosyası kullanırken, Faster R-CNN ve SSD modelleri ise XML dosya formatı kullanmaktadır. Son olarak RetinaNet modeli eğitim için CSV etiket dosyası istemektedir. Bu şekilde tüm eğitim ve test görüntülerine ait referans etiket dosyaları hazırlanmış ve eğitime hazır hale getirilmiştir.

4.2. Eğitim Aşaması

Eğitim aşaması için 16 sınıftan oluşan 7570 tane eğitim görüntüsü kullanılmıştır. Modellerin her biri Nvidia Quadro P5000 GPU üzerinde yaklaşık iki gün boyunca

eğitilmiştir. Tüm yöntemlerin eğitimi toplamda yaklaşık dokuz gün sürmüştür. Modellere ait detaylı eğitim bilgileri aşağıdaki gibidir :

Faster R-CNN : Faster R-CNN modeli hazırlanan veri kümesi üzerinde Tensorflow Object Detection API (Anonim, 2020) kaynak kodu kullanılarak eğitilmiştir. Tensorflow Object Detection API bir derin öğrenme kütüphanesi olan Tensorflow üzerine kurulmuş, nesne algılama modellerini oluşturmayı ve eğitmeyi kolaylaştıran bir açık kaynaklı bir yapıdır. Veri kümesi bu yapıda eğitilmek üzere düzenlenmiştir. Faster R-CNN modelinde önceden eğitilmiş model olarak MS-COCO veri kümesi üzerinde eğitilmiş Resnet50 sinir ağını kullanılmıştır. Önceden eğitilmiş ağ kullanılmasının sebebi, bir derin öğrenme modelinin sıfırdan eğitilmek için büyük miktarda eğitim verisi, parametre ve işlem gücü gerektirmesidir. Daha kısa sürede ve kolay bir eğitim için büyük veri kümeleri üzerinde eğitilmiş modeller kullanılmaktadır. Eğitimde optimizasyon algoritması olarak Stochastic Gradient Descent algoritması seçilmiştir. Öğrenme hızı 0,0005 olarak belirlenirken momentum parametresi 0,9 olarak ayarlanmıştır.

SSD : SSD modelinin eğitimi de Faster R-CNN modelinde olduğu gibi Tensorflow Object Detection API kodu kullanılarak gerçekleştirilmiştir. Önceden eğitilmiş model olarak MobileNet v1 (Howard vd., 2017) sinir ağı kullanılmıştır. Optimizasyon algoritması için Faster R-CNN ‘den farklı olarak Adam Optimizasyon kullanılmış olupi öğrenme hızı 0,0001 olarak ayarlanmıştır.

RetinaNet : Faster R-CNN ve SSD modellerinden farklı olarak, RetinaNet modeli bir başka derin öğrenme kütüphanesi olan Keras tabanlı bir kaynak kodu (Anonim, 2020) kullanılarak eğitilmiştir. Faster R-CNN modelinde olduğu gibi, önceden eğitilmiş model olarak MS-COCO veri kümesinde eğitilmiş Resnet-50 sinir ağı kullanılmıştır. Eğitim sırasında öğrenme hızı 0,001 olarak ayarlanmıştır.

YOLOv3 : YOLOv3 modeli Darknet kütüphanesi (Anonim, 2020) kullanılarak eğitilmiştir. Eğitim için Darknet kütüphanesindeki yapılandırma dosyaları eğitim verileri ve etiket dosyalarına göre düzenlenmiştir. Önceden eğitilmiş model olarak MS-COCO üzerinde eğitilmiş YOLOv3 modeli kullanılmıştır. 80 sınıflı MS-COCO veri kümesine göre düzenlenen yapılandırma dosyaları, hazırlanan 16 sınıflı veri kümesine göre tekrar

ayarlanmıştır. Optimizasyon algoritması olarak Stochastic Gradient Descent algoritması 0,9 momentum değeri ile birlikte kullanılmıştır. Öğrenme hızı 0,001 ve ağırlık olarak ayarlanmıştır. Bağlantı kutularının optimum en boy oranlarını belirlemek için, eğitim verisinin oluşturulan sınırlayıcı kutuları kutularına k-means kümeleme uygulanmıştır. Benzer boyutlardaki sınırlayıcı kutular kümelenip, bağlantı kutularının hangi boyutlarda olması gerektiği belirlenmiştir.

