T.C.
SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DERİN EVRİŞİMSEL SİNİR AĞLARI KULLANILARAK ARAÇ, İNSAN VE TRAFİK İŞARETLERİNİN TANINMASI

YÜKSEK LİSANS TEZİ

Gülyeter ÖZTÜRK

Enstitü Anabilim Dalı : MEKATRONİK MÜHENDİSLİĞİ

Tez Danışmanı : Prof. Dr. Raşit KÖKER

Ocak 2020

T.C.
SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DERİN EVRİŞİMSEL SİNİR AĞLARI KULLANILARAK ARAÇ, İNSAN VE TRAFİK İŞARETLERİNİN TANINMASI

YÜKSEK LİSANS TEZİ

Gülyeter ÖZTÜRK

Enstitü Anabilim Dalı : MEKATRONİK MÜHENDİSLİĞİ

Bu tez 03/01/2020 tarihinde aşağıdaki jüri tarafından oybirliği ile kabul edilmiştir.

JÜRİ	BAŞARI DURUMU
Jüri Başkanı: Prof. Dr. Raşit KÖKER	BAŞARILI
Üye: Prof. Dr. Durmuş KARAYEL	BAŞARILI
Üye: Doç. Dr. Devrim AKGÜN	BAŞARILI

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Gülyeter ÖZTÜRK

03/01/2020

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Gülyeter ÖZTÜRK
03/01/2020

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, beni yönlendiren değerli danışman hocam Prof. Dr. Raşit KÖKER'e teşekkürlerimi sunarım.

Çalışmam süresince her türlü desteklerini esirgemeyip yardımcı olan değerli hocalarıma, aileme ve arkadaşlarıma sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

KISALTMALAR LİSTESİ

ADAS	: Gelişmiş Sürücü Destek Sistemleri
CNN	: Evrişimsel Sinir Ağları
COCO	: Ortak Nesne İçeriği
Faster R-CNN	: Daha Hızlı Bölgesel Evrişimsel Sinir Ağları
GPS	: Küresel Konumlandırma Sistemi
GPU	: Grafik İşleme Birimi
HOG	: Yönlü Gradyan Histogramı
ILSVRC	: ImageNet Büyük Ölçekli Görüntü Tanıma Yarışması
MLP	: Çok Katmanlı Algılayıcı
RGB	: Kırmızı, Yeşil, Mavi
ROI	: İlgi Bölgesi
SSD	: Tek Çekim Çok Kutuplu Algılayıcı
SVM	: Destek Vektör Makineleri
TSRS	: Trafik İşareti Tanıma Sistemleri
YOLO	: Sadece Bir Kez Bak
YSA	: Yapay Sinir Ağları

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

DERİN EVRİŞİMSEL SİNİR AĞLARI KULLANILARAK ARAÇ, İNSAN VE TRAFİK İŞARETLERİNİN TANINMASI

ÖZET

Günümüzde robotik dünyasının ve otomasyonun bugünü ve geleceği olan otonom arabalar ile gelişmiş sürücü destek sistemleri (ADAS), sürüş ortamı bilgisi yoluyla sürücülere fayda sağlayabilecek en temsili teknolojidir. Özellikle bilgisayar teknolojilerindeki gelişmeler ve sektörde yer edinme ile giderek daha fazla araçta standart hale gelen ileri çarpışma uyarı sistemi, gece görüş sistemi, şeritten ayrılma uyarı sistemi ve trafik işareti izleme sistemi seçenekleriyle ADAS; yoldaki nesneleri çeşitli koşullarda tanır ve çevresindeki durumu sürücülere bildirir. Öte yandan otonom araba, bütün sistemlerden gelen bilgileri bütünleştirir ve sürüşü kendi başına kontrol eder. Bu otonom arabaların temel unsurlarından bazıları; çevreyi, engelleri, yayaları, trafik işaretlerini ve diğer araçları algılamaktır.

Geleneksel görüntü işleme teknikleri kullanılarak yapılan birçok etkili yaya tanıma, trafik işareti tanıma ve şerit tanıma yöntemleri uzun zamandır kullanılmaktadır. Bu tez çalışması kapsamında; görüntü işleme, konuşma tanıma, doğal dil işleme, sinyal işleme gibi makine öğrenmesi konularındaki problemlerin çözümünde son yıllarda büyük doğruluk oranı ve hız ile kendinden çokça söz ettiren derin öğrenme kullanılarak otonom arabanın bir parçası olan çevredeki nesneleri ve şeritleri otomatik algılama uygulaması yapılmıştır. Araçtan çekilen video görüntülerinden şeritler ve nesneler tespit edilmektedir. Geliştirilen yöntemde çeşitli ortamlarda, farklı görüş açılarında ve boyutlarda olan işaretler ve nesneler tanınabilmektedir.

Bu tez çalışmasında görüntü işleme yöntemleriyle şerit algılama gerçekleştirilmiş ve insan, araba, bisiklet ile 7 trafik işaretinden oluşan 10 nesneye ait 517 görüntü, derin öğrenmenin temel mimarisi kabul edilen derin evrişimsel sinir ağları modellerinden SSD Inception V2, Faster R-CNN Inception V2, Faster R-CNN Resnet 50 ve Faster R-CNN Resnet 101 kullanılarak nesne algılama analizi gerçekleştirilmiştir. COCO veri seti üzerinden önceden eğitilmiş olan bu modeller transfer öğrenimi yöntemi ile bir kısmı GRAZ-01 ve GRAZ-02 veri setlerinden elde edilen bir kısmı da cep telefonu kamerası kullanılarak elde edilip oluşturulan yeni veri seti üzerinde tekrardan eğitilerek değerlendirilmiş ve karşılaştırma yapılmıştır. Performans analizleri sonucunda, Faster R-CNN Resnet 101 modelinin hem görüntü hem de video üzerinde nesne algılama çalışmalarında %85.1 doğruluk oranıyla başarılı olduğu gözlemlenmiştir.

Anahtar kelimeler: Derin Öğrenme, Evrişimsel Sinir Ağları, Nesne Algılama, Şerit Algılama, Transfer Öğrenimi.

RECOGNITION OF VEHICLE, HUMAN AND TRAFFIC SIGNS USING DEEP CONVOLUTIONAL NEURAL NETWORKS

SUMMARY

Nowadays, with the development of autonomous cars, Advanced Driver Assistance Systems (ADAS) are the most commun used systems for now and future in robotic world. In the same time, ADAS are the most representative technology that can benefit the driver through existences of the driving environment. ADAS, which have forward collusion warning system, night vision system, lane departure warning system and traffic sign tracking system options and which is becoming more and more standard in the vehicles with the developments in computer technologies and gaining place in the sector, recognizes objects on the road under various conditions and informs drivers about surrounding situation. On the other hand, the autonomous vehicle integrates information from all systems and controls driving on its own. Some of the basic features of these autonomous cars; to detect the environment, obstacles, pedestrians, traffic signs and other vehicles. Many effective pedestrian recognition, traffic sign recognition or lane recognition methods using traditional image processing techniques have long been used.

With the scope of this thesis, automatic detection system of surrounding objects and lanes, which are part of the autonomous car, was developed as using deep learning which has been mentioned with great accuracy and speed in recent years to solve machine learning problems such as image processing, natural language processing, signal processing and speech recognition. With this system, strips and objects are detected from the video image taken from the vehicle. In the developed method, signs and objects in various environments, different viewing angles and dimensions can be recognized.

In this thesis, 517 images of 10 objects consisting of people, cars, bicycles and 7 traffic signs have been recognized by using image processing methods and by using SSD Inception V2, Faster R-CNN Inception V2, Faster R-CNN Resnet 50 and Faster R-CNN Resnet 101 models which are known as basis of deep learning, object detection analysis was performed. These models, which were pre-trained on COCO data set, were obtained by using transfer learning method, some of them were obtained from GRAZ-01 and GRAZ-02 data sets and some of them were obtained by using mobile phone camera and re-trained on the new data set and evaluated and compared. As a result of speed and accuracy performance analyzes, Faster R-CNN Resnet 101 model was found to be successful in object detection on both photographs and video with 85.1% accuracy.

Keywords: Deep Learning, Convolutional Neural Networks, Object Detection, Lane Detection, Transfer Learning.

BÖLÜM 1. GİRİŞ

Robotik araba veya akıllı araba olarak da bilinen otonom araba, sürücünün yardımı olmadan veya herhangi bir insan müdahalesi olmadan dış çevreyi tek başına algılayabilen ve gezinebilen bir araçtır. Otonom araba üzerine yapılan çalışmalar 1920'li yıllarda ticari olarak piyasaya çıktığı zamanlara dayanmasına rağmen, arabanın ilk çalışma modeli 1980'lerde ortaya çıkmıştır. Otonom araba üzerine, 1984 yılında Carnegie Mellon Üniversitesi NAVLAB ve ALV [1] projelerini, 1987'de Bundeswehr Üniversitesi Munich ve Mercedes Benz Eureka Prometheus adlı projesini gerçekleştirmiştir [2].

Otonom arabalar daha güvenli bir sürüş sağlamak için gelişmiş sürücü destek sistemlerini (ADAS) kullanır. Bilgisayar teknolojisindeki gelişmeler ile giderek daha fazla araçta standart hale gelen ADAS; ileri çarpışma uyarı sistemi, gece görüş sistemi, şeritten ayrılma uyarı sistemi ve trafik işareti izleme sistemi gibi seçeneklerle çevresindeki durumu sürücüye bildirir. Şeritten ayrılma uyarı sistemleri 2000'li yıllarda Nissan Motors tarafından piyasaya sürülmesinden sonra, birçok önde gelen otomobil üreticisi de bu sistemi sunmaya başlamıştır. Şerit izleme sisteminden alınan verilerle hem sesli alarm sistemleri hem de koltuk titreşim sistemleri sürücüyü uyarmakta ve şerit algılama sistemleri önleyici bir sistem olarak kullanılmaktadır [3].

Otonom arabaların temel unsurlarından bazıları; çevreyi, engelleri, yayaları ve diğer araçları algılamak için LIDAR'lardan, RADAR'lardan, GPS'den ve bilgisayarlı görme uygulamalarına kadar değişen algılama elemanlarını içerir. Günümüzde bu araçlar, Google, Tesla ve benzeri üreticilerin gelişmeleriyle hayatımızda önemli bir rol almaya başlamış ve daha etkili yol kullanımı ve daha güvenli sürüş ile birçok önleyici yöntem ve trafik kazalarının önlenmesine öncülük etmiştir. Bu amaçla otonom araba, ADAS gibi altyapı sistemleri ile koordineli olarak çalışıp, sürücüye araçtaki bir ekranda algıladığı şeritleri, yayaları, arabaları ve trafik işaretlerini gösterir.

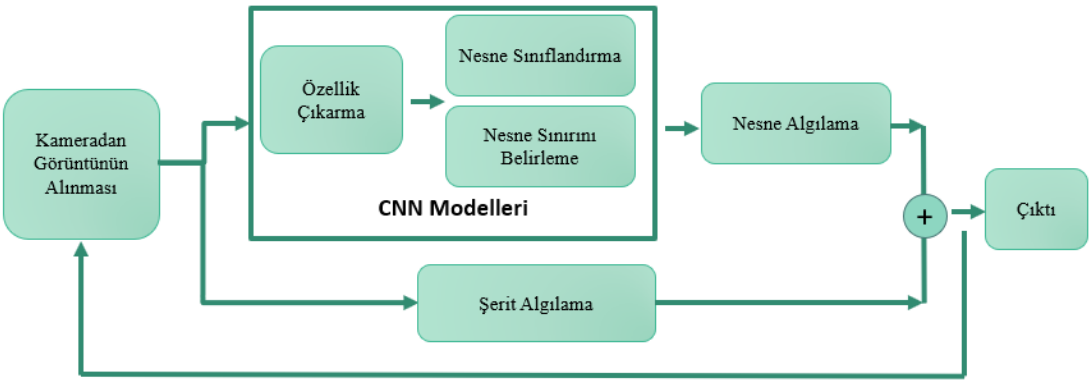
Otonom arabanın trafik kurallarına uymasını sağlamak için uyarı işaretlerini, yön veya gösterge işaretlerini, hız sınırı işaretlerini vb. trafik işaretlerini belirli bir doğruluk derecesine kadar tespit edilmesi ve tanımlanması gerekir. Trafik işaretlerinin sürücü tarafından zamanında ve doğru bir şekilde tespit edilmesi güvenli bir yolculuk için oldukça önemlidir. Trafik işareti tanıma; algılama ve sınıflandırma olarak iki aşamada yapılmaktadır. Algılama aşamasında görüntü üzerinde birçok analiz gerçekleştirilerek trafik işaretlerine dair özellikler çıkarılır. Çıkarılan özellikler sınıflandırma aşamasında analiz edilerek sınıflandırma gerçekleştirilir. Trafik işaretleri kırmızı, mavi, beyaz, sarı, siyah ve yeşil gibi renklerden ve üçgen, dikdörtgen, daire ve çokgen gibi basit şekillerden oluşur. İşaretlerin tasarımı insan sürücü için ayırt edici olsa da, uygulamalar hayata geçirildiğinde gölge, hava durumu, işaretlerin eskimesi ve mesafe sorun oluşturur. Özellik çıkarmada geleneksel HOG, Gabor ve SIFT gibi yöntemler kullanılır.

Nesne algılama, gerçek dünyadaki birçok nesneyi görüntülerden veya videodan bulma işlemidir. Görüntü içindeki birden fazla nesnenin tanınmasını, konumunun belirlenmesini sağlayan nesne algılama; ileri robotikler, savunma sistemleri, gözetim sistemleri, uzay araştırmaları, yüz tanıma ve daha birçok alanda kullanılabilir. Geleneksel makine öğrenimi ve bilgisayarlı görme modelleri, nesne sınıflandırma ve nesne algılama sürecinde yaygın bir rol almaktadırlar. Büyük doğruluk oranı ve hız ile son yıllarda çoğu çalışmalarda kullanılan derin öğrenme de nesne algılama sürecinde önemli bir rol almaktadır [4].

Derin öğrenme, beynin yapısal ve işlevsel özelliklerinden esinlenerek tasarlanan, çok katmanlı ağ yapıları olan yapay sinir ağları üzerinde çalışan algoritmalar ve modeller kümesi ile ilgili bir makine öğrenmesi alt alanıdır. Derin öğrenme konuşma tanıma, ses analizi, görüntü analizi, doğal dil işleme (NLP), robot navigasyon sistemleri ve otonom arabalar gibi birçok alanda kullanılmaktadır. 1980'lerde çok katmanlı algılayıcıların kullanılmaya başlanmasıyla derin öğrenmeye ilk adımlar atılmıştır. 2012 yılında Krizhevsky ve arkadaşlarının nesne tanıma alanında büyük bir yarışma olan ImageNet Büyük Ölçekli Görüntü Tanıma Yarışması'nı (ILSVRC) derin öğrenmenin temel mimarisi kabul edilen Evrişimsel Sinir Ağı'nı kullanarak kazanmaları ile derin öğrenme bilim dünyasında büyük bir etkiyi oluşturmuştur. Makine öğreniminde, insan programcıların oluşturduğu algoritmalarla verilerden özellikler çıkarma gerçekleştirilirken yani özellik çıkarma mühendisliği yapılırken, derin öğrenmede özellik

çıkarma konvolüsyon katmanlarında gerçekleşmektedir. Ham veriler derin öğrenme modeline verilir ve konvolüsyon katmanlarında özellik çıkarma işlemleri gerçekleştirilir. Elde edilen özellikler sınıflandırma katmanına verilerek sonuç elde edilir, böylece model kendi kendine öğrenmeyi sağlamış olur. Derin öğrenmenin sağladığı bu yararlı işlevin yanında çok miktarda veri gerektirmesi bir dezavantaj olarak görülmektedir. Kaliteli ve çok miktarda verinin toplanması çaba ve zaman gerektirir. Makine öğrenmesi ve benzeri çalışmaları yapmak amacıyla oluşturulan halka açık veri tabanları, yüzlerce nesne sınıfından oluşmakta ve milyonlarca resim içermektedir. Farklı görevleri gerçekleştirmek amacıyla bu veriler kullanılarak farklı derin öğrenme modelleri elde edilmiştir.

Bu tez çalışması kapsamında, otonom arabaların bir parçası olan şerit algılama ve nesne algılama çalışması araba içine yerleştirilen bir kameradan elde edilen görüntüler üzerinde gerçekleştirilmiştir. Bu kapsamda oluşturulan çalışmanın şeması Şekil 1.1 'de gösterilmiştir.



Şekil 1.1: Nesne ve şerit algılama şeması

Mühendislik ve bilgisayar bilimleri disiplinlerinde temel araştırma alanı olan görüntü işleme; gelişmiş bir görüntü elde etme veya ondan bazı yararlı bilgiler elde etmek için görüntü üzerinde bazı işlemleri gerçekleştirme yöntemidir. Kameradan alınan görüntülerden şerit tespitini sağlamada görüntü işleme yöntemleri kullanılmıştır. Nesne algılama adı altında insan, araba, bisiklet ile dur, yaya geçidi, yol ver, döner kavşak, kavisli yol, 20 hız sınırı ve 30 hız sınırı trafik işaretleri algılanmaya çalışılmıştır. Veri seti oluşturulurken GRAZ-01 ve GRAZ-02 veri setlerinden yararlanılmıştır ve ülkelere göre trafik işaretleri renk veya şekil olarak değiştiğinden 7 trafik işaretine dair resimlerin tümü cep telefonu kullanılarak elde edilmiştir. Bir ağı sıfırdan eğitmek için çok sayıda verinin var olması ve iyi bir donanıma sahip bilgisayar kullanılması gerekmektedir. Transfer

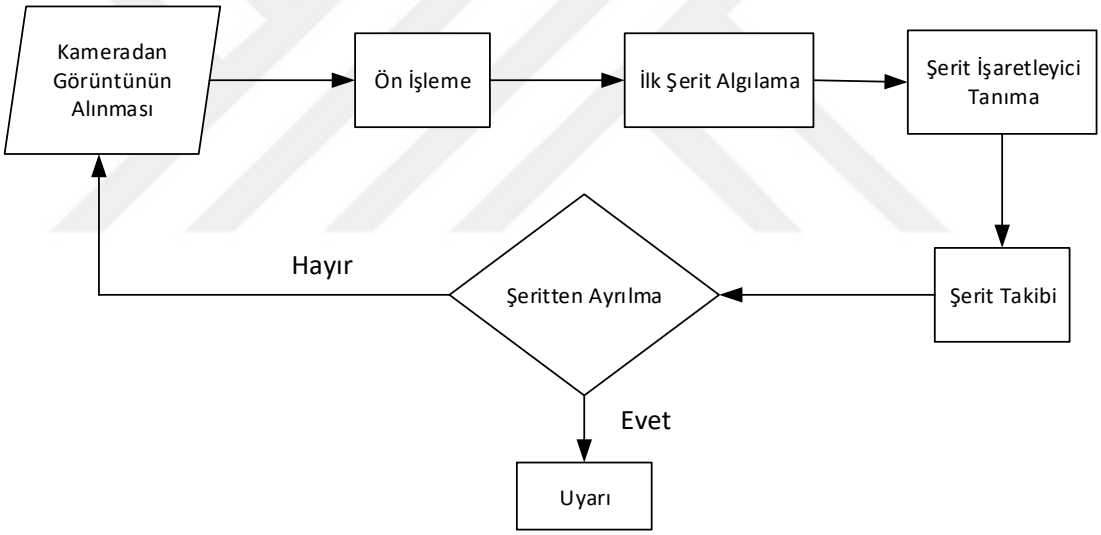
öğrenimi, yeni modelleri eğitmek için önceden eğitilmiş modellerden bilgileri (özellikleri, ağırlıkları vb.) kullanabilmeyi ve yeni görev için daha az veri ile eğitimi gerçekleştirip iyi performans elde edebilmeyi sağlar. Bu çalışmada 10 nesne sınıfı için elde edilen görüntüler az olduğundan transfer öğrenimi yöntemi ile önceden eğitilmiş 4 model kullanılmıştır. Google tarafından daha önce COCO veri seti kullanılarak derin öğrenme kütüphanesi olan TensorFlow üzerinde eğitilen bu modellerde ince ayarlar yapılarak nesne algılamada iyi performans elde edilmeye çalışılmıştır. TensorFlow üzerine inşa edilen TensorFlow Nesne Algılama API, nesne algılama modellerini oluşturmayı, eğitmeyi ve eğitilen modelleri kullanmayı kolaylaştıran açık kaynaklı bir yazılımdır. Bu API aracılığıyla insanları, arabaları, bisikletleri ve trafik işaretlerini algılamak için derin öğrenme temelli modeller eğitilmiş, performansları değerlendirilmiştir.

Tezin ikinci bölümünde şerit algılama, trafik işareti algılama, nesne algılama ve CNN modelleri genel olarak açıklanmış ve bu konularla ilgili çalışmalardan bahsedilmiştir. Üçüncü bölümde temel görüntü işleme işlevleri ile şerit algılama çalışması yapılmıştır. Nesne tespitinde son teknoloji olan SSD Inception V2, Faster R-CNN Inception V2, Faster R-CNN Resnet 50 ve Faster R-CNN Resnet 101 mimarilerinin doğruluk performans analizleri gerçekleştirilmiştir. Dördüncü bölümde nesne algılama ve şerit algılama sonuçlarının değerlendirilmesi yapılmış ve gelecek çalışmalar için önerilerde bulunulmuştur.

BÖLÜM 2. LİTERATÜR ARAŞTIRMASI

2.1. Şerit Algılama

ADAS, sürüş esnasında şeritleri tespit ederek sürücü şeritlerini kazara değiştirme veya şeridi kaçırma gibi durumları önleyecek sistemlerle sürücü güvenliğini sağlamakla birlikte trafik kazalarını engellemektedir. Şerit algılama sistemleri kamera, lazer, LIDAR, IMU sensörler ve GPS gibi farklı teknolojileri kullanır [5]. Şekil 2.1'de tipik bir şerit algılama sistemi akışı gösterilmektedir.



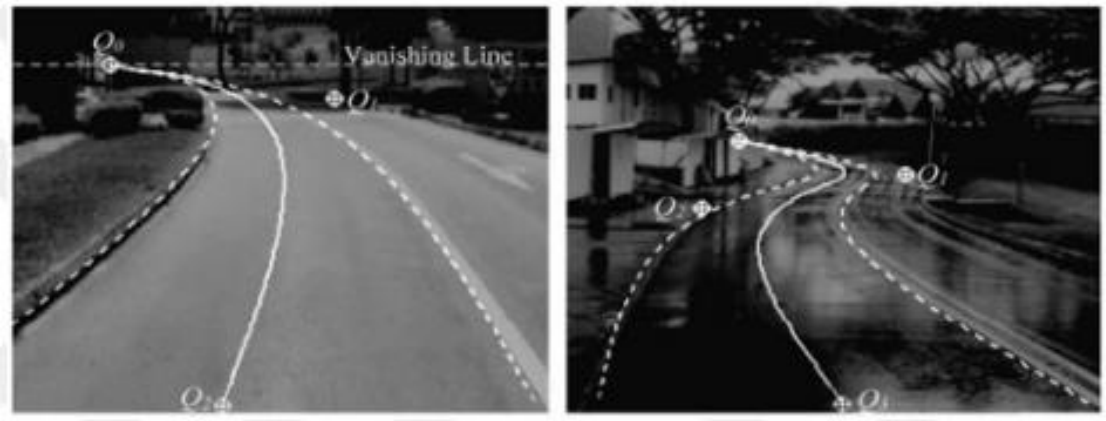
Şekil 2.1: Tipik şerit algılama sistemi [6].

Şerit algılama çalışmasında özelliğe dayalı veya modele dayalı olarak kullanılan pek çok yaklaşım vardır [7]. Özelliğe dayalı yöntemlerin yoldaki şeritleri, şeritlerin yan çizgilerini bulma çalışmasında gölgenin, gün ışığının etkisinde kalma ve silinme gibi çeşitli nedenlerle daha az belirgin olan çizgileri veya şeritleri algılama başarısı azalmaktadır. Model tabanlı yöntemler, yol şeritlerini bir çeşit eğri modeli ve bu modelde birkaç önemli geometrik parametre olarak tanımlar [3]. Bu yöntem, kötü şerit görüntülerine karşı özelliğe dayalı yöntemlere kıyasla daha dayanıklıdır fakat belirli sahnelere uygun olarak oluşturulduklarından farklı bir sahnede başarılı olmayabilir.

1996 yılında Pomerleau ve Jochem, araçtaki kameradan alınan görüntülerin işlenmesi ve görüş sistemi ile şerit ofsetlerinin hesaplanması yoluyla aracın ilgili şeridi takip etmesini sağlayan RALPH adlı sistemi önerdiler [8]. RALPH, yaklaşık 2850 millik bir yolda yapılan testlerde başarılı olurken yoğun trafikten veya güneş ışığından kaynaklı olarak şeritlerin gözükmemesinden etkilenmiştir.

1998 yılında Broggi, araç içindeki kameradan aldığı görüntüyü yeni bir görüntü üzerinde şerit çizgilerini neredeyse dikey çizgiler haline getiren kuşbakışı görünüme dönüştüren GOLD (Genel engel ve şerit tespiti için gerçek zamanlı stereo vizyon paralel sistemi) adlı bir sistemi önerdi [9].

2004 yılında Wang ve arkadaşları, B-Snake şerit olarak adlandırdıkları bir yöntemi kullanmışlardır. Oluşturdukları yöntemde, CHEVP'i (Ufuk noktalarının Canny ve Hough tahmini) kullanarak yolu geometrik bir model olarak belirlemiş ve geometrik model parametrelerini çıkarmışlardır [10]. İlgili algoritma, özellikle gölgelerin şerit verileriyle karıştırıldığı yerlerde oldukça başarılı olmuştur.



Şekil 2.2: 3 veya 4 kontrol noktası kullanarak B-Snake tabanlı şerit modeli [10].

2005 yılında Jung ve arkadaşları yaptıkları çalışmada, kareler açısal yaklaşımı ile kenar algılama yöntemini kullanarak şerit çizgilerini tespit edebilmişlerdir [11].

Kim 2008 yılında yayınlanan makalesinde, standart olmayan kavisli şeritler, standart olmayan şerit değişiklikleri ve beklenmedik işaretler için kararlı bir algoritma önermiştir [12].

Hashim ve Khalifa 2009 yılında yaptıkları çalışmada, araca yerleştirdikleri kamerayla gerçek zamanlı olarak videodan elde ettikleri görüntüleri işleyerek, ışık ve gölge değişimlerinden bağımsız olarak yol şeritlerini tespit edebilmişlerdir [13].

2012 yılında şerit işaretleyicilerini belirleyen ve seyahat yönünü tespit edebilen bir yöntem öneren Mariut, görüntüdeki çizgileri tespit etmede Hough Dönüşümünü kullanmış, şerit işaretleri tespit edebilmek amacıyla şeridin iç kenar boşluğunu çıkarma yoluna gitmiştir [14].

2016 yılında Sun ve arkadaşları RGB renk uzayı yerine HSI renk uzayını kullanarak şerit tespitini gerçekleştirmişlerdir [15].

2.2. Trafik İşareti Algılama

Trafik İşareti Tanıma Sistemleri (TSRS) gelişmiş sürücü destek sistemlerinin önemli bir bileşenini oluşturur ve otonom sürüş, trafik gözetimi, sürücü güvenliği, yol ağı bakımı ve trafik ortamlarının analizi gibi birçok gerçek dünya uygulamalarında gereklidir. Bir TSRS trafik işaretini algılama ve trafik işaretini tanıma olmak üzere iki konuyu ele alır. Trafik işaretini algılamada, ortamdaki işaretin bir çerçevede yerelleştirilmesine çalışılır. Trafik işareti tanımada ise tespit edilen işaretin tipini tanımlamak için bir sınıflandırma gerçekleştirilir [16].

Otomatik trafik işareti tespiti konusunda bilinen ilk araştırma Japonya'da 1984 yılında başladı [17]. Yıllar içerisinde trafik işaretlerinin tespitinde şekil ve renk gibi bir görüntünün görsel yönlerine dayalı çeşitli yaklaşımlarla görüntü işleme teknikleri kullanılmıştır.

Genellikle geleneksel sınıflandırma ve baştan sona öğrenme yöntemleri kullanılarak trafik işaretlerinin tespiti ve sınıflandırılması gerçekleştirilir. Geleneksel sınıflandırmada görüntüden özellikler çıkarmak için bir algoritma tasarlanır ve çıkarılan özelliklerin üstüne bir sınıflandırıcı eğitilir. Bu yolla elde edilen özelliklere çoğunlukla el yapımı özellikler denilir. Geleneksel yöntemlerde özellik çıkarma algoritmaları ne kadar iyi çalışırsa elde edilen doğruluk o kadar artar. Özellik çıkarma algoritmaları nedeniyle sınıflar özellik alanıyla çakışırsa sınıflandırıcılar doğru bir şekilde sınıfları ayırt edemezler. Buna karşılık, CNN gibi baştan sona öğrenme yöntemleri, ham görüntünün,

sınıfların doğrusal olarak ayrılabilir olduğu ve sınıflar arasında örtüşme olmayan bir özellik alanına yansıtması için oldukça doğrusal olmayan bir işlev öğrenir.

2000 yılında Paclik ve arkadaşları trafik işaretinin rengine bağlı olarak bir ikili görüntü üretmiştir ve bu ikili görüntüden birkaç anlık değişmeyen özellikleri çıkarıp hepsini bir arada bir laplace çekirdek sınıflandırıcısına vermiştir [18]. Bu yöntemde görüntünün sınıflandırıcıya getirilmeden önce ikili hale getirilmesi gerektiği sorunu vardır. Maldonado-Bascon ve arkadaşları, görüntüyü HSI renk uzayına dönüştürüp doygunluk ve ton bileşenlerinin histogramını hesaplama yoluna giderek bu sorunu çözmeye çalışmışlardır. Histogram çok sınıflı bir SVM kullanılarak sınıflandırılmıştır [19].

2011 yılında Timofte ve Gool, Larsson ve Felsberg ve 2012 yılında Stallkamp ve arkadaşları, araştırma toplulukları tarafından kullanılabilecek kamuya açık, ek açıklamalar dahil olmak üzere üç zorlu veri tabanını oluşturmuşlardır. Bu veri tabanları sırasıyla Belçika Trafik İşareti Sınıflandırması (BTSC), İsveç Trafik İşareti ve Alman Trafik İşareti Tanıma Benchmark (GTSRB) veritabanları olarak adlandırılmaktadır [20].

Zaklouta ve Stanciulescu 2011, 2012 ve 2014 yılında farklı konfigürasyona sahip Yönlendirilmiş Gradient (HOG) tanımlayıcılarını görüntüyü temsil etmek için çıkarmış ve eğitmek için Rastgele Orman (Random Forest) ile SVM algoritmalarını kullanarak GTSRB veri setindeki trafik işaretlerini tanımlamışlardır. Takip eden yıllarda birçok araştırmacı tarafında HOG tanımlayıcıları kullanılmış ve bu gerçekleştirilen çalışmalar arasındaki temel fark sınıflandırma aşamasında kullanılan algoritmaların farklılığı olmuştur [20].

2011 yılında Fleyeh ve Davami ilk olarak görüntüyü ana bileşen uzayına yansıtmışlardır. Veri tabanındaki görüntüler ile yansıtılan görüntünün öklid mesafesini hesaplayarak görüntünün sınıfını bulmuşlardır [21].

Cireşan ve arkadaşları 2012 yılında, GTSRB yarışması sırasında trafik işareti tanımada CNN'i kullanmışlardır. Önerdikleri CNN insan performansını aşmış ve test görüntülerini %99,46 doğrulukla sınıflandırmayı başararak yarışmayı kazanmışlardır [22].

2013 yılında Huang ve arkadaşları yaptıkları çalışmada, iki aşamalı bir sınıflandırma kullanmışlardır. Görüntü ilk seviyede benzer şekil ve renkte birkaç trafik işaretini içeren süper sınıflardan birine ayrılmıştır. Ardından süper sınıfına göre giriş görüntüsünün

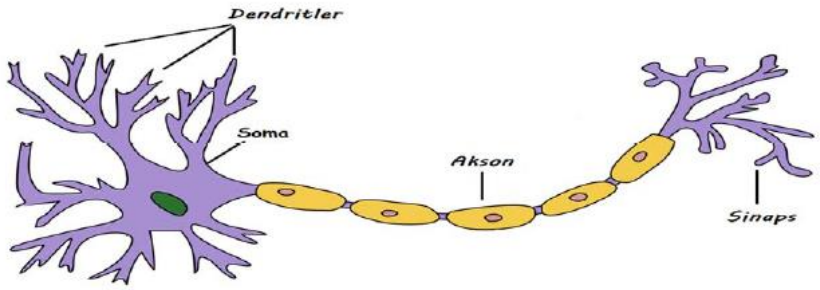
perspektifi ayarlanmış ve ayarlanan görüntüye başka bir sınıflandırma modeli uygulanmıştır [23].

2016 yılında Aghdam ve arkadaşları, GTSDB ve GTSRB veri setleri ile trafik işaretlerini tespit etme ve sınıflandırma çalışmalarında CNN'ni kullanarak trafik işaretini tespit etmede %99,89, sınıflandırmada % 99,55 başarı elde etmişlerdir [20].

2017 yılında Shustanov ve Yakimov, GTSRB veri setini kullanarak 16 trafik işaretini gerçek zamanlı tanımak için derin öğrenme kütüphanesi olan TensorFlow'u kullanarak bir CNN modelini tasarlamışlardır. Modelin performansını hızlandırmak için CUDA kullanan yazarlar trafik işaretini sınıflandırmada %99.94 başarı elde etmişlerdir [24].

2.3. Yapay Sinir Ağları

İnsan beyni, her bir nöronun yaklaşık 10.000 diğer nöronla bağlandığı yaklaşık 86 milyar nörondan oluşmaktadır. Bir nöron dentritler, soma ve aksondan oluşur. Nöron, dendritleri yoluyla diğer nöronların sinapslarından elektrokimyasal sinyaller veya girdiler alır ve bu elektrokimyasal girdilerin toplamı nöronu etkinleştirmek için yeterince güçlü olduğunda, sinyal aksona iletilir. Akson boyunca taşınan sinyalleri bir sonraki nöronlara iletmek için sinapslar kullanılır. Nöronlar arası iletişim sinaps bağlantılarıyla sağlanmakla birlikte nöronların birbirine bağlanabilmeleri için eşik değerini geçmeleri gerekir.



Şekil 2.3: Biyolojik nöron [25].

Beynin tamamı, bu basit birimlerin çok büyük bir kısmının çok karmaşık bir görevi yerine getirmeyi başardığı birbirine bağlı bu elektrokimyasal iletim nöronlarından oluşur.

Bu biyolojik model yapay sinir ağlarının temelidir, ancak yapay sinir ağları (YSA) hala beynin karmaşık modeline yaklaşmaz. İnsan beyni görüntü tanıma gibi bir işlevi yapay sinir ağlarının yaptığı gibi eğitim sürecine ihtiyaç duymadan gerçekleştirir. İnsan

beyninin gerçekleştirdiği işlevlerin yanında YSA'ların gerçekleştirdikleri hala çok kolaydır.

Beyindeki sinir ağlarından ve nöronlardan ilham alınarak oluşturulan YSA, öğrenme sürecini nöronlar arasındaki iletişimle oluşturmakta ve böylece öğrenebilen modeller oluşturmak mümkün hale gelmektedir. Yapay sinir ağları ilk defa 1943 yılında nöroloji doktoru Warren McCulloch ve matematikçi Walter Pitt's tarafından modellendi [26].

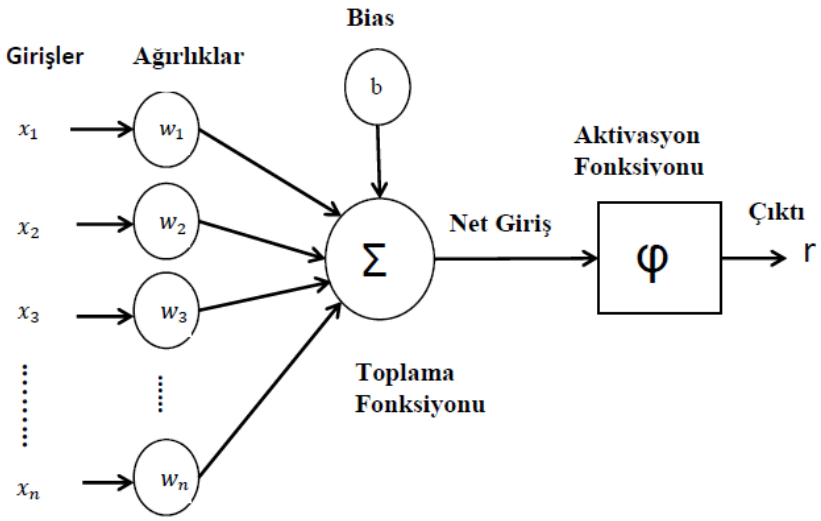
Nörolog uzmanı Donald Hebb 1949 yılında sinir hücresinin biyolojik yapısı üzerine birçok araştırma gerçekleştirerek beynin çalışma yapısını çözmeye çalışmıştır. Çalışmaları neticesinde sinir ağının bağlantı sayısı ile öğrenebilme ve uyum sağlayabilmenin ilişkili olduğunu saptamıştır.

1957 yılında Frank Rosentblatt'ın perceptronu diğer adıyla algılayıcıyı keşfetmesiyle yapay sinir ağı alanındaki çalışmalar hız kazanmıştır. Perceptron tek katmandan ve tek bir çıkıştan oluşan yapay sinir ağıdır.

1969 yılında Minsky ve Papert, yaptıkları çalışmalar ile YSA'nın XOR gibi doğrusal olmayan problemlerde çözüme gidemediklerini belirtmiş ve YSA çalışmaları duraklama devrine girmiştir [27].

1984 yılında, YSA'nın genelleştirilebileceğini ve geleneksel bilgisayar programlama ile çözülmesi zor olan problemlere çözüm üretilebileceğini Donalt Hopfield gerçekleştirdiği çalışmalarla göstermiş ve yapay sinir ağları yeniden popüler olmuştur.

1986 yılında Rumelhart'ın paralel programlama çalışmalarıyla ileri beslemeli modellerde hatanın geriye yayılma algoritmasını (Back propogation algorithm) geliştirmesi ile doğrusal olmayan XOR probleminin çözümü sağlanmış ve çok katmanlı algılayıcıların ortaya çıkmasında önemli bir adım atılmıştır [15]. Broomhead ve Lowe 1988 yılında radyal tabanlı fonksiyonlar modelini çok katmanlı algılayıcılara alternatif olarak geliştirmişlerdir. Yapılan bu çalışmalar ve bilgisayar donanımındaki gelişmelerle yapay sinir ağları giderek daha çok gelişme göstermiştir.



Şekil 2.4: Bir yapay nöron modeli (perceptron).