YOLOv4 : YOLOv4 modelinin eğitimi de YOLOv3 modelinde kullanılan kütüphane ve kaynak kod ile gerçekleştirilmiştir. YOLOv3 modeline benzer bir şekilde MS-COCO üzerinde eğitilmiş YOLOv4 modeli önceden eğitilmiş model olarak seçilmiştir. Öğrenme hızı 0,1 olarak ayarlanmıştır. YOLOv3 modelinde olduğu gibi bağlantı kutularının optimum en boy oranlarını belirlemek için, eğitim verisinin oluşturulan sınırlayıcı kutuları kutularına k-means kümeleme uygulanmıştır.

Çizelge 4.2.Modellerin mAP değerleri

Test Edilen Model	**mAP**
Faster R-CNN	% 92,66
SSD	% 87,91
RetinaNet	% 89,31
YOLOv3	% 93,72
YOLOv4	% 94,48

4.3. Test Aşaması

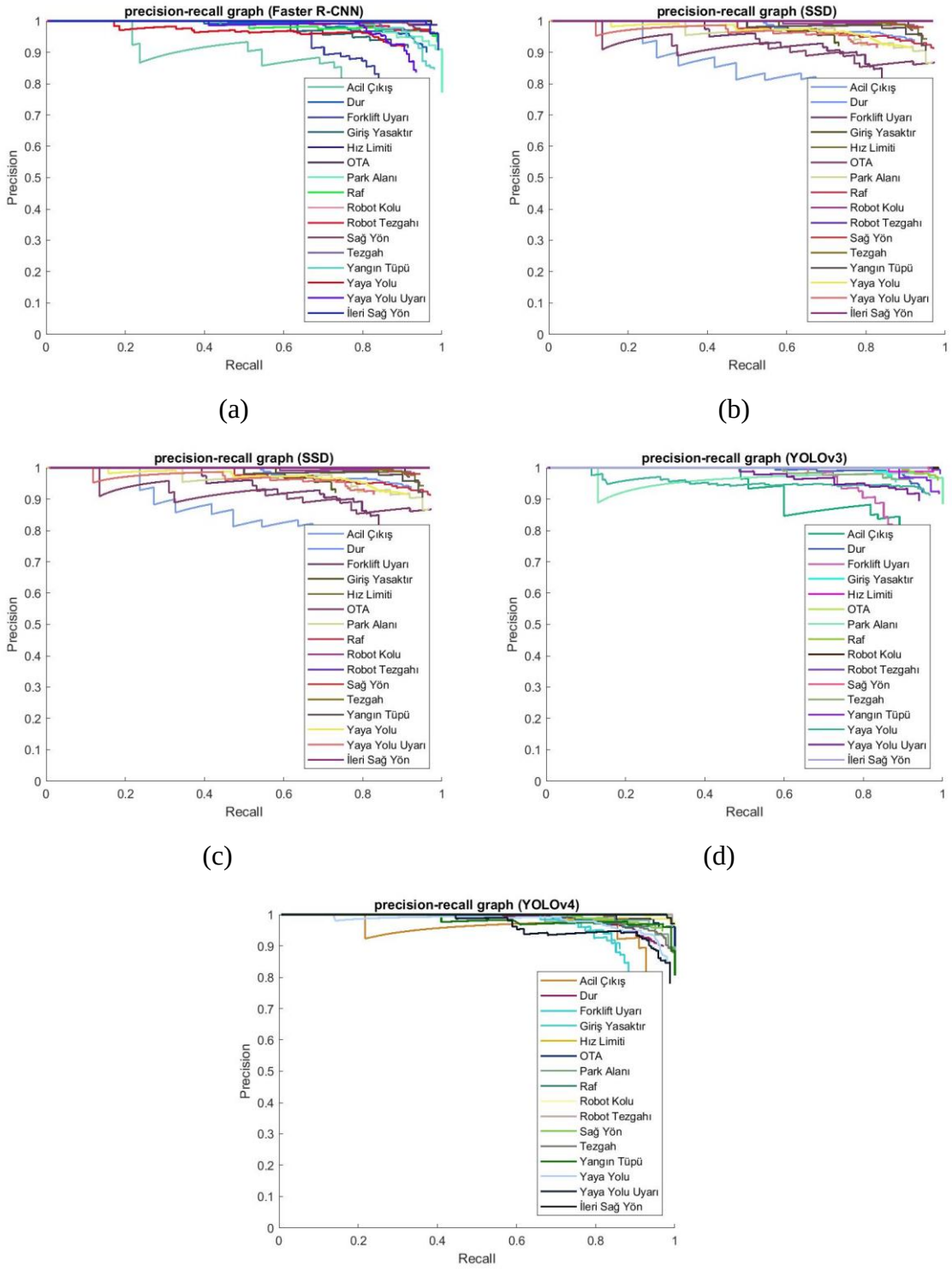
Tüm modellerin eğitimi tamamlandıktan sonra, 800 adet test görüntüsü ile test aşaması gerçekleştirilmiştir. Tüm dedektörler 800 adet test verisi ile test edilmiştir. Test edilen modellere ait mAP değerleri Çizelge 4.2'de verilmiştir. Çizelge 4.2' de görüldüğü gibi veri kümesi üzerinde test edilen dedektörlerden en yüksek doğruluk değerini YOLOv4 modeli elde ederken, en düşük doğruluk değerine sahip model SSD modeli olmuştur. Modeller 16 sınıflı bir veri kümesinde test edilmiştir. Her model bazı nesneleri daha kolay algılarken, bazılarını algılamakta zorluk çekmiştir. Hangi modelin hangi sınıfta daha başarılı