Yapay sinir ağlarında; girdiler ağırlıkları ile çarpılır ve elde edilen değerler toplanır. Toplam değere bias değeri de eklenerek nöronun uyarılıp uyarılmadığını belirleyen bir aktivasyon fonksiyonuna verilir. Aktivasyon fonksiyonundan alınan çıkış bir sonraki katmanda bulunan nöron veya nöronlarının girişi olarak kullanılır. Bu işlem son tabakaya kadar takip edilir. Ağ sınıflandırma için tasarlanmışsa, giriş son katmandaki en yüksek aktivasyon değerine sahip nöron tarafından etiketlenir [28]. Toplama fonksiyonu çıkış denklemi ise 2.1'deki gibi;

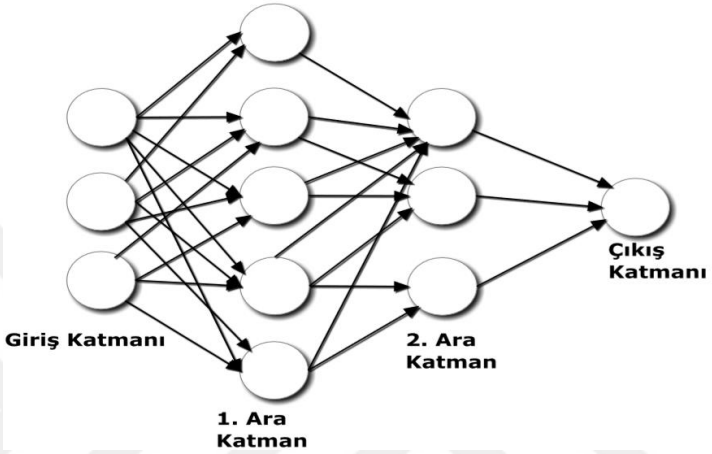
$$r = \sum_{i=1}^{N} w_{i*} x_i + b \tag{2.1}$$

elde edilebilmektedir. Denklem 2.1'de bulunan x giriş değerini, N giriş sayısını, w her bir girişin ağırlığını ve b bias değerini göstermektedir. Toplama fonksiyonu sonucunda elde edilen değer bir aktivasyon fonksiyonundan (φ) geçirilerek nörondan çıkış değeri elde edilir. Nöronun çıkış denklemi ise 2.2'deki gibi;

$$y=\varphi(r) = \varphi(\sum_{i=1}^{N} w_i * x_i + b) \tag{2.2}$$

elde edilmektedir. Kullanılan aktivasyon fonksiyonları çözülmek istenilen problemin yapısına göre değişmektedir. En çok kullanılan aktivasyon fonksiyonları; sigmoid, tanjant veya eşik değer fonksiyonlarıdır.

Ağın giriş katmanı ile son katmanı arasında kalan gizli katmanın sayısı birden fazla olursa çok katmanlı algılayıcı (Multi Layer Perceptron-MLP) olarak adlandırılır. Şekil 2.5'te giriş katmanında 3 nöron, ilk gizli katmanında 5 nöron, ikinci gizli katmanda 3 nöron ve çıkış katmanında 1 nöron bulunan çok katmanlı bir YSA örneği gösterilmektedir. Bu ağ, son nöronun aktif olup olmadığını denetleyerek giriş verisine çıkışta sınıf etiketleme yoluyla ikili bir sınıflandırma işlemi gerçekleştirir. YSA genellikle girişlere bir sınıf etiketi atama veya belirli bir değere eşleme işlevinde kullanılır.



Şekil 2.5: Çok katmanlı YSA yapısı.

YSA modelini tasarlarken kullanılacak gizli katmanların sayısı için veya gizli katmanlardaki nöron sayısı için genel bir formül yoktur. Eğitim hatasını azaltmak amacıyla ağdaki gizli katmanların sayısı arttırıldığında eğitim hatası azaltılabilmekte, fakat algoritmanın karmaşıklığı artmakta ve sistemin genelleme yeteneği azalmaktadır. Bir tasarım konusu olan gizli katman ve gizli katmandaki düğüm sayısı ağda sistem için yeterli sayıda kullanılmazsa model düzgün çalışmaz. Çok fazla düğüm eğitimi daha uzun hale getireceğinden ağ, genelleme yeteneğini kaybedebilir. Aksine, çok az sayıda düğüm olduğunda, karmaşık modelleri çözemeyebilir. Bu hiper parametreler, eğitim sürecinde ağdan istenilen sonuçları elde edebilmek için deneme yanılma ile belirlenir.

İleri yayılım, ağı eğitme sürecinde girdilerin giriş katmanından başlayıp çıkış katmanına ulaşmasıdır. Çıkış katmanında hücrenin hedefteki çıktı değerini vermesi beklenir, istenilen değer elde edilmediği durumda hücrede bilgi taşıyan ağırlıklar değiştirilir. Ağırlıkların değiştirilme işlemi yapay sinir ağlarında öğrenme işlemi olarak bilinir ve bu amaçla geri yayılım algoritması kullanılır. Bir hücreden elde edilmek istenen asıl değer

di, hücreden elde edilen çıkış değeri yi olduğunda, i. nörondaki hata değeri geri yayılım algoritması ile denklem 2.3' teki gibi

$$e_i = d_i - y_i \tag{2.3}$$

hesaplanır. Geri yayılım algoritmasıyla amaç her bir hücre çıkışında bulunan hataların karesel toplamı olan hata fonksiyonunu minimum yapmaktır. Minimum yapılması beklenen hata veya maliyet fonksiyonunun denklemi ise 2.4'teki gibi;

$$E = \frac{1}{2}\sum_i (e)^2 = \frac{1}{2}\sum_i (d_i - y_i)^2 \tag{2.4}$$

elde edilebilmektedir. Çıkışta oluşan hata nedeniyle ağırlıklardaki değişimi ifade eden delta kuralı, hata fonksiyonunun minimum değerini oluşturması amacıyla geliştirilmiştir. Denklemde kullanılan delta değeri nöronda kullanılan aktivasyon fonksiyonunun türevidir. Denklem 2.5 'te ve denklem 2.6'da sırasıyla bir nöron için delta değerinin ve yeni ağırlık değerlerinin hesaplanması gösterilmiştir.

$$\delta_i = \varphi^*(y_i) \times e_i \tag{2.5}$$

$$w_i^* = w_i + \delta \times x_i \tag{2.6}$$

Delta kuralı uygulandığında ağın öğrenme hızı etkilenmemektedir ve dolayısıyla ağın öğrenmesi yavaş olabilmektedir. Bu nedenle bir η öğrenme katsayısı denklem 2.7'de kullanılmıştır. Böylece öğrenme işlemi kademeli olarak gerçekleştirilir. Bu işlem kademeli (gradiyent) azalış işlemi olarak adlandırılır. Hücredeki ağırlıkların hesaplanması amacıyla kullanılan en son denklem ise 2.7'deki gibi;

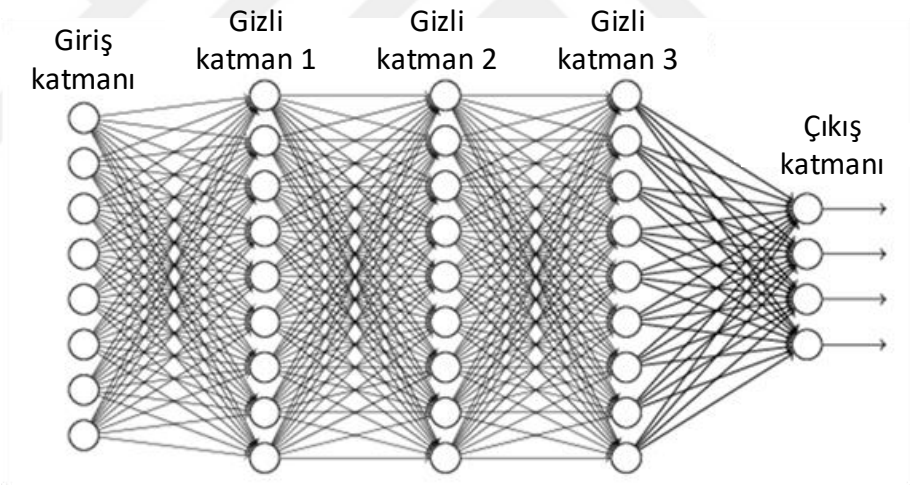
$$w_i^* = \eta \times \delta \times x_i \tag{2.7}$$

verilir. Geri yayılım algoritmasında; her bir tekrarlama işleminde ileri yayılım yapılarak çıkış verileri bulunur ve hata değerleri için tekrar geri yayılım uygulanır ve bu bir devir olarak adlandırılır. Bu işlem hata fonksiyonu minimum değere ulaşana kadar devam eder.

2.4. Derin Öğrenme

Derin öğrenme, beynin yapısal ve işlevsel özelliklerinden esinlenerek tasarlanan, çok katmanlı ağ yapıları olan yapay sinir ağları üzerinde çalışan algoritmalar ve modeller kümesi ile ilgili bir makine öğrenmesi alt alanıdır. Derin kelimesi, birden fazla gizli katmanı ifade eder.

YSA'da oluşturulan modele veriler/girdiler verilmeden önce klasik makine öğrenme algoritmalarıyla özellik vektörleri çıkarılır. Makine öğrenmesi alanında uzun yıllardır gerçekleştirilen bu işlemler derin ağlar ile ortadan kaldırılmıştır. Derin öğrenme ham verileri bünyesinde bulundurduğu birden fazla gizli katmandan geçirirken gerçekleştirdiği işlemler ile geleneksel makine öğrenmesi ve görüntü işleme tekniklerinin aksine, öğrenme sürecinde özellik vektörlerini elde etmektedir. Otomatik öğrenilen özellikler, ağların derin katmanlarında mevcuttur.



Şekil 2.6: Derin sinir ağı [29].

İlk evrişimsel sinir ağı Yann LeCun tarafından 1988 yılında ortaya atılan ve 1998'lere kadar iyileştirmesi devam eden LeNet adlı modeldir. Çoğu YSA özellikle eğitim sırasında büyük bellek ve hesaplama gereksinimlerine sahiptir. Bu gereksinimler ilk zamanlar derin öğrenme çalışmalarında yavaşlamaya neden olmuştur. Sinir ağları üzerine yapılan daha önceki çalışmalarla temelleri atılan derin öğrenme, ilk defa 2012 yılında nesne tanımlama alanında büyük bir yarışma olan ImageNet'te Krizhevsky ve arkadaşlarının derin

öğrenmede temel mimari kabul edilen evrişimsel sinir ağını kullanmaları ile bilim dünyasında büyük etki oluşturmuştur.

Son zamanlarda derin öğrenme çalışmalarının başarılı bir şekilde kullanılmasının başlıca sebeplerinden biri toplumun dijitalleşmesinin gittikçe artmasıyla yeteri kadar verinin olmasıdır. Bir diğer neden ise günümüzde daha büyük modelleri çalıştırmak için hesaplama kaynaklarının var olmasıdır. Birden fazla gizli katmana sahip bir ağın eğitilmesi esnasında geriye yayılım algoritmasının kullanılması için yapılan hesaplamalar paralel işlemciler ile daha hızlı gerçekleştirilebilir. Bu sebeple derin ağların eğitimi için merkezi işlem birimi (CPU) yerine genel amaçlı kullanılmak üzere ortaya çıkan grafik işleme birimi (GPU) kullanılmaktadır [30].

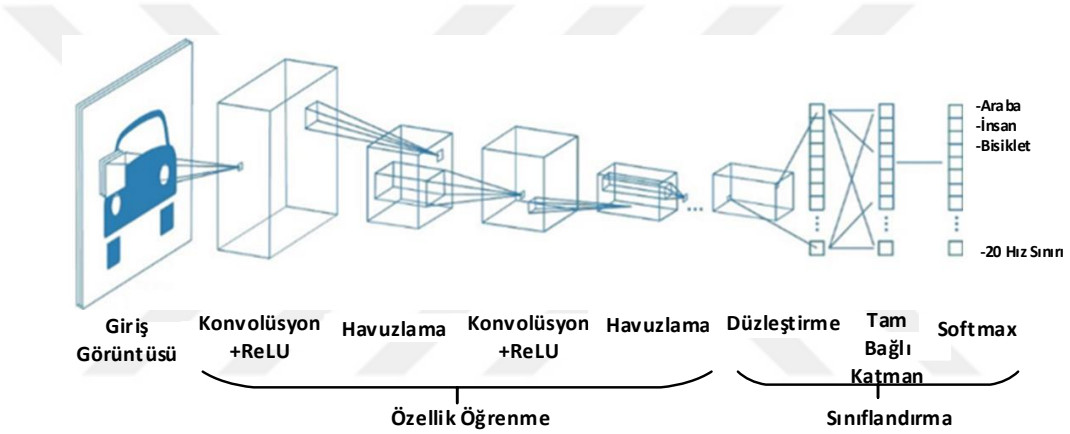
Büyük veri ve GPU'ların geliştirilmesiyle farklı derin öğrenme modellerinin tasarlanılmasına olanak sağlanmıştır. Derin öğrenme modelleri, giriş verisinden kullanıcı tarafından belirlenen özellikler olmadan öğrenme işlemini kendileri yapmaktadır [31]. Bu öğrenme işlemini konvolüsyon işlemleri sayesinde, farklı katmanlarda veriye ait farklı özellikler elde ederek gerçekleştirmektedirler. Bu mimarilerin temel modeli evrişimsel sinir ağı (CNN) olarak kabul edilir. CNN'ler görüntü sınıflandırma, nesne tanımlama, görüntü segmentasyon vb. problemlerde başarılı bir şekilde uygulanmaktadır. YSA'lardaki gizli katman sayılarının daha da arttırıp geliştirilmesiyle derinleşen ağ CNN olarak tanımlanabilir. CNN'deki bu derinlik 2 boyutlu filtrelerin kullanılmasıyla gerçekleştirilir. CNN'nin derinliklerinde öğrenme gerçekleşmekte ve her bir katmanda giriş verisine dair bir alt özellik keşfedilerek özellik çıkarma yapılmaktadır. CNN'nin derinleşen ağ yapısının eğitilmesi aşamasında ezberlemeyi önlemek için dropout [32] yöntemi kullanılmaktadır. Bu yöntem, eğitim aşamasında her iterasyonda ağa ait bazı düğümleri gelişigüzel kaldırarak ezberlemeyi önlemeyi amaçlamaktadır.

Derin öğrenme; görüntü analizi, ses analizi, robotik, otonom araçlar, gen analizleri, kanser teşhisleri ve sanal gerçeklik vb. birçok alanda kullanılmaktadır. Problemlerin çözümünde elde ettiği yüksek doğruluktan dolayı çok yaygın olarak kullanılır. Hatta ses tanımlama, görüntü tanımlama gibi bazı problemlerin doğruluklarında insan performansının üzerine çıkılmıştır.

Facebook ve Google gibi büyük şirketler de görüntülerin etiketlenmesi, nesnelerin tanımlanması ve görüntülerin sınıflandırılması çalışmalarını derin öğrenme modelleri ile gerçekleştirmiştir.

2.4.1. Evrişimsel sinir ağları

Evrişimsel sinir ağı derin öğrenmenin temel mimarisi kabul edilir. Bir CNN mimarisi, insan beynindeki nöronların bağlantı modeline benzer ve görsel korteksin organizasyonundan ilham almıştır. Bireysel nöronlar uyaranlara yalnızca alıcı alan olarak bilinen görme alanının sınırlı bir bölgesinde cevap verir. Bu tür alanların bir koleksiyonu, tüm görsel alanı kapsayacak şekilde üst üste gelir.

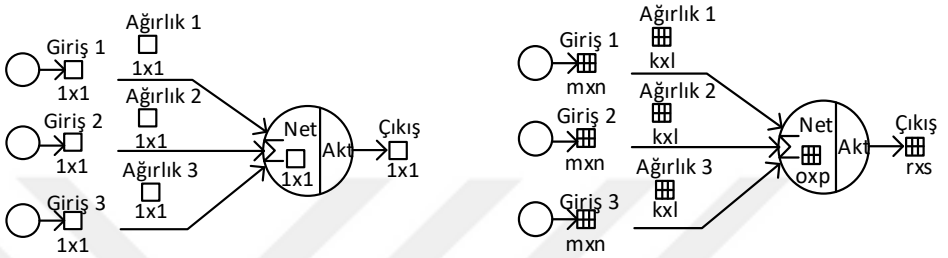


Şekil 2.7: Evrişimsel sinir ağının genel mimarisi [33].

Klasik sinir ağları tek boyutlu özellik vektörleri ile çalışırken evrişimsel sinir ağları, verileri matris formatta alır ve her konvolüsyon katmanında bulunan eğitilebilir filtreler ile işler. CNN, bir görüntüdeki mekansal ve zamansal bağımlılıkları ilgili filtreler kullanarak başarılı bir şekilde yakalayabilir. CNN mimarisi, içerilen parametrelerin azalması ve ağırlıkların yeniden kullanılabilirliği nedeniyle çoğunlukla görüntü veri setinde çalışmaktadır ve bu yöndeki başarısından dolayı Facebook'un otomatik fotoğraf etiketlemesinden otonom arabalara kadar çoğu bilgisayar görmesi sisteminin çekirdeğini oluşturmaktadır.

Basit bir CNN mimarisi oluşturmak için temel olarak konvolüsyon katmanı, havuzlama katmanı ve tam bağlı katman olmak üzere üç ana katman türü kullanılır. CNN aldığı giriş verilerini ardı ardına gelen katmanlarda filtreler kullanarak işler. Filtreler, eğitim sırasında değerlerini kendi kendine öğrenir ve verilerdeki belirli desenleri ortaya çıkarır.

Havuzlama katmanında verileri işleme kolaylığı sağlamak için belirli yöntemler kullanılarak konvolüsyon katmanlarından gelen verilerin boyutunda azaltmaya gider. Son adım olarak, elde edilen veriler vektör haline getirilir ve çok katmanlı algılayıcılar kullanılarak sonuç elde edilir. Elde edilen sonuç ile istenen sonucun farkı kadar bir hata oluşur. Bu hatanın minimum olması istenir. Ağırlıkların güncellenerek hatanın azaltılması için geri yayılım algoritması kullanılarak hata tüm ağırlıklara aktarılır. Her bir iterasyonla ağırlıkların güncellenmesi yapılarak hatanın azaltılması sağlanır.

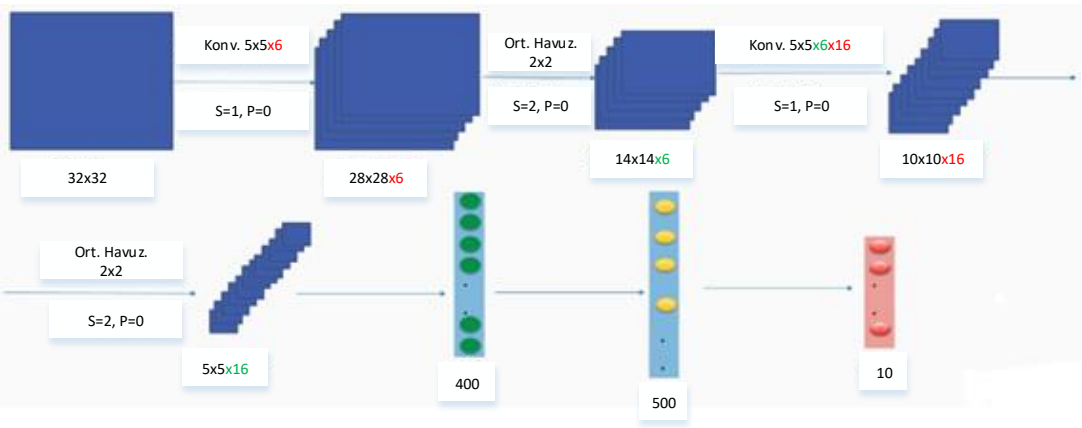


a) Geleneksel YSA düğümü; tüm girişler ve ağırlıklar skaler sayılardır ve skaler çarpım yapılır.

(b) CNN düğümü; matris girişler ve matris ağırlıklarda konvolüsyon işlemi gerçekleşir.

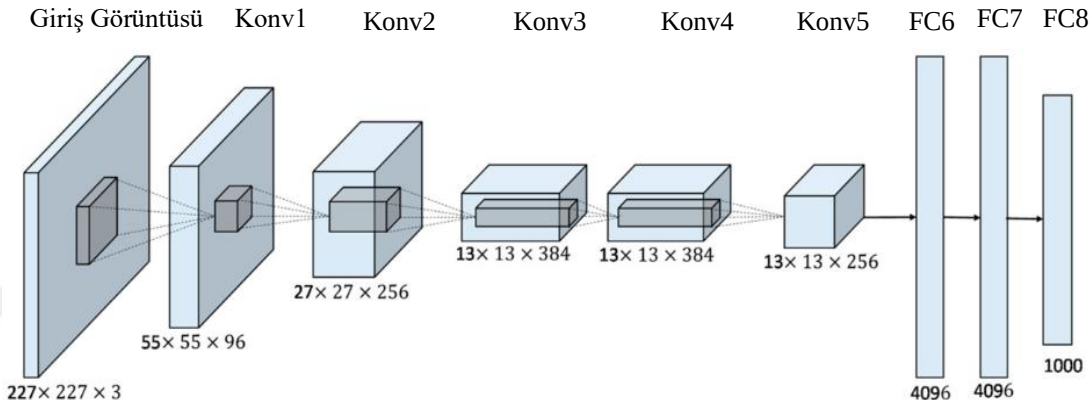
Şekil 2.8: YSA düğümü ve CNN düğümü arasındaki fark [34].

İlk başarılı CNN modeli 1998 yılında Lecun ve arkadaşları tarafından posta numaraları, banka çekleri üzerindeki sayıların okunması için geliştirdikleri LeNet-5 modelidir. Bu modelde boyut azaltma adımlarında ortalama havuzlama işlemi yapılmış, aktivasyon fonksiyonu olarak sigmoid ve hiperbolik tanjant kullanılmıştır. Tam bağlı katmana giren parametre sayısı 5x5x16=400 ve y çıkışında 0–9 arasındaki rakamları sınıflandırdığı için 10 sınıflı softmax bulunmaktadır.



Şekil 2.9: LeNet-5 mimarisi.

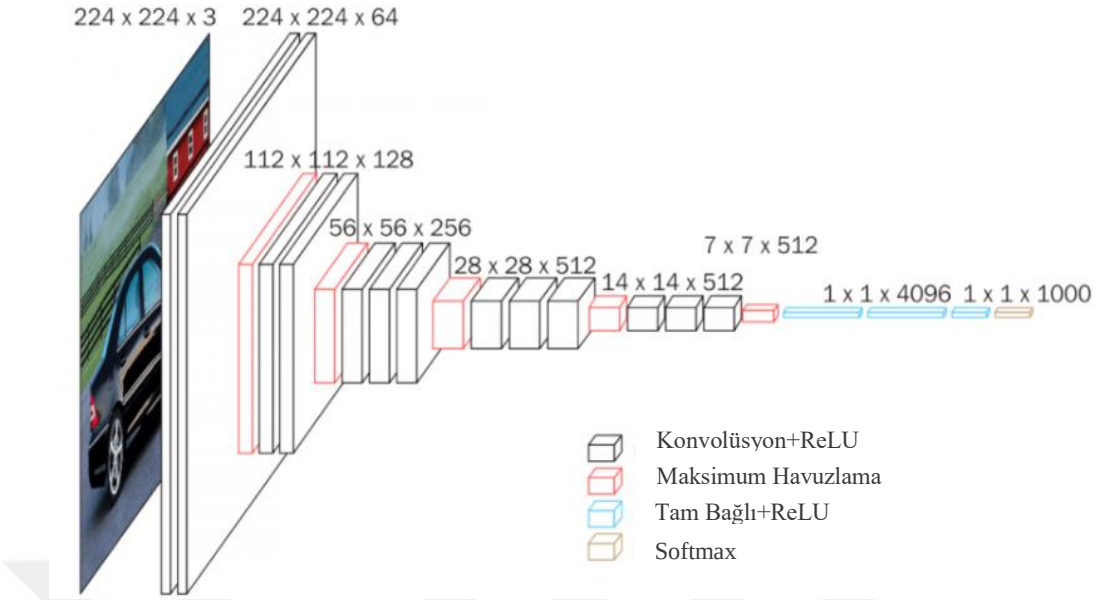
2012 yılında evrişimsel sinir ağ modellerini ve derin öğrenmenin tekrar popüler hale gelmesini sağlayan AlexNet [35] modeli, ReLU aktivasyon fonksiyonunu ve maksimum havuzlamayı kullanmaktadır. AlexNet ImageNet [36] yarışmasında sınıflandırma doğruluk oranını %74,3'ten %83,6'ya artırarak görüntü sınıflandırma probleminde bir kırılma noktası oluşturmuştur.



Şekil 2.10: AlexNet mimarisi [35].

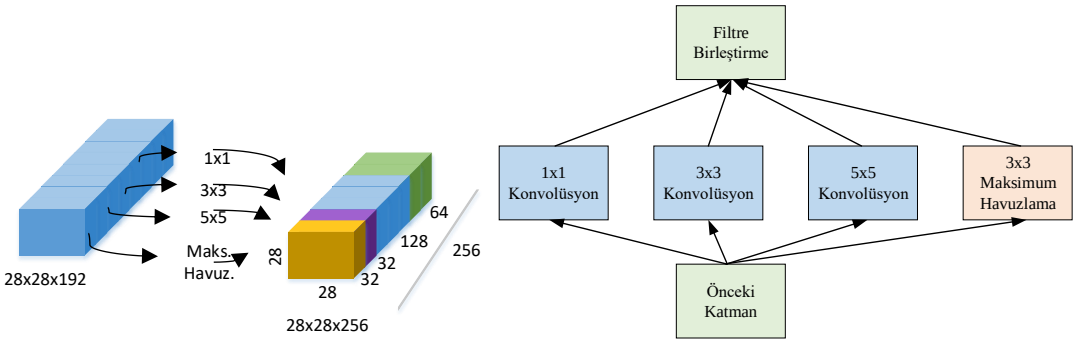
ZFNet [37], CNN'nin iyi performans gösterme nedenlerini analiz etmek ve görselleştirmek için bir prosedür sunmuş ve ağın derinliğinin performans için önemli bir faktör olduğunu deneysel olarak kanıtlamıştır.

Basit bir CNN modeli olan VGG-16 [38], konvolüsyon katmanlarını 2 'li ya da 3 'lü kullanarak ağın derinliğini arttırmaktadır. Tam bağlı (FC) katmanında 7x7x512=4096 nöronlu bir öznitelik vektörü elde eden model, iki FC katmanı çıkışında 1000 sınıflı softmax sınıflandırıcıyı kullanmaktadır. Modelin girişinden çıkışına doğru matrislerin yükseklik ve genişlik boyutları azalırken derinlik değeri artmaktadır.



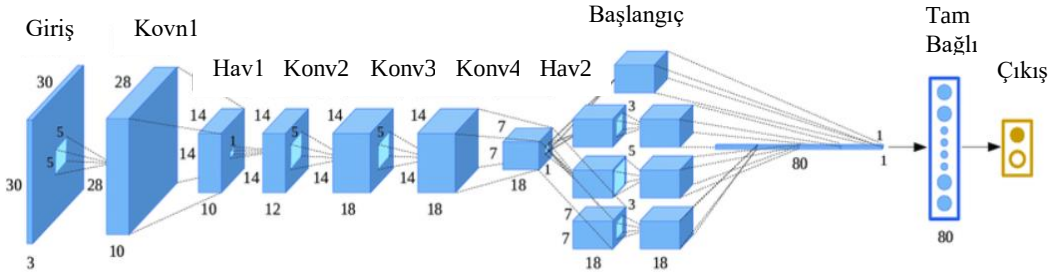
Şekil 2.11: VGG-16 mimarisi [38].

22 katmandan oluşan GoogLeNet [39] modeli 2014 yılında ImageNet resim sınıflandırma yarışmasında birinci olmuştur. Bu model, veri kümesinin çok fazla olması ve katman sayısının artırılması ile sınıflandırma işleminde performansın arttığını ispatlamıştır. GoogLeNet, ağın genişliğini artırmak ve ağın aynı aşamasında kullanılacak 1x1, 3x3, 5x5 gibi farklı boyuttaki evrişimli filtreleri birleştirmek için başlangıç(inception) modüllerini kullanmaktadır.



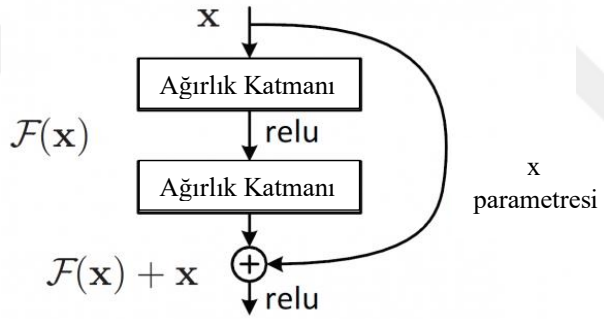
Şekil 2.12: GoogLeNet inception modülü [39].

Katmanları üst üste yığarak bellek boyutu artırımı, zaman kaybı vb. olumsuz faktörleri ortaya çıkarmak yerine girdi değerlerini paralel bir şekilde işleyerek bu gibi durumların önüne geçmektedir.



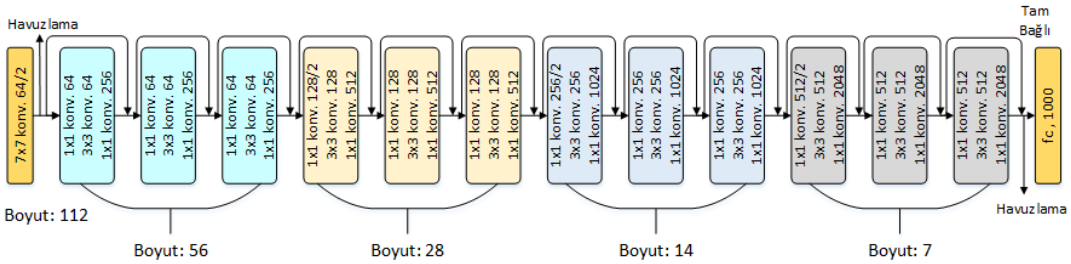
Şekil 2.13: GoogLeNet mimarisi [39].

Diğer mimari yapılarına göre daha derin olan RestNet [40] mimarisi 152 katmandan oluşmaktadır. 2015 yılında yapılan ImageNet resim sınıflandırma yarışmasında, %5-10 arası olan insan hata oranını %3.57 'ye düşürerek birinciliği kazanmıştır. Teorik olarak, modelde katman sayısı arttıkça başarımın artacağı düşünülürken gerçekte böyle olmadığı deneyimlenmiştir. Buradan hareketle oluşturulan ResNet, artık değerleri (residual value) sonraki katmanlara besleyen blokların (residual block) modele eklenmesiyle oluşmaktadır. ResNet bu özelliği ile klasik bir model olmaktan çıkmaktadır.



Şekil 2.14: Artık (residual) blok.

Doğrusal ve ReLU arasında iki katmanda bir eklenen bu değer Şekil 2.14'te gösterilmektedir.



Şekil 2.15: ResNet mimarisi.

Şekil 2.16'da farklı katmanlardan oluşan ResNet mimarisinin boyutları belirtilmiştir.

katman adı	çıkış boyut	18-katman	34-katman	50-katman	101-katman	152-katman
konv1	112×112	7×7, 64, adım 2				
konv2_x	56×56	3×3 maks havuzlama, adım 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
konv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
konv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
konv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	ortalama havuzlama, 1000-d tam bağlı, softmax				

Şekil 2.16: Farklı katmanlardan oluşan ResNet mimarisi.

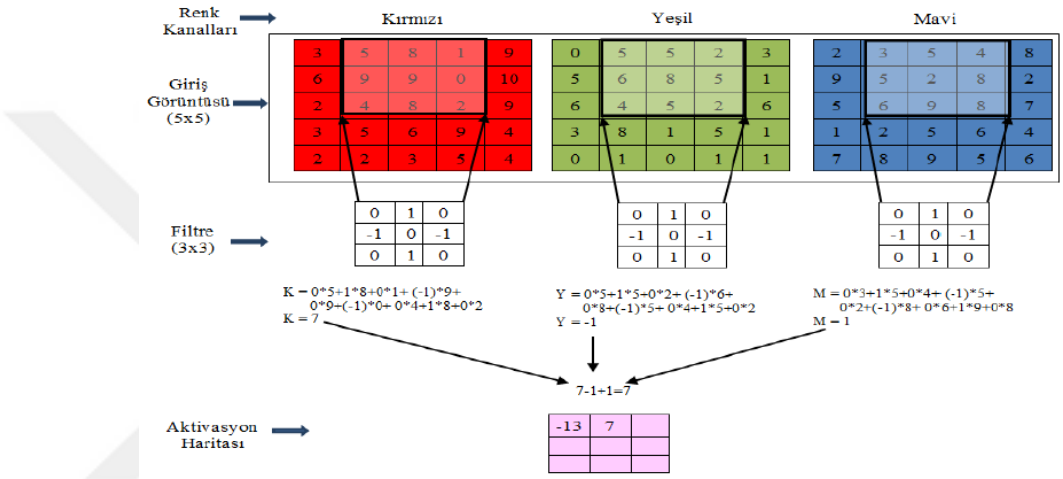
Artık öğrenmenin ve başlangıç mimarisinin bir araya getirilmesiyle sonuçlanan ResNeXt [41], ImageNet sınıflandırma çalışmalarında son teknoloji performans göstermiştir.

2.4.1.1. Konvolüsyon katmanı

Veriler ham olarak CNN'nin ilk katmanı olan konvolüsyon katmanına verilir. Giriş görüntüleri boyutunun büyük veya küçük olması tasarlanacak modelin başarımında önemlidir. Büyük boyuta sahip giriş görüntülerinin kullanımı ağın başarısını artırabillirken, kullanılacak parametrelerin sayısının çok olmasından dolayı yüksek bellek ihtiyacından dolayı eğitim süresini artırabilir, görüntü başına düşen test süresini de uzatabilir. Küçük boyuta sahip görüntülerin kullanılması durumunda ağın derinliği azalacağından eğitim süresi kısalır ve bellek ihtiyacı azalır. Bu durumda ağın performansı düşük olabilir. Giriş görüntü boyutu; ağ derinliği, donanımsal hesaplama maliyeti ve ağ başarısı için uygun seçilmelidir [30].

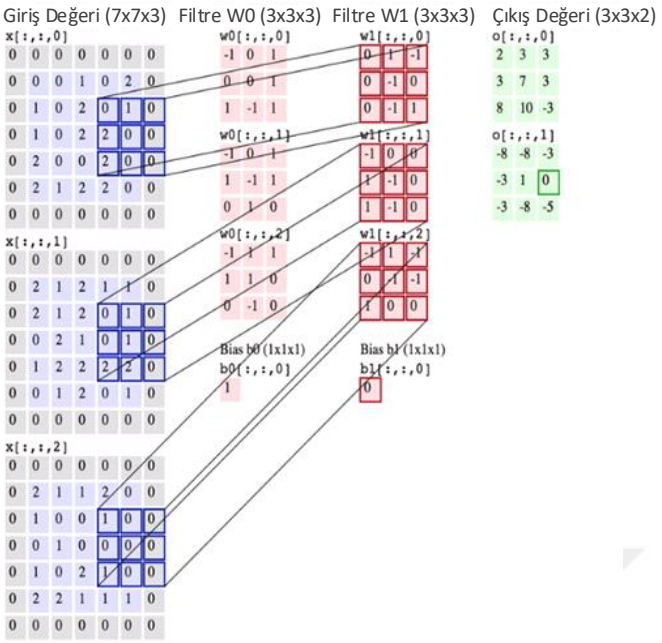
Konvolüsyon katmanı (convulation layer) ayrıca giriş verilerinden özellik çıkartan ilk katmandır. Küçük giriş verisi kareleri kullanarak görüntü özelliklerini öğrenip pikseller arasındaki ilişkiyi korur. Konvolüsyon, görüntü matrisi ile filtre/çekirdek gibi iki girdinin matematiksel bir işlemidir. Bu işlemde, belirli bir filtreyi görüntünün sol üst köşesinden başlatıp sağ alt köşeye kadar tüm görüntü üzerinde dolaştırırken tüm değerler matriste bire bir çarpılır ve tüm değerlerin toplamı çıkış matrisinin ilgili elemanı olarak kaydedilir. Konvolüsyon işleminde kullanılan filtrenin derinliğinin giriş verisinin derinliği ile aynı

olması gerekir. 2x2, 3x3, 5x5 gibi farklı boyutlarda olabilen filtreler bir sayı dizisinden oluşur. Bu sayılar ağırlıkları veya parametreleri ifade eder. Her konvolüsyon işlemi sonucunda 2 boyutlu bir aktivasyon haritası veya özellik haritası (feature map) elde edilir. Filtrelerin her biri giriş verilerinin farklı desenlerini ve özelliklerini öğrenir. CNN'nin eğitimi boyunca bu filtrelerin katsayıları, eğitim kümesindeki her öğrenme yinelemesiyle değişir. Böylelikle ağ, özelliklerin belirlenmesi için, verinin hangi bölgelerinin önem taşıdığını belirler.



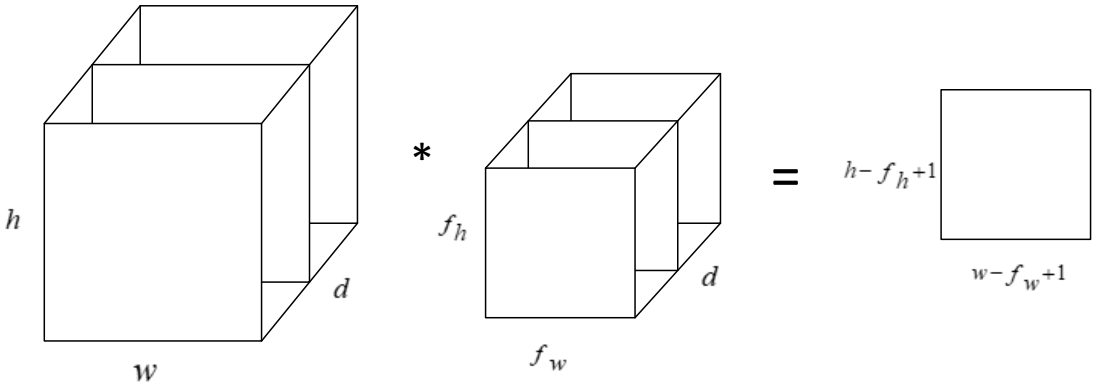
Şekil 2.17: 5x5x3 boyutta bir giriş görüntüsüne 3x3'lük filtrenin uygulandığı konvolüsyon işlemi [30].

Şekil 2.17'de 5x5 matris olan giriş görüntüsü renkli (RGB) olduğundan giriş veri boyutu 5x5x3 olur. 3x3'lük bir filtre belirli bir adım(stride) kaydırılıp giriş görüntüsü üzerinde dolaştırılarak konvolüsyon işlemi gerçekleştirilir. Burada kullanılan görüntü; kırmızı, yeşil ve mavi renk düzleminden oluşan RGB renk uzayında yer alabileceği gibi gri tonlama, HSV, CMYK ve HLS gibi başka renk uzaylarında da yer alabilir. Görüntü matrisinin tümü üzerinde dolaşma gerçekleştirilirken filtre matris sınırına geldiğinde adım sayısı kadar basamak aşağı kayıp soldan sağa tekrar devam eder. Filtre katsayıları her bir renk kanalındaki değerlerle çarpılıp bunların toplamı alınır ve bias değeri de bu toplanan değere eklenerek bir sonuç elde edilir. Bu sonuçlar 3x3'lük iki boyutlu bir aktivasyon haritasını oluşturur.



Şekil 2.18: Birden fazla filtre ile konvolüsyon işleminin gerçekleştirilmesi [42].

Konvolüsyon işleminden sonra Şekil 2.19'da gösterildiği gibi giriş boyutu ve çıkış boyutu arasında farklılık meydana gelir. Burada; h, w, d sırasıyla giriş görüntüsünün yüksekliğini, genişliğini ve derinliğini ifade etmektedir. f_h, f_w, d ise kullanılan filtrenin yüksekliğini, genişliğini ve derinliğini ifade etmektedir.



Şekil 2.19: Konvolüsyon işlemi sonucu özellik haritasının boyut hesabı.