olduğunu anlamak için sınıf bazında başarı tablosu hazırlanmıştır. Çizelge 4.3'de test edilen beş modelin 16 sınıf bazında başarı değeri verilmiştir.

Çizelge 4.3. Sınıflara ait doğruluk değerleri

Sınıflar	**AP(%)**				
	Faster R-CNN	**YOLOv3**	**SSD**	**RetinaNet**	**YOLOv4**
Acil Çıkış	75.03	86.71	68.55	85.44	**90.68**
Dur	94.64	94.56	90.61	82.46	**95.31**
Forklift Uyarı	81.53	85.58	82.01	78.77	**88.65**
Giriş Yasaktır	**88.92**	86.99	71.49	88.78	84.09
Hız Limiti	**97.65**	96.45	93.85	96.59	97.38
İleri Sağ Yön	97.89	94.95	95.96	95.82	**98.95**
OTA	95.95	95.95	87.79	94.58	**98.31**
Park Alanı	**97.42**	95.06	92.02	81.73	97.39
Raf	96.93	**98.04**	94.31	95.81	92.74
Robot Kolu	95.71	98.40	92.26	98.36	**98.85**
Robot Tezgahı	96.11	**97.97**	91.24	94.12	92.62
Sağ Yön	**97.46**	96.33	93.91	90.71	94.48
Tezgah	91.91	91.37	90.92	93.17	**94.76**
Yangın Tüpü	96.01	**97.59**	93.10	94.40	97.41
Yaya Yolu	87.94	91.99	88.89	74.59	**94.76**
Yaya Yolu Uyarı	91.53	91.52	79.69	83.63	**95.30**

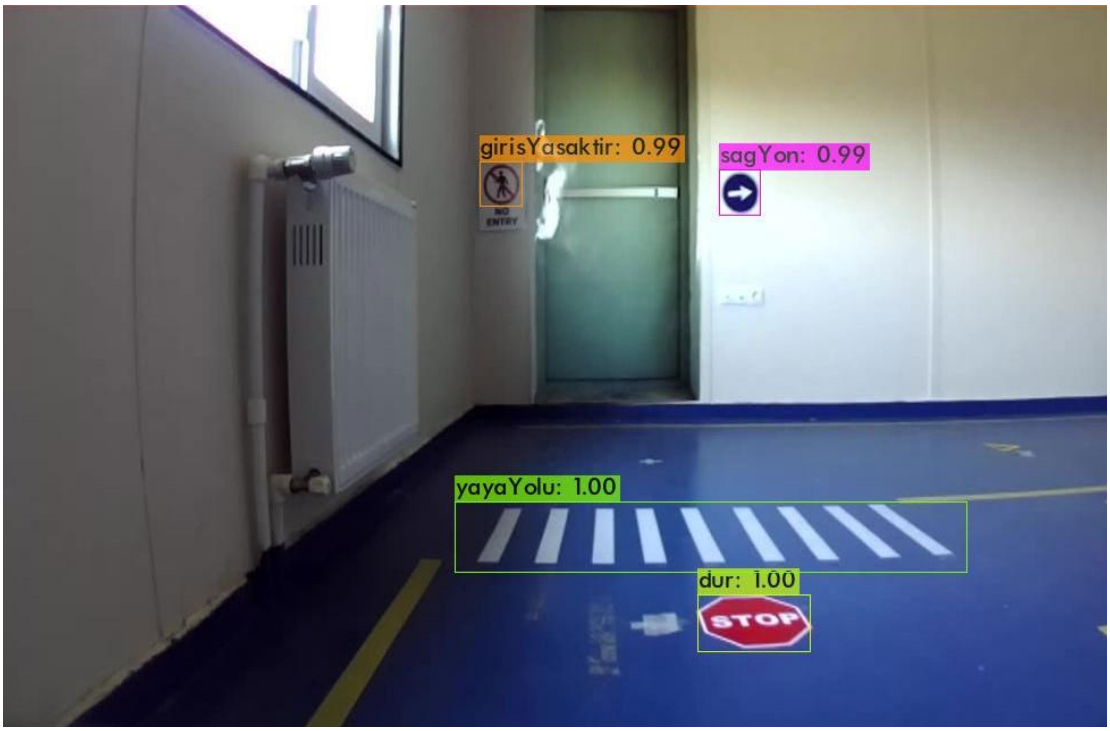
Çizelge 4.3 detaylı olarak incelendiğinde, YOLOv4 modeli dokuz tane sınıfta, YOLOv3 modeli üç tane sınıfta ve Faster R-CNN modeli dört tane sınıfta en yüksek doğruluk değerini elde etmiştir. En yüksek doğruluk değerleri koyu renk ile belirtilmiştir. Doğruluk değerlerine bakıldığında Forklift Uyarı, Giriş Yasaktır, Acil Çıkış ve Yaya Yolu sınıfları en düşük doğruluk değerlerine sahiptir.



Şekil 4.4. Modellere ait precision-recall eğrileri, (a) Faster R-CNN, (b) SSD, (c) RetinaNet, (d) YOLOv3, (e) YOLOv4

Bu işaret ve nesneler modeller tarafından en zor algılanan sınıflar olmuşlardır. En kolay tespit edilen sınıflar ise Robot Kolu, Hız Limiti, OTA ve İleri Sağ Yön sınıfları olmuştur. Diğer sınıflara göre daha yüksek doğruluk değerleri elde etmişlerdir. Bazı sınıfların diğerlerine göre daha zor tespit edilmesinin sebebi, veri kümesindeki görüntülerde bazı nesnelerin diğerlerine göre daha küçük boyutta olması veya başka bir nesnenin arkasında kalması sebep olmuştur.