Görüntü matrisinin boyutu;

$$hxwxd \tag{2.8}$$

filtre boyutu:

$$f_h x f_w x d \tag{2.9}$$

olduğundan çıkış verisinin/özellik haritasının boyutu:

$$(h - f_h + 1)x(w - f_w + 1)x1 \tag{2.10}$$

olarak hesaplanır. Konvolüsyon işleminin ardından giriş verisiyle çıkış verisi arasındaki boyut farkının oluşmasını engellemek için giriş matrisine ekstra pikseller eklenebilir. Piksel ekleme işlemine padding denir. 'p' piksel boyutunu ifade eder. Bu durumda çıkış verisinin boyutunu bulmada kullanılan denklem;

$$(h + 2p - f_h + 1)x(w + 2p - f_w + 1) \tag{2.11}$$

olur. Kullanılan padding değeri ise;

$$p = \left(\frac{f-1}{2}\right) \tag{2.12}$$

şeklinde hesaplanır. Burada $f = f_h = f_w$ dir.

Konvolüsyon işleminde ağırlık matrisi olan filtre, görüntü üzerinden belirli adımlarla kaydırılır. Çıkış boyutu kullanılan adım sayısına göre farklılık gösterir. Adım sayısı 's' olarak ifade edilir ve konvolüsyon işleminden sonra çıkış verisinin boyutunu hesaplarken;

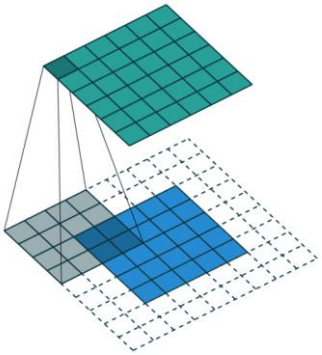
$$\left(\frac{h+2p-f_h}{s} + 1\right) x \left(\frac{w+2p-f_w}{s} + 1\right) \tag{2.13}$$

denklemi kullanılır.

Şekil 2.20'de boyutu 5x5 olan giriş görüntüsüne padding işlemi ile 2 piksel eklenmiş, kayma adımı ise 1 olarak verilmiştir. 4x4 boyutunda filtre kullanılarak gerçekleştirilen evrişim/konvolüsyon işlemi sonucu elde edilen çıkış görüntüsü:

$$\left(\frac{5 + 2x2 - 4}{1} + 1\right) x \left(\frac{5 + 2x2 - 4}{1} + 1\right) = 6x6$$

olur.



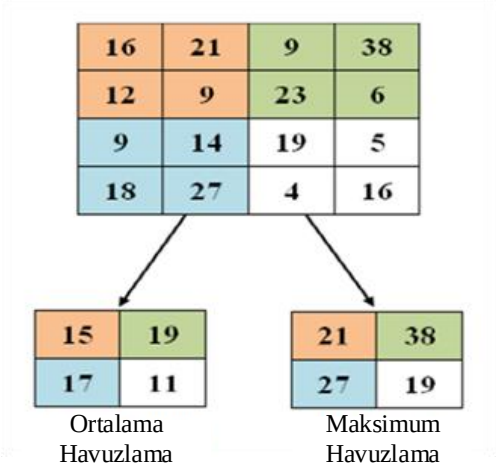
Şekil 2.20: Piksel ekleme(padding).

2.4.1.2. Havuzlama katmanı

Giriş görüntüsü çok büyük olduğunda, eğitilebilir parametrelerin sayısı da çok olur ve bu durumda hesaplama yükü fazla olur. Havuzlama katmanın temel amacı kendinden sonra gelecek katman için giriş verisinin derinliğini sabit tutarak genişlik ve yüksekliği azaltmaktır. Havuzlama katmanı sonucu boyuttaki azalma bilgi kaybına neden olurken sistemin ezberlemesini de önler. Bu katmanda da belirli filtreler tanımlanır ve görüntü üzerinde belli bir adım değerine göre gezdirilerek görüntüdeki piksellerin maksimum değerlerini veya değerlerin ortalamasını alarak işlem yapılır. Genellikle iyi performans gösterdiğinden maksimum havuzlama türü tercih edilir. Bir önceki katman sonucu oluşan filtre sayısınca havuzlama işlemi tüm görüntüler için gerçekleştirilir.

Havuzlama işlemi için belirtilen boyutlarda bir pencere yine belirtilen adım değeri kadar giriş verisinin üzerinden kaydırılır ve belirtilen havuzlama türüne göre işlemler gerçekleştirilir. Maksimum havuzlamada pencerenin kapsadığı alandaki en büyük piksel değeri alınırken, ortalama havuzlamada pencerenin kapsadığı alandaki piksel değerlerinin ortalama değeri alınır. Ortalama havuzlamada çıkan sonuç kesirli ise en yakın sayıya yuvarlama yapılır.

Şekil 2.21'de 2x2 boyutlarında oluşturulan bir pencere her adımda 2 piksel kayma sağlayarak, 4x4 görüntü üzerinde gerçekleştirdiği havuzlama işlemi ve türleri gösterilmektedir.



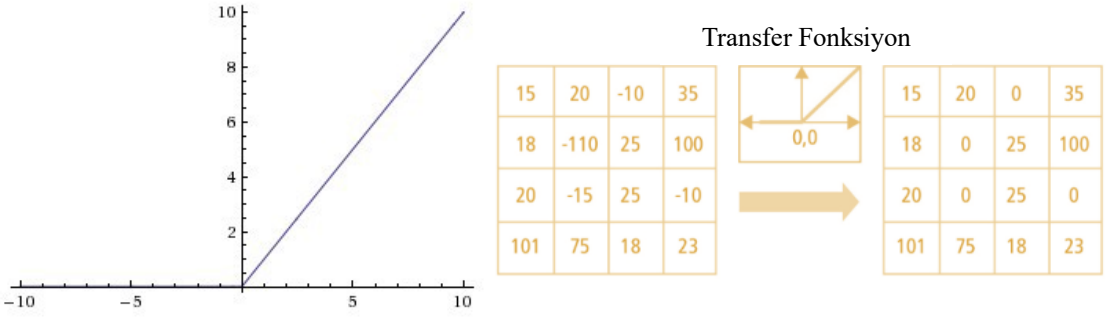
Şekil 2.21: Maksimum havuzlama ve ortalama havuzlama [3].

$$\left(\frac{h+2p-T}{s}+1\right) x \left(\frac{w+2p-T}{s}+1\right) \tag{2.14}$$

Burada 'T' poolin boyutunu yani havuzlama işlemini gerçekleştirecek pencere boyutunu ifade etmektedir. Giriş görüntüsünün yüksekliği h genişliği w ve kayma adım sayısı s 'dir.

2.4.1.3. Düzleştirilmiş doğrusal birim (ReLU)

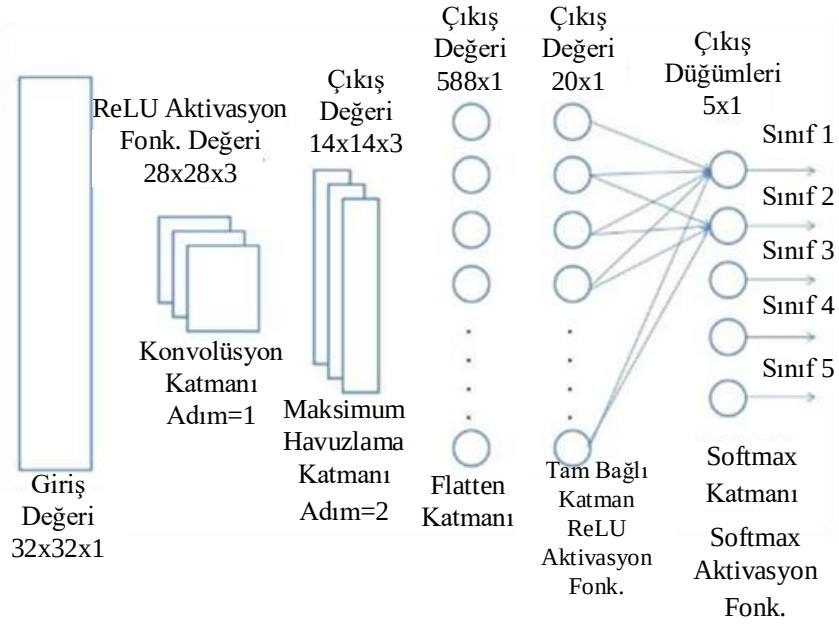
Düzleştirilmiş doğrusal birim (Rectified Linear Unit) son zamanlarda yaygın şekilde kullanılan bir aktivasyon fonksiyonudur. F(x)=max(0,x) işlevini hesaplar. Konvolüsyon katmanından sonra bazen de havuzlama katmanından sonra kullanılan ReLU giriş verilerinden negatif değerleri sıfır yaparken pozitif değerleri olduğu gibi bırakır. Konvolüsyon katmanında gerçekleştirilen işlemlerden sonra doğrusal bir yapı kazanan ağı doğrusal olmayan bir yapıya sokan ReLU, ağın daha hızlı öğrenmesini sağlar. Sigmoid ve tanh aktivasyon fonksiyonları kullanılırken hemen hemen tüm nöronlar aktif edildiğinden hesaplama maliyetli olur. ReLU aktivasyon fonksiyonunun kullanımı ile bazı nöronlar aktif edilmediğinden ağ daha etkin kullanılır.



Şekil 2.22: Düzleştirilmiş doğrusal birim (ReLU)

2.4.1.4. Tam bağlı katman (FC)

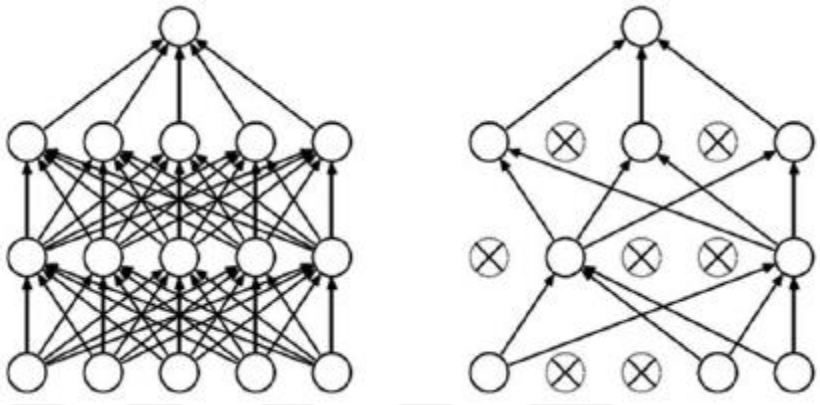
Giriş görüntüsünün daha önceki katmanlardan geçirilerek elde edilen özellik haritaları matristen vektöre düzleştirilir. Tam bağlı katmandaki (full connected layer) nöronlar normal sinir ağlarında görüldüğü gibi, önceki katmandaki tüm nöronlarla tam bağlantılara sahiptir. Şekil 2.23'te gösterildiği gibi 32x32x1 boyutundaki giriş verisi çeşitli katmanlardan geçirilerek özellikler çıkarılır. Bu özellikler düzleştirilerek elde edilen 14x14x3= 588 nöron ileri beslemeli bir sinir ağına beslenir. İleri geçiş tamamlandıktan sonra, geri yayılım algoritması ile hatayı azaltmak için ağırlıklar ile biaslar güncellenir. Her 588 nöron FC katmanında belirtilen 20 nöronlarla bağlantı kurar. CNN mimarisine göre birden fazla FC katmanı da kullanılabilir.



Şekil 2.23: CNN yapısının detaylı görünümü [33].

2.4.1.5. Dropout

Tam bağlı katmanlarda ağın ezberlemesinin önüne geçmek için seyreltme (dropout) işlemi uygulanır. Seyreltme işleminde temel mantık ağın bazı düğümlerini kaldırmaktır. Şekil 2.24'te ağın orijinal yapısı ve dropout işleminden sonraki yapısı gösterilmiştir.



a) Standart CNN ağ yapısı

b) Dropout katmanından sonraki ağ yapısı

Şekil 2.24: Standart bir CNN ağına dropout işleminin uygulanması

Seyreltmede [0,1] aralığında bir eşik değeri kullanılarak zayıf bilgilerin unutulması sağlanabileceği gibi rastgele eleme yöntemiyle de bazı nöronlar kaldırılabilir. Genellikle kullanılan dropout değeri 0.5'tir. Probleme ve veri setine göre istenilen katmanda istenilen dropout değerleri kullanılabilir.

2.4.1.6. Sınıflandırma katmanı

Derin öğrenme mimarilerinde tam bağlı katmandan sonra gelen sınıflandırma katmanında adı üstünde sınıflandırma işlemi gerçekleştirilir. Bu katmanın çıkış sayısı, sınıflandırması yapılacak nesne sayısına eşittir. Örneğin 10 farklı nesnenin sınıflandırılması yapılacaksa, sınıflandırma katmanı çıkış sayısı 10 olmalıdır. Tam bağlı katmanda çıkış değeri 256 olarak seçilirse, bu çıkış değerine göre sınıflandırma katmanı için 256x10 ağırlık matrisi elde edilir [30].

Bir dizi devir (epoch) boyunca model, görüntülerde baskın ve belirli düşük seviye özellikleri ayırt edebilir ve bunları farklı sınıflandırıcılar kullanarak sınıflandırabilir.

Softmax fonksiyonu bir sınıflandırıcıdır. Lojistik regresyon adlı sınıflandırıcı ikili sınıflandırma yaparken softmax fonksiyonu ikiden fazla sınıflandırma yapar. Çoğunlukla tercih edilip kullanılan softmax sınıflandırıcı, sınıflandırmada belirtilen her nesne için 0-1 aralığında bir değer üretir. 10 nesneyi sınıflandırma çalışmasında softmax sınıflandırıcının ürettiği sonuçlardan 1'e yakın sonucu veren nesne ağın tahmin ettiği nesne olmuş olur. 10 sınıf için üretilen değerlerin toplamı 1 olacak şekilde olasılıklar hesaplanır. Örneğin 4 sınıf için gerçekleştirilen sınıflandırma çalışmasında, softmax çıkışında 0.75 kedi, 0.1 köpek, 0.05 ördek, 0.1 kurt şeklinde değerler elde edilirse büyük orana sahip olan kedi girişte verilen görüntüdeki asıl nesne olmuş olur.

2.5. Nesne Algılama

Makinelere görme yeteneği kazandırmayı amaçlayan bilgisayarlı görmede, nesne algılama en önemli konulardan biridir. Fotoğraf veya video karesinde yer alan nesnenin konumuyla birlikte sınıfını tahmin etmeye nesne algılama denilmektedir. Bilgisayarlı görme altındaki farklı çalışmalarda; görüntü sınıflandırma, yerelleştirme, nesne tanıma ve görüntü segmentasyonu gibi işlevler gerçekleştirilmektedir. Bu işlevlerin çalışma yapısı;

a) Görüntü sınıflandırma: Verilen bir giriş görüntüsünün birçok sınıftan birine sınıfla yakınlık derecesini gösteren güven puanı ile doğru sınıflandırılmasıdır.
b) Yerelleştirme: Nesneleri sınıflandırmayla birlikte sınırlayıcı kutu ile tam konumlarının belirtilmesidir.
c) Nesne algılama: Bilgisayarlı görmenin ana görevlerinde biri olan nesne algılama görüntüde bulunan farklı nesnelerin yerelleştirilmesini kapsar.
d) Görüntü segmentasyonu: Görüntüde mevcut olan birden çok nesnenin piksel piksel bölümlendirmesini gerçekleştirir.

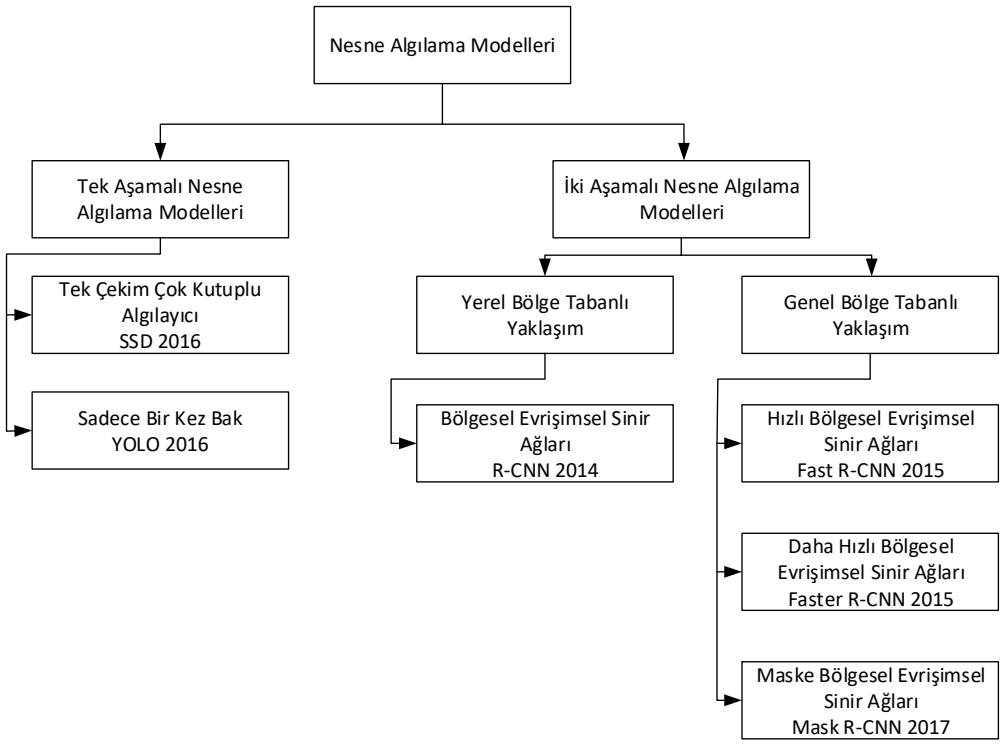
Geleneksel bilgisayarlı görme işlevleri derin öğrenmenin ortaya çıkışı ile tamamen değişmiştir. Derin öğrenme teknolojisi uygulanarak, nesne algılama performansı önemli ölçüde artmıştır ve nesne algılamada en sık evrişimsel sinir ağı (CNN) tabanlı yöntemler kullanılmıştır. CNN; nöron denilen insan beyni hücrelerinin işlevlerinin detaylı çalışılmasından ilham alan araştırmacı Hubel ve Wissel tarafından önerilen teoriye dayanarak, biyolojik olarak esinlenmiş modeldir. Böylece insan beyninin görüntüleri

veya videoları nasıl işlediği, nesnelerin daha soyut bir görüntüsünü elde etmekten sorumlu olan milyarlarca kümelenmiş nöron kümesinin simülasyonuyla nasıl yorumlayacağı gösterilmiştir. Evrişimsel sinir ağlarının ortaya çıkışıyla [35], nesne sınıflandırma ve algılama görevlerinde klasik makine öğrenme yaklaşımlarının karmaşıklığına çözüm getiren birkaç ağ önerilmiştir. Çoğunlukla görüntü sınıflandırma problemi için geliştirilen CNN mimarileri, farklı görsel görevlerde de kullanılmıştır.

CNN, en gelişmiş performansa sahip nesne algılama çalışmaları için yoğun olarak kullanılmaktadır. Nesne algılama, arka plan yamalarının nesne yamalarından ayırt edilmesi olarak da ifade edilebilir. Yama, sahneden kırpılmış keyfi boyutta bir dikdörtgen bölgedir. Bu durumda bir görüntüde; N, tespit edilecek nesne sınıflarının sayısı ve arka plan yamalarını temsil eden bir sınıf olmak üzere toplamda N+1 sınıf vardır.

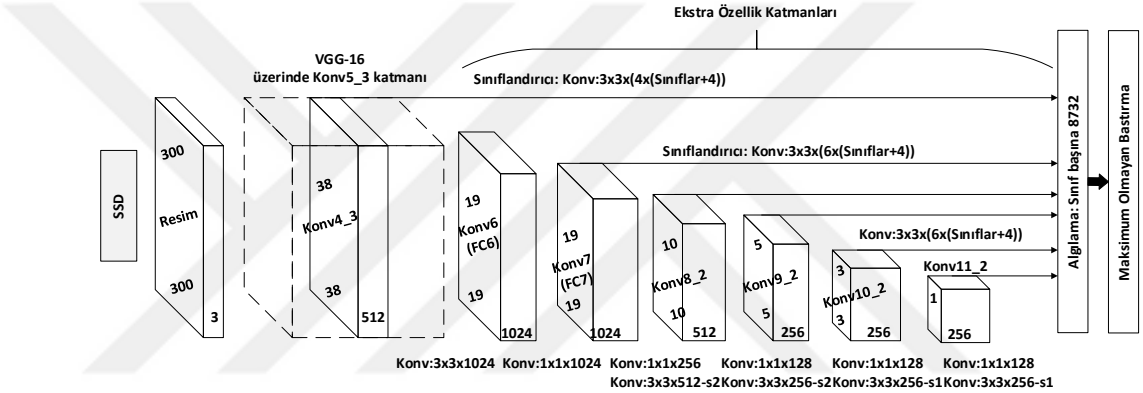
Nesne algılama; ADAS, gelişmiş robotikler, gözetim sistemi, yaya algılama, yüz tanıma, medikal sistemler, güvenlik sistemleri, askeri sistemler, uzay araştırmaları vb. birçok alanda uygulanır.

Girshreik ve ekibi oluşturdukları tek aşamalı nesne algılama modeli ve iki aşamalı nesne algılama modeli ile bir görüntüde bulunan tüm nesneleri, sınırlayıcı kutular ve uygun etiketlerle tanımlamaya çalışmışlardır [43].



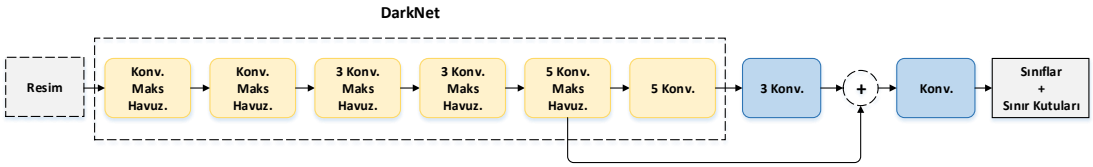
Şekil 2.25: Farklı nesne algılama modelleri [43].

İleri beslemeli olan tek aşamalı nesne algılama modelleri, nesne önerisi üretme yöntemlerini ve yeniden örnekleme aşamalarını kullanmazlar; bunun yerine, performansı artırmak için tüm bu aşamaları tek bir aşamada birleştirerek hesaplama maliyetini azaltırlar. Bu modeller hızlıdır ama doğrulukta yetersizlerdir. SSD (Single Shot Multibox Detector) [44], farklı özellik haritalarından üretilen tahminleri farklı büyüklükteki nesneleri işlemek için değişken bir çözünürlükle birleştirme kapasitesine sahiptir. İki aşamalı nesne algılama modelleriyle kıyasla daha hızlı ancak daha az hassas olan bu model, nesne algılama modülü gerektiren diğer sistemlerle doğrudan entegre edilebilir. SSD özellik çıkarıcı olarak 16 katmandan oluşan VGG16 ağını kullanır.



Şekil 2.26: SSD mimarisi [44].

YOLO [45] görüntüyü tek bir seferde nöral ağdan geçirerek görüntüdeki tüm nesnelerin sınıfını ve koordinatlarını tahmin edebilir. Yani nesne tespitini tek bir regresyon problemi olarak ele almaktadır. PASCAL VOC veri setinde iyi performans göstermesine rağmen hassasiyette iki aşamalı nesne algılama modellerine göre düşüktür.



Şekil 2.27: YOLO mimarisi.

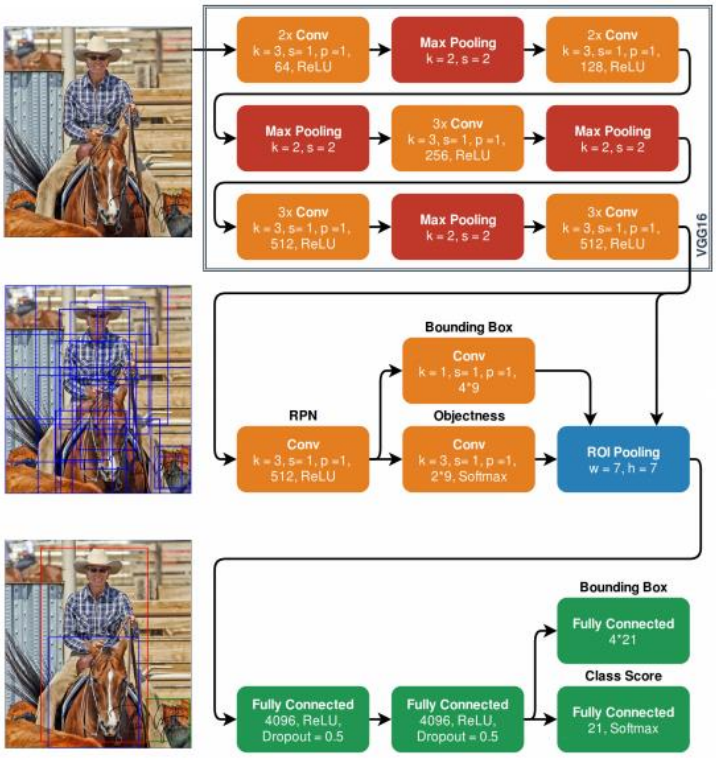
İki aşamalı nesne algılama modellerinin ilk aşaması, nesne önerisi oluşturma yöntemlerini ve ardından karar iyileştirme yöntemlerini kullanmaktır. Nesne önerileri üretmede seçici aramayı kullanmakta ve sınıflandırıcıya özellikler çıkarmak ve beslemek

için CNN'ni kullanmaktadırlar. İkinci aşaması, sınır kutularını iyileştirmeyi, görüntüde tespit edilen diğer nesnelere dayalı olarak kutuları yeniden düzenlemeyi ve sahte algılamaları tanımlamayı içerir. İki aşamalı nesne algılama modelleri, çok yüksek hassasiyete sahiptirler ancak SSD ve YOLO modellerine kıyasla eğitilmeleri zor ve yavaştır.

Nesne tespiti problemi üzerine yapılan CNN tabanlı bir çalışma olan R-CNN [46], orijinal uygulamada bir dış nesne önerme yöntemi olan seçici arama [47] tarafından sağlanan önerilerin kullanıldığı, teklif temelli bir yaklaşım kullanır. Seçici aramanın sağladığı öneriler sabit boyuta getirildikten sonra evrişimsel sinir ağı girişine uygulanır. Her teklif bölgesinin ağdan geçirildikten sonra elde edilen özellik vektörü, her sınıf için bire bir sınıflandırmayı sağlamak amacıyla ikili doğrusal destek vektör makinesine (SVM) verilir ve teklif bölgesine bir sınıfa özgü sınırlayıcı kutu regresyonu uygulanır.

R-CNN'nin bir sonraki sürümü olan Fast R-CNN [48], tüm görüntü için konvolüsyonel özellik haritası oluşturma yaklaşımını kullanır ve hesaplamaları paylaşarak görüntü başına hesaplama maliyetini azaltmak için teklif bölgesini SPPNet tarafından önerilen özellik haritasından elde eder. Özellik haritasından elde edilen bölge önerilerinin boyutları eşitlenerek tam bağlı katmana aktarılır. Daha sonra tespit edilen nesnelerin sınırları belirlenir ve softmax katmanıyla da sınıflandırma yapılır.

Faster R-CNN [49], 3 farklı ölçek ve 3 en boy oranına sahip nesne önerileri oluşturmak için bir Bölge Teklif Ağı (RPN) kullanır. RPN, özellik haritası boyutu her bir giriş görüntüsü için farklı olduğundan, tamamen evrimsel bir yaklaşıma dayalı olarak nesne teklifleri için nesnellik puanları ve karşılık gelen sınırlayıcı kutuları oluşturur.



Şekil 2.28: CNN olarak VGG 16 modelini kullanan Faster R-CNN mimarisi [51].

Faster R-CNN, teklif bölgesine maksimum havuzlamayı uygulayan ve 7x7 gösterimi oluşturan bir ROI havuzlama katmanı kullanır. Nesne tekliflerinin son sınıflandırması, N + 1 çıkış nöronlarına sahip bir softmax sınıflandırma katmanı ile elde edilir ve son sınırlama kutuları, her bir sınıf için ayrı ayrı haritalama parametrelerine karşılık gelen 4 * (N + 1) çıkış nöronları içeren bir regresyon katmanı tarafından üretilir. Burada; 2xConv, aynı tanımlamaya sahip iki adet konvolüsyon katmanını ifade ederken k, s ve p parametreleri, sırasıyla filtre boyutu, adım ve piksel ekleme(padding) olarak tanımlanır. Kırmızı ve yeşil katmanlar sırasıyla maksimum havuzlama katmanları ve tam bağlı katmanlar olarak tanımlanır. Özellik haritası oluşturma, VGG16'nın orijinal uygulamasında tanımlanan son havuzlama katmanından önce VGG16 ağ katmanları ile sağlanır [50]. Özellik haritası RPN katmanına iletilir ve mavi katman olarak tanımlanan ROI havuzlama katmanıyla en yüksek nesne puanlarına sahip olan teklifler kırpılır ve maksimum havuzlanır. Son sınıflandırma ve sınırlayıcı kutu regresyonu için tam bağlı katmanlar kullanılır.

Mask R-CNN [51], piksel düzeyinde segmentasyon gerçekleştirmektedir. Mask R-CNN, maske adı verilen ilave adımla birlikte Faster R-CNN'in üzerine inşa edilmiştir. Bir piksel

alır ve nesnenin bir parçası olup olmadığını tahmin eder. Maske R-CNN, Microsoft COCO veri setinde iyi performans göstermiştir.

Nesne algılama ve tanımanın performansı, büyük ölçüde çıkarılan özelliklerin kalitesine ve sınıflandırıcıların sağlamlığına bağlıdır, çünkü görüntülerin ortaya çıkması aydınlatma koşulları, poz, nesnelerin yansıması ve kameraların gerçek özellikleri gibi birçok faktörden etkilenebilir.

Kim ve arkadaşları 2016 yılında karayolu ortamında nesne tespitini sağlamak için SSD modelini kullanmışlardır. Eğitimde KITTI veri setini kullanan araştırmacılar SSD modeli üzerinde ince ayarlar gerçekleştirerek ve veri setini artırma yoluna giderek nesne tespiti çalışmasında performans artışı sağlamışlardır [52].

Nesne tespiti için derin öğrenme modelleri, nesneleri belirleme yeteneğine sahiptir, ancak her model için tespit doğruluğu değişmektedir. 2018 yılında Shetty ve arkadaşları, SSD modelini kullanarak, sonuçların üretilmesinde çok hızlı olan ancak doğruluğun çok az olduğu bir nesne tespitini gerçekleştirmişlerdir. Faster-RCNN modelini kullanarak gerçekleştirdikleri nesne tespiti çalışmasında ise tespit edilen nesnenin doğruluğunun SSD'ye kıyasla daha yüksek olduğunu ve sonuçları üretmek için gereken zamanın SSD'ye kıyasla daha fazla olduğunu bulmuşlardır [43].

2.6. Transfer Öğrenimi

Son yıllarda önemli ilerleme kaydeden derin öğrenme sayesinde birçok karmaşık sorunların üstesinden gelinmekte ve şaşırtıcı sonuçlar elde edilmektedir. Bununla birlikte, eğitim süresi ve bu derin öğrenme sistemleri için gereken veri miktarı geleneksel makine öğrenmesi sistemlerinden çok daha fazladır. Geleneksel makine öğrenimi ve derin öğrenme algoritmaları çözmeleri gereken probleme özgü tasarlanıp eğitilmektedirler. Transfer öğrenimi, daha yeni modelleri eğitmek için önceden eğitilmiş modellerden bilgileri (özellikleri, ağırlıkları vb.) kullanabilmeyi ve hatta daha yeni görev için daha az veri ile eğitimi gerçekleştirip iyi performans elde edebilmeyi sağlar.

Karmaşık sorunları çözen çoğu model çok fazla veriye ihtiyaç duymakta ve denetlenen modeller için çok miktarda etiketlenmiş veri elde etmek için de çaba ve zaman gerekmektedir. Elde edilen büyük verilerin standart bilgisayar işlemcilerinde

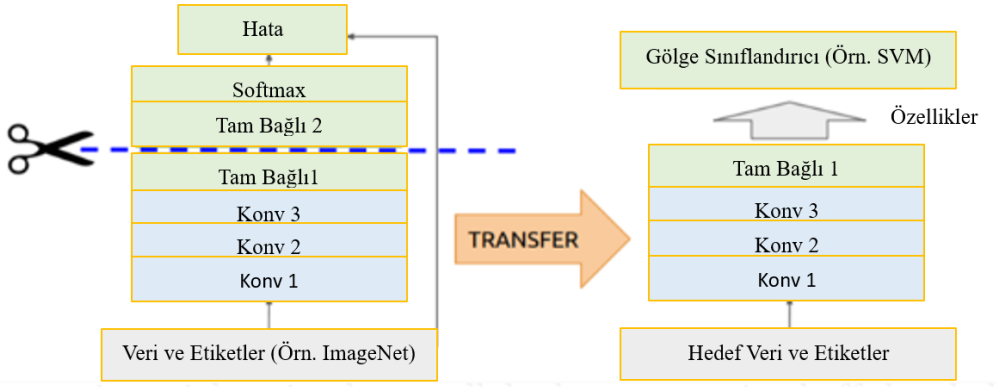
gerçekleştirilmesi de sorun oluşturduğundan grafik işleme birimlerinin kullanılması gerekir. Bir ağ sıfırdan eğitilmek istendiğinde ve ağı eğitmek için yeterince büyük veri setine sahip olunmadığında, transfer öğrenimi ile önceden eğitilmiş bir ağı yeni problemin temeli olarak kullanıp bu durumun üstesinden gelinebilir.

Sinir Bilgi İşlem Sistemleri (NIPS) 1995 çalıştayında "Öğrenmeyi öğrenme: Endüktif sistemlerde bilgi konsolidasyonu ve aktarımı" adlı konu, makine öğrenimi alanında transfer öğrenimi için ilk temel motivasyon olmuştur. Transfer öğrenimi üzerine araştırma, 1995'ten bu yana öğrenmeyi öğrenme, yaşam boyu öğrenme, bilgi transferi, endüktif transfer, çoklu görev öğrenme, bilgi birleştirme, içeriğe duyarlı öğrenme, artımlı / kümülatif öğrenme gibi farklı isimlerde daha fazla dikkat çekmiştir [53]. 2010 yılında Pan ve Yang, Transfer Öğrenimi Üzerine Bir Anket adlı makalelerinde transfer öğrenimini anlamayı sağlamak adına sundukları çerçevede etki alanı, görev ve marjinal olasılıkları kullanmışlardır [54].

Transfer öğrenimi ile önceden eğitilmiş model üzerinde bazı ince ayarlar yapıldığında daha kolay ve hızlı bir şekilde eğitim gerçekleşir. Transfer öğreniminin en önemli artısı, önceden hazırlanmış ağın zaten zengin bir özellik kümesini öğrenmiş olmasıdır. Bu özellikler çok farklı problemlerde de işe yaramaktadır. Örneğin, milyonlarca görüntü üzerinde eğitilmiş bir ağın elde ettiği zengin özellik çıkarımlarıyla sadece yüzlerce görüntü barındıran bir veri kümesi üzerinde yeni bir sınıflandırma yapmak için yeniden eğitilebilir.

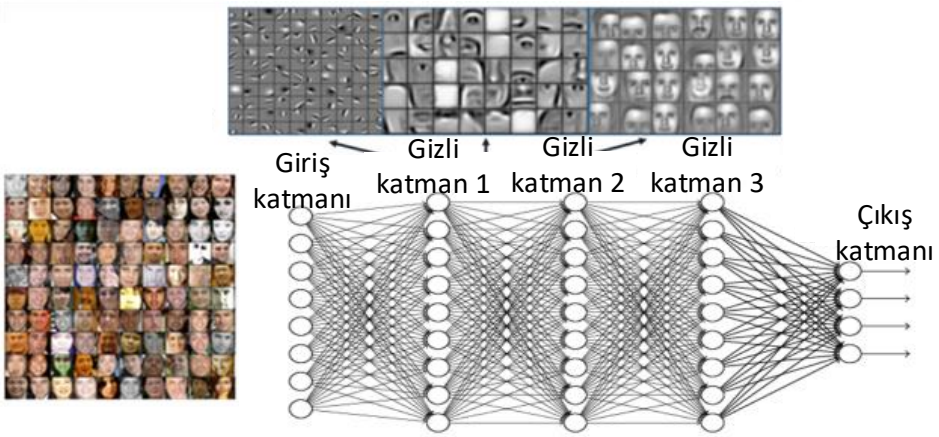
Transfer öğrenimi için en popüler iki strateji:

a) Özellik çıkarıcı olarak hazır önceden eğitilmiş modeller: Derin öğrenme sistemleri ve modelleri, farklı katmanlarda farklı özellikleri öğrenen katmanlı mimarilerdir. Özellikleri çıkarmak için önceden eğitilmiş Inception V3, VGG vb. modellerin ağırlıklı katmanlarını güncellemeden sadece softmax çıkışını yeni görev için gereken sınıf sayısına göre ayarlamak gerekir. Bu, transfer öğrenimi gerçekleştirmek için en yaygın kullanılan yöntemlerden biridir.



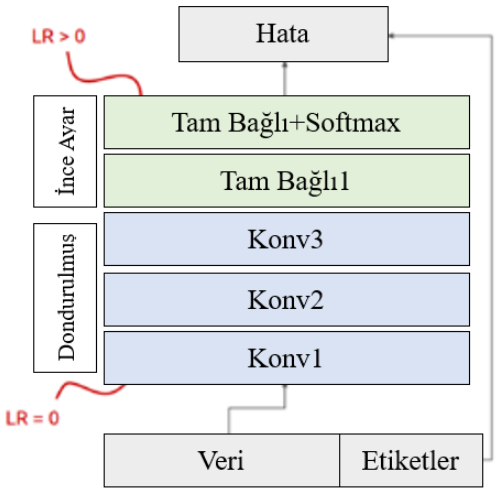
Şekil 2.29: Özellik çıkarıcı olarak transfer öğreniminin gerçekleştirilmesi [55].

b) Önceden eğitilmiş modellerde ince ayar: Bu, sadece son katmanın değiştirilmediği, aynı zamanda önceki katmanlardan bazılarının da seçilip yeniden eğittildiği bir tekniktir. Derin sinir ağları, çeşitli hiper parametrelere sahip, yapılandırılabilir mimarilerdir. Bu mimarilerin ilk katmanlarında genel özellikler öğrenilir, daha sonraki katmanlarda ise göreve özgü özellikler öğrenilir. Şekil 2.30'da yüz tanıma probleminde katmanlarda elde edilen özellikler gösterilmektedir.



Şekil 2.30: Yüz tanıma çalışmasında katmanlardan özelliklerin elde edilmesi [55].

Bu stratejide, yeniden eğitim gerçekleştirildiğinde belirli katmanlar dondurulur yani ağırlıklar sabitlenir ve geri kalan katmanlarda gereksinime göre ince ayarlamalar yapılır. Böylece sabitlenen ağırlıklar yeniden eğitim aşamasında başlangıç noktası olarak kullanılır ve daha az eğitim süresiyle daha iyi performans elde edilmesi sağlanır.



Şekil 2.31: Önceden eğitilmiş ağlarda ince ayar ve dondurma işlemleri [55].

Transfer öğrenimi için temel gerekliliklerden biri, yeni görevde iyi performans gösterecek modelin seçilmesidir. VGG-16, VGG-19, Inception V3, XCeption ve ResNet-50 gibi birçok popüler modeller derin öğrenme dünyasında kullanıcılara sunulmuştur [55].