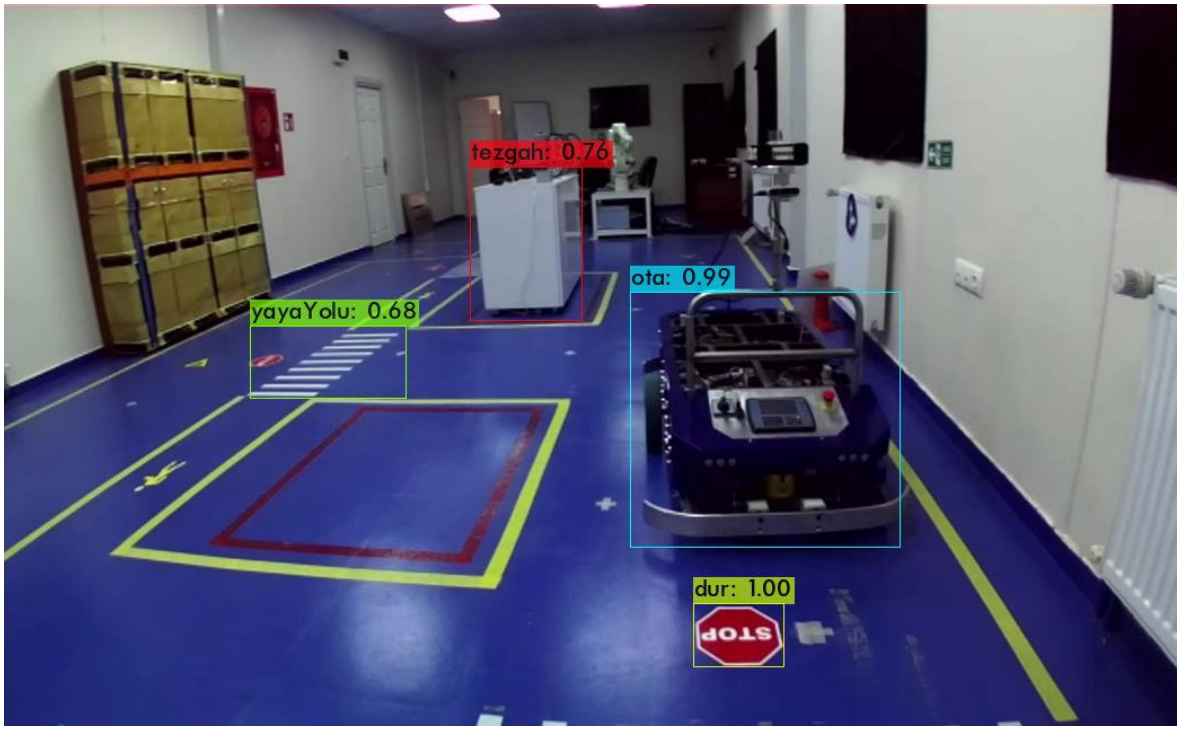
Sınıflandırıcıların başarı kriterleri olarak doğruluk değerinden başka precision-recall eğrisi de kullanılmaktadır. Bu eğri sınıflandırıcının precision ve recall değerleri arasındaki ilişkiyi göstermektedir. Şekil 4.4'de beş modelin her sınıfa ait ait precision-recall eğrileri verilmiştir.