Transfer öğrenimi gerçekleştirilmek istendiğinde yukarıdaki iki strateji de göz önüne alındığında dört senaryo ortaya çıkmaktadır.

a) Veri benzerliği çok yüksekken veri seti boyutunun küçük olması: Bu durumda, veri benzerliği çok yüksek olduğundan, modelin yeniden eğitilmesine gerek yoktur. Yapılması gereken tek şey, çıktı katmanlarını problem tanımına göre değiştirip özelleştirmektir. Örneğin yeni eğitilecek modelde son katmanda 5 sınıfa göre bir çıkış almak isteniliyorsa ve önceden eğitilmiş model 23 sınıfa göre çıktı veriyorsa; önceden eğitilmiş modelin tam bağlı katmanına ve softmax çıkışına müdahale ederek 5 çıktı verecek şekilde bir ayarlamaya gidilir.
b) Veri benzerliği çok düşükken veri seti boyutunun küçük olması: Bu durumda, önceden eğitilmiş modelin ilk k katmanları dondurulur ve geriye kalan (n-k) katmanlarını tekrar eğitilir. Dondurulmayan katmanlar daha sonra yeni veri setine göre uyarlanır. Yeni veri setinin düşük benzerliği olduğundan, daha yüksek katmanları yeni veri setine göre yeniden eğitmek ve özelleştirmek önemlidir.
c) Veri benzerliği çok düşükken veri seti boyutunun büyük olması: Burada sahip olunan veriler, daha önce modeli eğitmek için kullanılan verilerle

karşılaştırıldığında çok farklı olduğundan önceden eğitilmiş modeller kullanılarak yapılan tahminler etkili olmayacaktır. Bu durumda sahip olunan bu büyük veri kümesini sıfırdan eğitme yoluna gidilir.

d) Veri benzerliği yüksekken veri seti boyutunun büyük olması: Bu ideal durumdur ve önceden eğitilmiş model en etkili şekilde kullanılır. Modeli kullanmanın en iyi yolu, modelin yapısını ve modelin ilk ağırlıklarını koruyarak yeni veri setiyle eğitimi gerçekleştirmektir.

Şeker 2018 yılında kumaş hatalarının tespiti çalışmasında, 1000 nesneyi sınıflandırmak amacıyla bir milyondan fazla görüntü üzerinde eğitilmiş olan AlexNet ağını kullanarak transfer öğrenimi gerçekleştirmiştir. Katmanların ve hiper parametrelerin AlexNet'inki ile tamamen aynı olan bir ağı da sıfırdan eğiten Şeker, % 74.87 performans elde ederken transfer öğrenimi ile % 98.75 performans elde etmiştir [56].

BÖLÜM 3. MATERYAL VE YÖNTEM

Bu tez çalışması kapsamında görüntü işleme yöntemleri kullanılarak şerit algılama yapılmıştır. Derin öğrenme modellerinden SSD Inception V2, Faster R-CNN Inception V2, Faster R-CNN Resnet 50 ve Faster R-CNN Resnet 101 aynı veri seti üzerinde çalıştırılıp nesne algılamadaki doğrulukları karşılaştırılarak en iyi performansa sahip model bulunmaya çalışılmıştır.

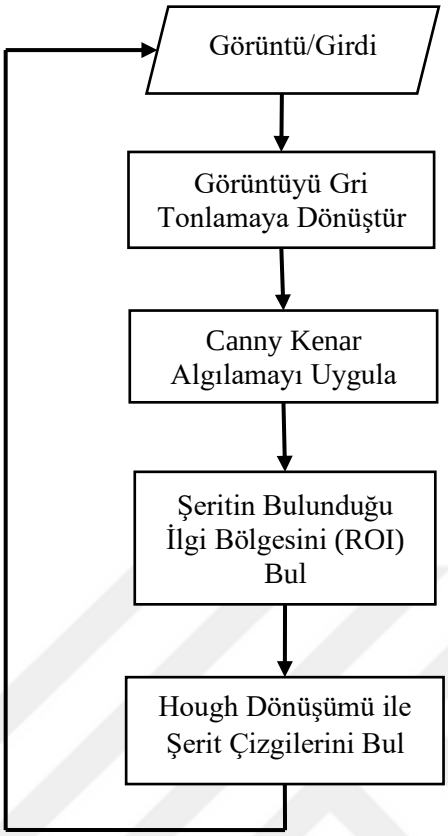
3.1. Şerit Algılama

Bu çalışmada araç içine yerleştirilen kameradan alınan görüntülerden aracın içinde bulunduğu şerit görüntü işleme yöntemleri ile tespit edilmiştir.

Son zamanlarda popüler çalışmalardan olan otonom arabaların en önemli işlevlerinden biri şerit algılamadır. Herhangi bir sürüş senaryosunda şerit çizgileri; trafik akışını ve bir aracın sürmesi gereken yeri gösteren önemli bir bileşendir. Python programlama dili kullanılarak; OpenCV, Canny kenar algılama ve Hough dönüşümü ile aracın içinde hareket ettiği beyaz renkteki şeridi algılama gerçekleştirilmiştir.

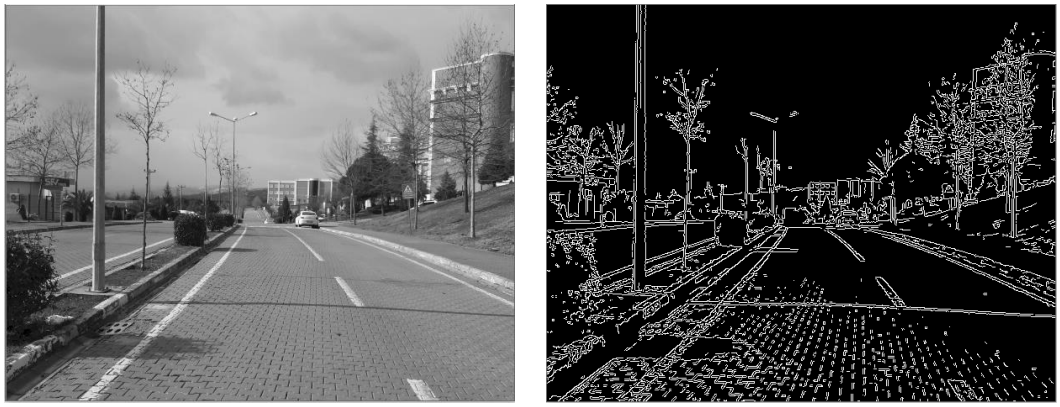
Şekil 3.1'de gösterildiği gibi şerit algılamanın gerçekleştirilmesi için girdi verisi olarak resim veya kameradan kare(frame) alınır.

Şekil 3.1: Görüntünün alınması.



Şekil 3.2: Şerit algılama akış diyagramı.

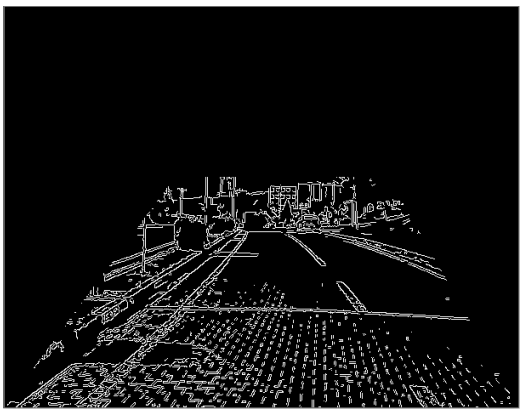
Elde edilen görüntü öncelikle gri tonlamaya dönüştürülür, ardından birbirine bitişik piksellerin gradyanı (değişim oranı) hesaplanır. Kenarları hesaplamak için Canny kenar algılama kullanılmıştır.



Şekil 3.3: Görüntünün griye dönüştürülmesi ve Canny kenar algılamanın uygulanması.

Kenar algılama işleminin ardında görüntü, şerit çizgileri içermeyen bazı kısımlara sahiptir. Bunları kaldırmak ve şerit çizgilerini izole etmek için çokgen(poligon) olarak

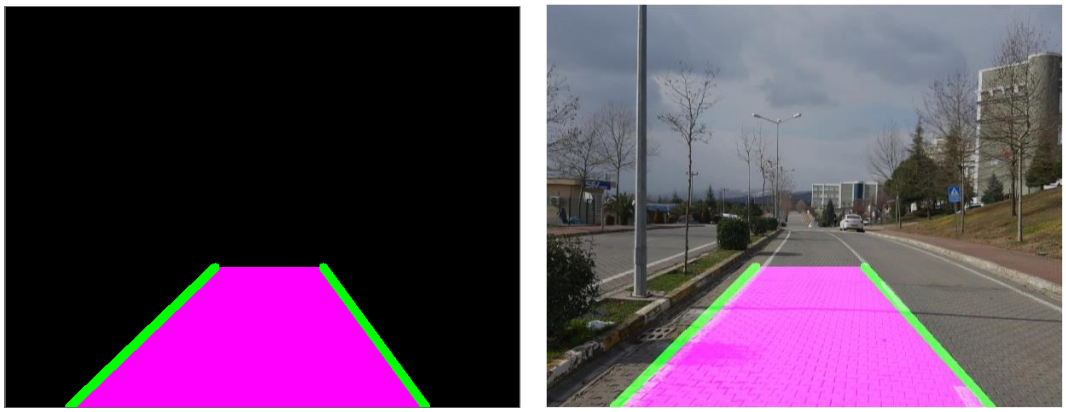
temsil edilen ilgi bölgesi (ROI) kullanılmıştır ve bu bölge dışındaki diğer alanlar maskelenmiştir.



Şekil 3.4: İlgi bölgesi (ROI) bulma.

Canny kullanılarak elde edilen kenarlar birçok noktadan oluşur. Hough; kenar pikselleri içeren bir görüntüden çizgileri üretme işlevine sahiptir. Hough dönüşümü kullanılarak Canny kenar algılamada bulunan noktalardan çizgiler tespit edilir ve bu tespit edilen çizgilerin bir listesi oluşturulur. Elde edilen çizgileri sağ ve sol şerit çizgileri olarak ayırt etmek ve ayırt edilen çizgileri tek bir çizgi ile göstermek gerekmektedir. Çizgileri sağ ve sol gruplara ayırmak için her bir çizginin eğimi bulunmalıdır. Bir çizginin eğiminin yönü, çizgi boyunca soldan sağa doğru ilerlerken çizginin yukarı veya aşağı doğru hareket edip etmediğini açıklar. Negatif eğimli bir çizginin aşağı doğru hareket ettiği, pozitif eğimli bir çizginin yukarı doğru hareket ettiği kabul edilir. Eğim yönü, kaynağın sol alt köşesinde olduğu normal koordinat sistemlerinde bu şekilde çalışır. Fakat bir görüntüdeki koordinat sistemi sol üst köşeden başladığından eğim yönleri yukarıda anlatılanla ters olmaktadır; negatif eğimler yukarı doğru ve pozitif eğimler aşağı doğru hareket edecek şekildedir. Görüntüde, sol şerit işaretlerinin hepsinin negatif bir eğimi vardır, bu da çizgiler boyunca soldan sağa doğru hareket ederken çizgilerin ufka doğru hareket ettiği anlamına gelir. Öte yandan, sağ şerit işaretlerinin hepsinde pozitif bir eğim vardır, bu durumda çizgiler boyunca soldan sağa doğru hareket ederken çizgilerin görüntünün alt kısmına doğru hareket ettiği anlamı çıkar. Bu, sol ve sağ çizgileri gruplamak için kullanılan ayrım olmaktadır. Şerit işaretleri eğimde aşırı görünür, bu nedenle mutlak değeri 0,5'ten düşük bir eğime sahip çizgiler dikkate alınmamaktadır. Tespit edilen tüm çizgiler üzerinden geçilerek eğimler hesaplanır ve eğimi negatif olanlar sol şerit çizgisi grubuna, pozitif olanlar sağ şerit çizgisi grubuna dahil edilir. Şerit grubundaki her bir çizginin eğimi ve

kesişimi hesaplanır, daha sonra eğimi ve kesişimi ortalanarak şeritte tek bir çizgi üretilir. Bu işlem her iki şeritte yapılır.



Şekil 3.5: Hough dönüşümü ile şeritleri bulma.

Son adım olarak tespit edilen çizgiler ve şerit giriş görüntüsüne yerleştirilir. Bu işlem iki görüntünün ağırlıklı toplamı ile yapılır.

3.2. Nesne Algılama

Bu tez çalışmasında içerisine kamera yerleştirilen bir arabanın seyir halindeyken veya durağan iken görüş alanı içerisinde bulunan insan, araba, bisiklet ve belirlenen trafik işaretlerini algılaması ve tanıması sağlanmıştır.

3.2.1. Veri seti oluşturma

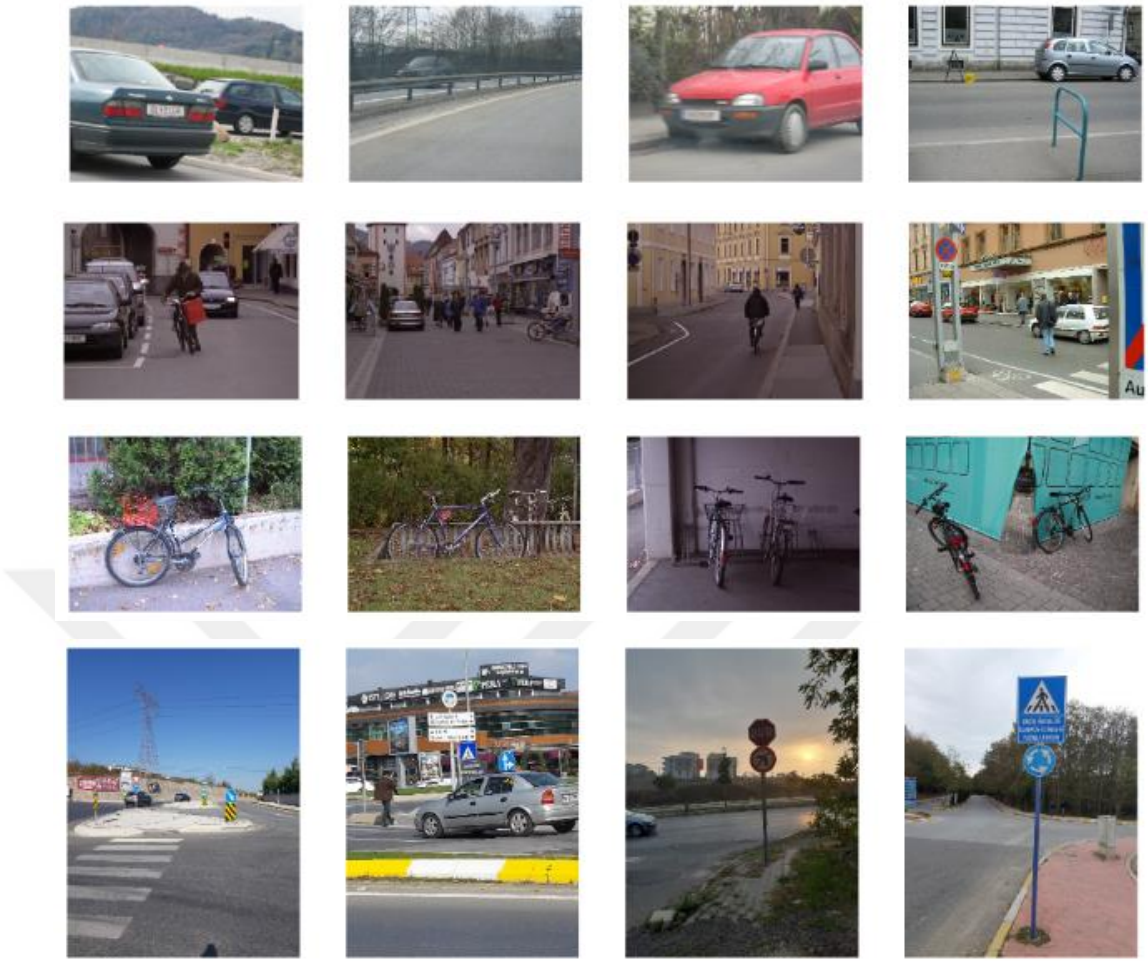
Derin öğrenme ve genel olarak makine öğrenmenin önemli bir unsuru veri setidir. Görüntülerde nesne algılama alanında, araştırmacılar ve uygulayıcılar tarafından sıkça kullanılan ve referans alınan veri setleri; Pascal, ImageNet, COCO ve SUN'dır. Bu çalışmada ağı eğitmede kullanılacak insan, araba ve bisiklet görüntülerinin bir kısmı nesne sınıflandırma ve nesne tanıma çalışmaları için oluşturulan GRAZ-01 ve GRAZ-02 [57] veri setlerinden bir kısmı da cep telefonu kamerasıyla elde edilmiştir. Trafik işaretlerinden eğitimde kullanılacak yaya geçidi, dur, yol ver, döner kavşak, kavisli yol, 20 hız sınırı ve 30 hız sınırı işaretleri cep telefonu kamerasıyla çekilmiştir. Resimler farklı açılar ve farklı ışıklandırmalar altında elde edilmiştir. Farklı boyutlara sahip olan resimlerin tümü jpg formatındadır.

Çok katmandan oluşan büyük ağ, az veri ile eğitilmeye çalışıldığında aşırı uyum sağlamaya neden olabilirken büyük miktarda veri içeren az katmanlı sığ ağ ise eğitimde yeterli doğruluk vermeyebilir. Bu nedenle, ağ derinliği ve veri miktarı arasında bir denge kurmak çok önemlidir. Ağ derinliği, veri miktarı ile ilişkilidir. Çalışmada kullanılacak toplam 517 görüntünün %20 'si test veri setine ayrılmıştır. Veri setini eğitim ve test verileri olarak ayırırken her iki veri setinde aynı görüntünün olmamasına dikkat edilmelidir. Model eğitildiğinde aşırı uyum sağlama probleminin(overfitting) ortaya çıkmasını engellemek için seyreltme (dropout) kullanılabilir.

Tablo 3.1: Veri setinde kullanılan nesneler ve sayıları.

Nesneler	Eğitim Seti	Test Seti
İnsan	169	35
Araba	503	107
Bisiklet	96	25
Dur	30	12
Yaya Geçidi	87	28
Yol Ver	59	15
Döner Kavşak	54	12
Kavisli Yol	30	5
30 Hız Sınırı	25	7
20 Hız Sınırı	13	7

10 nesneyi sınıflandırmak için veri setinde bulunan 517 görüntü, bir modeli sıfırdan eğitmekte iyi bir performans göstermeyebilir. Az veriyle model eğitimi gerçekleştirme imkânı sağlayan transfer öğrenimi yönteminde dahi her nesne için en az 200 görüntünün olması idealdir. Önceden eğitilmiş modelin bu tez çalışmasında kullanılacak sınıfları içeriyor olması yeni eğitim sonucunda daha fazla verim almayı sağlayacaktır. Bu çalışmada transfer öğrenimi yöntemi kullanılarak belirlenen 4 modelin eğitimleri gerçekleştirilmiş ve iyi performans sağlayan model tespit edilerek nesne algılama çalışmalarına bu modelle devam edilmiştir.