Şekil 4.5. Başarılı bir algılama örneği

Oluşturulan görsel nesne algılama sisteminin daha iyi anlaşılması için, bazı test verilerinin algılama çıktıları örnek olarak verilmiştir. Test edilen modeller arasında en iyi performansı YOLOv4 modeli elde ettiği için örneklerde bu modelin çıktıları kullanılmıştır. Şekil 4.5'de iyi tespit edilmiş işaret ve nesneler örneklendirilmiştir. Şekil 4.5'deki örnekte görüldüğü gibi dört adet güvenlik işareti de başarılı bir şekilde tespit edilmiştir. Şekil 4.6' da ise tespit edilemeyen bazı işaret ve nesnelerin olduğu örnekler gösterilmiştir. Şekil 4.6'ya

bakıldığında Dur ve OTA sınıfları gibi görüntüde en önde yer alan işaret ve nesneler yüksek doğruluk ile tespit edilirken, Yaya Yolu ve Tezgah sınıfları daha uzakta oldukları için daha düşük doğruluk ile tespit edilmiştir. Yaya Yolu Uyarı ve İleri Sağ Yön işaretleri ise diğer nesnelere göre daha küçük olduğu için tespit edilememiştir.



Şekil 4.6. Tespit edilemeyen işaret ve nesnelerin olduğu bir örnek

6. SONUÇ VE ÖNERİLER

Otonom taşıyıcı araçlar Endüstri 4.0 ile birlikte akıllı fabrikaların önembir bir parçası haline gelmiştir. Bu araçların fabrika ortamı içerisinde kendi kendilerine güvenli bir şekilde hareket edebilmeleri için ortamdaki işaret ve nesneleri başarılı bir biçimde tespit edebilmeleri gerekmektedir. Bu çalışmada, akıllı fabrika benzetimi için oluşturulan bir laboratuvar ortamından fabrika ortamında karşılaşılan 16 farklı güvenlik işareti ve nesnelerden oluşan bir veri kümesi toplanmıştır. Toplanan veri kümesi ile güncel derin öğrenme modellerinden 5 tanesi eğitilmiştir. Bu modeller Faster R-CNN, SSD, RetinaNet, YOLOv3 ve YOLOv4 modelleridir. Eğitilen ağ modelleri oluşturulan test veri kümesinde doğruluk değeri bakımından karşılaştırılmıştır. En iyi performansı YOLOv4 modeli elde ederken, sırasıyla YOLOv3 ve Faster R-CNN modelleri ise en iyi ikinci ve üçüncü doğruluğu elde etmiştir. SSD ve RetinaNet modelleri diğer modellere göre daha düşük bir başarı sağlamışlardır.

KAYNAKLAR DİZİNİ

Abdi, L., Meddeb, A., 2017, Deep learning traffic sign detection, recognition and augmentation, Proceedings of the ACM Symposium on Applied Computing, Part F1280, 131–136.

Anonim, 2020, Darknet, Yolo v4, v3 and v2 for Windows and Linux, https://github.com/AlexeyAB/darknet, erişim tarihi : 03.12.2020.

Anonim, 2020, Keras RetinaNet , https://github.com/fizyr/keras-retinanet , erişim tarihi : 03.12.2020.

Anonim, 2020, TensorFlow Object Detection API, https://github.com/tensorflow/models/tree/master/research/object_detection, erişim tarihi : 03.12.2020.

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y. M. ,2020, YOLOv4: Optimal speed and accuracy of object detection, http://arxiv.org/abs/2004.10934

Brownlee J., 2019, Understand the Impact of Learning Rate on Neural Network Performance, https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/, erişim tarihi : 12.09.2020.

Çarkacı, N., 2018, Derin Öğrenme Uygulamalarında En Sık Kullanılan Hiperparametreler, https://medium.com/deeplearning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametrele r-ece8e9125c4, erişim tarihi : 12.09.2020.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014, Rich feature hierarchies for accurate object detection and semantic segmentation, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 580–587.

Girshick, R., 2015, Fast R-CNN, Proceedings of the IEEE International Conference on Computer Vision, 1440–1448.

Gode, C. S., Khobragade, A. S., 2016, Object detection using color clue and shape feature, Proceedings of the 2016 IEEE International Conference on Wireless Communications, Signal Processing and Networking, 464–468.

Gupta, S., Singh, Y. J., 2014, Object detection using shape features, 2014 IEEE International Conference on Computational Intelligence and Computing Research, 111–122.

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, 770–778.

KAYNAKLAR DİZİNİ (devam)

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.

Hijazi, S., Kumar, R., Rowen, C., 2015, Using convolutional neural networks for image recognition, Cadence Design Systems Inc.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., vd., 2017, MobileNets: Efficient convolutional neural networks for mobile vision applications, *arXiv*.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. vd., 2016, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, 1–13, http://arxiv.org/abs/1602.07360.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2017, ImageNet classification with deep convolutional neural networks, Communications of the ACM, 60(6), 84–90.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E. vd., 1989, Backpropagation applied to digit recognition, Neural computation, 1(4), 541–551, https://www.ics.uci.edu/~welling/teaching/273ASpring09/lecun-89e.pdf.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86(11), 2278–2323.

Lee, H. S., Kim, K., 2018, Simultaneous traffic sign detection and boundary estimation using convolutional neural network, IEEE Transactions on Intelligent Transportation Systems, 19(5), 1652–1663.

Lin, T. Y., Goyal, P., Girshick, R., He, K., Dollar, P., 2017, Focal loss for dense object detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2), 318–327.

Liu, S., Qi, L., 2018, Path aggregation network for instance segmentation, IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., vd., 2016, SSD: Single shot multibox detector, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9905 LNCS, 21–37.

O'Shea, K., Nash, R., 2015, An introduction to convolutional neural networks, 1–11, http://arxiv.org/abs/1511.08458.

KAYNAKLAR DİZİNİ (devam)

Peng, X., 2015, Combine color and shape in real-time detection of texture-less objects, Computer Vision and Image Understanding, 135, 31–48.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016, You only look once: Unified, real-time object detection, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 779–788.

Redmon, J., Farhadi, A., 2017, YOLO9000: Better, faster, stronger, Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 6517–6525.

Redmon, J., Farhadi, A., 2018, YOLO v3. Tech report, 1–6, https://pjreddie.com/media/files/papers/YOLOv3.pdf

Ren, S., He, K., Girshick, R., Sun, J., 2017, Faster R-CNN: Towards real-time object detection with region proposal networks., IEEE Transactions on Pattern Analysis and Machine Intelligence, *39*(6), 1137–1149.

Saha, S., 2018, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, erişim tarihi: 20.09.2020.

Simonyan, K., Zisserman, A., 2015, Very deep convolutional networks for large-scale image recognition, 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 1–14.

Supreeth, H. S. G., Patil, C. M., 2016, An approach towards efficient detection and recognition of traffic signs in videos using neural networks, Proceedings of the 2016 IEEE International Conference on Wireless Communications, Signal Processing and Networking, 456–459, https://doi.org/10.1109/WiSPNET.2016.7566175.

Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., vd., 2020, CSPNet: A new backbone that can enhance learning capability of CNN, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 1571–1580.

Wang, G., Ren, G., Quan, T., 2013a, A traffic sign detection method with high accuracy and efficiency, Iccsee, 1426–1429.

Wang, G., Ren, G., Wu, Z., Zhao, Y., Jiang, L., 2013b, A robust, coarse-to-fine traffic sign detection method, Proceedings of the International Joint Conference on Neural Networks, 1–5.

KAYNAKLAR DİZİNİ (devam)

Wang, G., Ren, G., Wu, Z., Zhao, Y., Jiang, L., 2014, A fast and robust ellipse-detection method based on sorted merging, The Scientific World Journal.

Zhang, J., Huang, M., Jin, X., Li, X., 2017, A real-time Chinese traffic sign detection algorithm based on modified YOLOv2, Algorithms, 10(4), 1–13.

Zuo, Z., Yu, K., Zhou, Q., Wang, X., Li, T., 2017, Traffic signs detection based on Faster R-CNN, Proceedings - IEEE 37th International Conference on Distributed Computing Systems Workshops, 286–288.