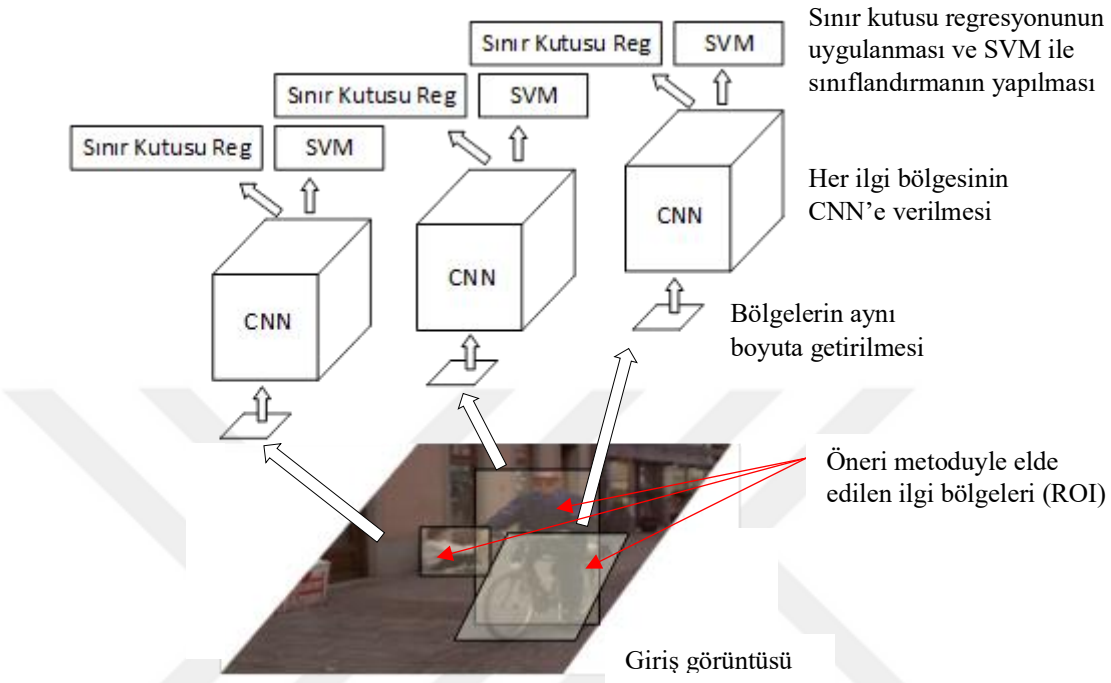
Şekil 3.6: Veri setinden örnekler.

3.2.2. CNN modelleri

Görüntü sınıflandırmada girdi olarak alınan resimdeki bir nesnenin hangi sınıfa ait olduğu tahmin edilmektedir. Nesnenin konumunu ve onun ait olduğu sınıfı birlikte tahmin etmeye nesne algılama denir. Sınıflandırmada çıkışta sadece bir nesneyi tahmin etme işlemi varken nesne algılamada çıkışta bir veya daha fazla nesne tahmin edilebilmektedir. Geleneksel algoritmalar, kısıtlamaları nedeniyle nesneleri verimli olarak tanıyamadığından nesneleri algılamak için R-CNN, R-FCN, YOLO ve SSD gibi son teknoloji ürünü derin öğrenme modelleri kullanılmaktadır.

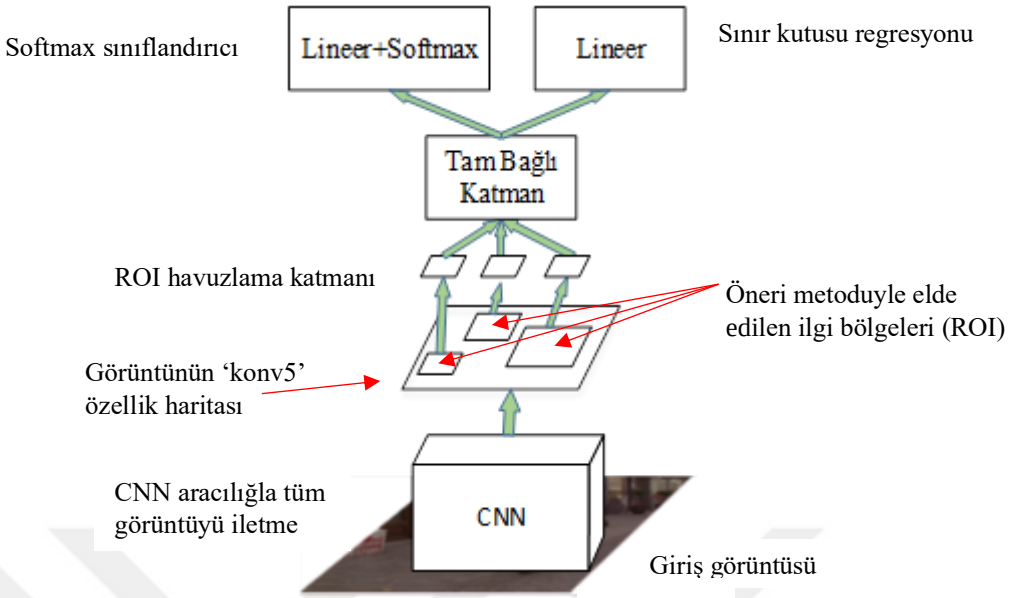
R-CNN algoritmaları çalışmaları sonucunda nesneleri sınıflarına ayırıp her bir nesneyi sınırlayıcı kutuyla döndürür. R-CNN, seçici aramayı kullanarak verilen görüntüden bir grup bölgeyi çıkarır ve ardından bu bölgelerden herhangi birinin bir nesne içerip içermediğini kontrol eder. Bu amaçla her bölgeden, belirli özellikleri çıkarmak için CNN

kullanılır. Daha sonra çıkarılan özellikler nesneleri tespit etmek için kullanılır. İşleyişte yer alan bu çoklu adımlar nedeniyle R-CNN oldukça yavaşlar.



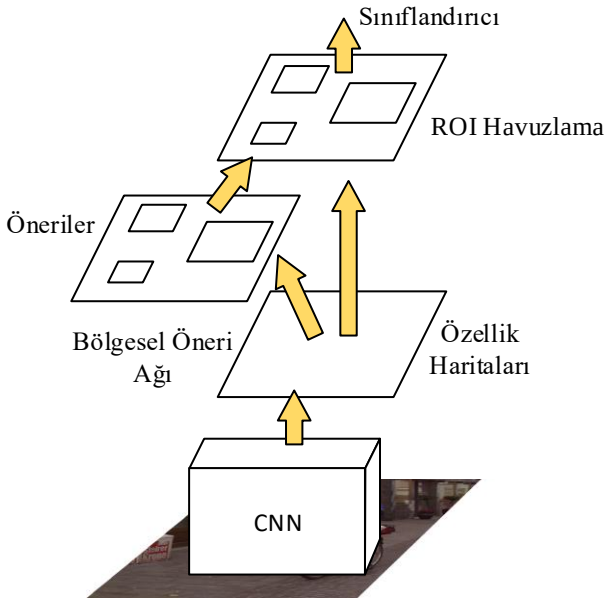
Şekil 3.7: R-CNN çalışma yapısı.

Fast R-CNN, başlangıç olarak görüntüyü ilgi bölgelerini oluşturan CNN'e verir. Üç farklı model kullanmak yerine bölgelerden özellikler çıkaran, bunları farklı sınıflara ayıran ve sınırlayıcı kutuları döndüren tek bir model kullanır. Tüm bu adımlar eş zamanlı yapılır, böylece R-CNN'e kıyasla daha hızlı çalışmayı sağlar. Sınıflandırma işleminde ise SVM yerine softmax sınıflayıcıyı kullanır. İlgi bölgesi (ROI) havuzlama katmanı her nesne için CNN'den elde ettiği özellik haritasından sabit uzunlukta bir özellik vektörü çıkarır ve bu özellik vektörlerini tam bağlı katmana aktarır. Fast R-CNN büyük bir veri setine uygulandığında, nesne öneri bölgelerini çıkarmak için seçici aramayı kullandığından yeterince hızlı değildir.



Şekil 3.8: Fast R-CNN çalışma yapısı.

Faster R-CNN, Fast R-CNN'deki yavaşlık sorununu seçici aramayı kaldırıp onun yerine bölge teklif ağı (RPN) kullanarak düzeltir. Önce CNN'i kullanarak giriş görüntüsünden özellik haritaları çıkarır ve daha sonra bu haritaları nesne önerilerini döndüren bir RPN'den geçirir. Son olarak, bu haritalar sınıflandırılır ve sınırlayıcı kutular tahmin edilir.



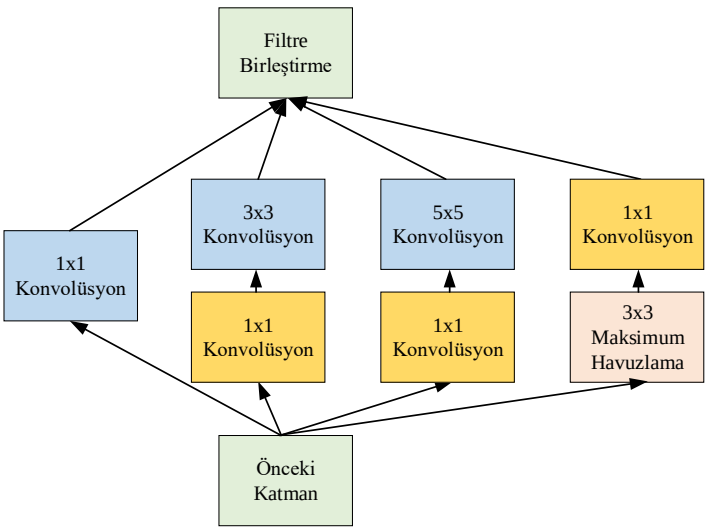
Şekil 3.9: Faster R-CNN çalışma yapısı.

Bir görüntüdeki nesneleri algılamak için Faster R-CNN algoritmasında 4 adım uygulanır;

a) Giriş görüntüsü özellik haritalarını döndüren CNN'e iletilir.
b) Bu özellik haritalarına bölge teklif ağı uygulanır ve nesne önerileri alınır.
c) Tüm önerileri aynı boyuta getirmek için ilgi bölgesi (ROI) havuzlama katmanı uygulanılır.
d) Aynı boyuta getirilen öneriler, görüntü için sınırlayıcı kutuları sınıflandırmak ve tahmin etmek için tam bağlı katmana iletilir.

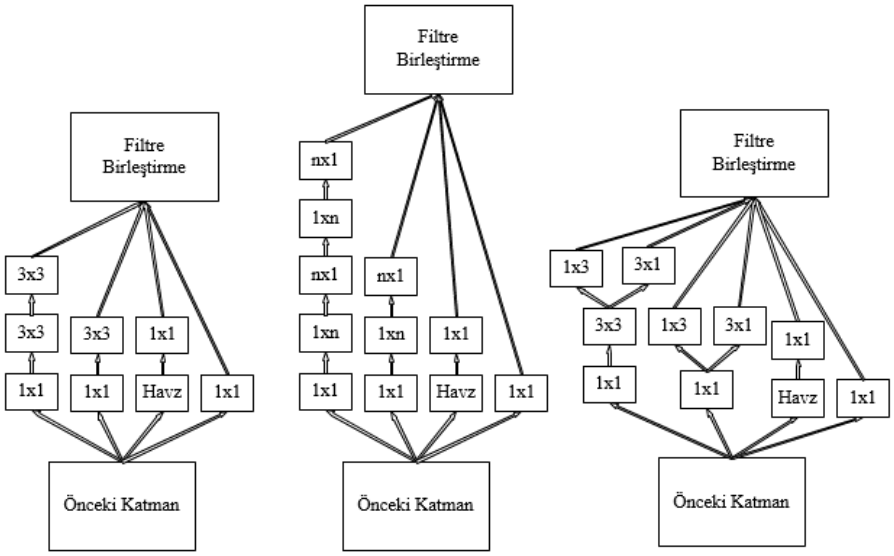
Faster R-CNN, otonom arabalarda, imalatta, güvenlikte vb. birçok alanda kullanılmaktadır.

Görüntü tanıma ve algılama algoritmalarının kıyaslanması için tanınmış bir platform olan ImageNet Görsel Tanıma Yarışması'nda Google ekibi 2014 yılında evrişimsel sinir ağlarının hesaplama karmaşıklığını azaltmada ve performansı artırmada çığır açan Inception(başlangıç) modülünü ortaya koydu. Bir giriş görüntüsü için, inception modülü paralel olarak birden fazla konvolüsyon işlemi gerçekleştirir. Sonra onların çıktıları tek bir çıktı vektöründe birleştirilir. Farklı ölçeklerde bilgiler çıkarmak için 1x1, 3x3 ve 5x5 gibi farklı filtre boyutları kullanılır. Boyutsallığı azaltmak için 1x1 konvolüsyonlar kullanılarak hesaplama karmaşıklığı azaltılır.



Şekil 3.10: Inception V1 [58].

Inception V2 modülü üç farklı tip modüle sahiptir. İlk modül (A), 5x5 konvolüsyonu 2 katmanlı 3x3 konvolüsyon olacak şekilde değiştirilerek, hesaplama yükü %28 azaltılmış ve performansı artırılmıştır [58].



a) A modülü b) B modülü c) C modülü

Şekil 3.11: Inception V2 mimarisi [58].

3x3 konvolüsyonu 3x1 ve 1x3 konvolüsyonlu alt katmanlara ayrıldığında %33 kazanç sağlanmıştır (B). Filtre genişletilerek daha yüksek boyutlu gösterimler ilkesinin bir ağ içinde yerel olarak işlenmesi daha kolaylaşmıştır ve burada modül genişletilmiştir (C) [58].

Tablo 3.2: Inception V2 mimarisi.

Tür	Parça boyutu/Adım sayısı	Giriş boyutu
konvolüsyon	3x3/2	299x299x3
konvolüsyon	3x3/1	149x149x32
konvolüsyon	3x3/1	147x147x32
havuzlama	3x3/2	147x147x64
konvolüsyon	3x3/1	73x73x64
konvolüsyon	3x3/2	71x71x80
konvolüsyon	3x3/1	35x35x192
3xbaşlangıç	A modülündeki gibi	35x35x288
5xbaşlangıç	B modülündeki gibi	17x17x768
2xbaşlangıç	C modülündeki gibi	8x8x1280
havuzlama	8x8	8x8x2048
lineer	logits	1x1x2048
softmax	sınıflandırıcı	1x1x1000

Transfer öğrenimi, daha önce öğrenilmiş olan ilgili bir görevden bilginin transferi yoluyla yeni bir görevde öğrenmenin geliştirilmesidir. Bu süreçte ikinci görev modellenirken hızlı ilerleme veya gelişmiş performans sağlanır. Bu tez çalışmasında COCO veri seti ile önceden eğitilmiş Faster R-CNN Inception V2, SSD Inception V2, Faster R-CNN Resnet 50 ve Faster R-CNN Resnet 101 modelleri kullanılmıştır. COCO veri seti günlük sahnelerden çekilmiş 328 bin resim içermektedir ve görüntüleri insan, kitap, araba, vazo, trafik işareti gibi 91 nesne kategorisine ayrılır.

3.2.3. CNN modellerinin eğitilmesi

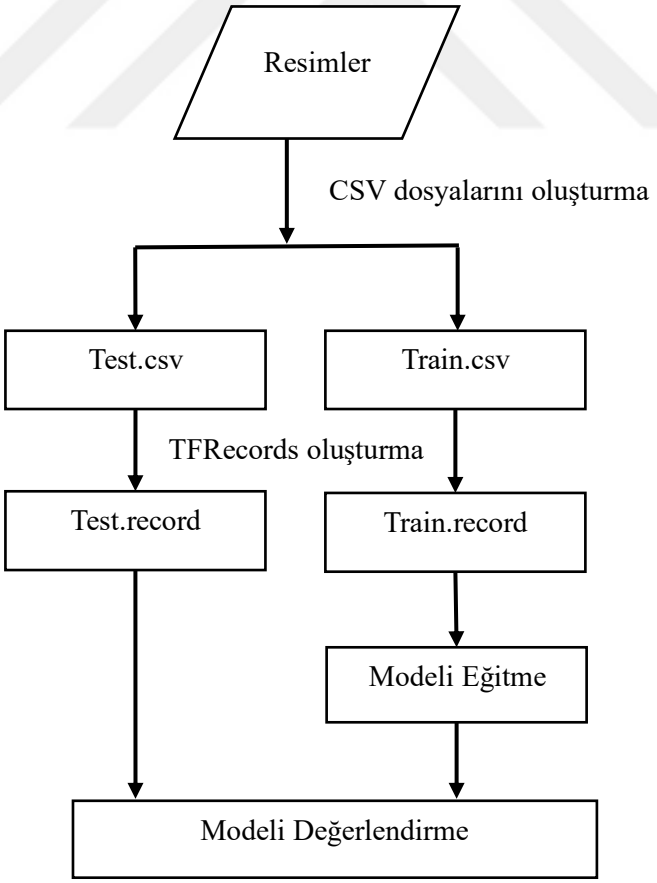
Google araştırmacıları tarafından oluşturulan TensorFlow, derin öğrenme kütüphaneleri içinde en popüler olanıdır. Derin öğrenme alanında, sinir ağları muazzam bir başarı elde etmiş ve çeşitli alanlarda geniş popülerlik kazanmıştır. Bu tez çalışmasında kullanılan görüntüler günlük sahnede çekilmiştir ve bir görüntü içerisinde birden fazla nesne barındırabilmektedir. Tek bir görüntüdeki birden fazla nesnenin yerini bulma ve tanımlama yeteneğine sahip doğru makine öğrenme modelleri oluşturmak, bilgisayarlı görmede temel bir zorluktur. TensorFlow Nesne Algılama API'si, TensorFlow'un üzerine inşa edilmiş, araştırmacılara ve geliştiricilere kendi özel görüntü sınıflandırıcılarını ve nesne algılama modellerini oluşturmaya imkân sağlayan güçlü bir araçtır. Makine öğrenmesi ve derin öğrenme çalışmalarını gerçekleştirmek için önceden eğitilmiş modelleri kullanmayı sağlar. Bu modellerde, modelin ilgilenilen nesneler için ne kadar hassas olduğunu ve modelin sahte algılamaların üstesinden gelmede ne kadar iyi olduğunu belirten MAP (ortalama ortalama hassasiyet) değeri vardır. MAP değeri arttıkça, modeller daha fazla doğruluk gösterme eğiliminde olur.

En düşük işlem hızında en yüksek mAP değerini veren ideal model bulunmalıdır. Tablo 3.3 incelendiğinde SSD MobileNet işlem hızı olarak daha iyi iken mAP değeri diğer modellere oranla daha düşüktür. Faster R-CNN Resnet 101 ise mAP değeri olarak iyi değere sahipken hız açısından daha yavaştır. Yapılan literatür çalışmaları sonucunda görüntü sınıflandırmada başarılarını kanıtlamış önceden eğitilmiş Faster R-CNN Inception V2, SSD Inception V2, Faster R-CNN Resnet 50 ve Faster R-CNN Resnet 101 modellerinin kullanılması uygun görülmüştür.

Tablo 3.3: Tensorflow kütüphanesinde COCO veri seti ile eğitilmiş bazı modeller [59].

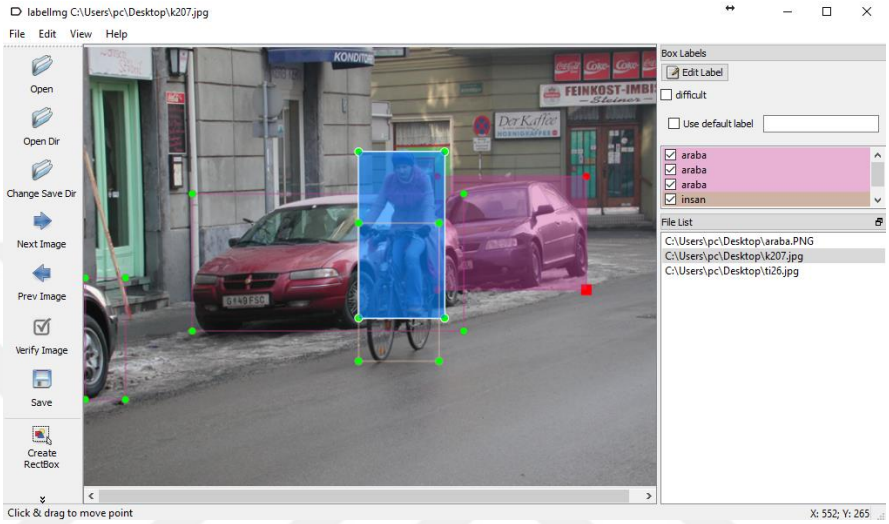
Model İsmi	İşlem Hızı (ms)	COCO mAP	Çıkışlar
ssd_mobilenet_v1_coco	30	21	kutular
ssd_inception_v2_coco	42	24	kutular
faster_rcnn_inception_v2_coco	54	28	kutular
faster_rcnn_resnet50_coco	89	30	kutular
faster_rcnn_resnet101_coco	106	32	kutular
mask_rcnn_inception_v2_coco	79	25	maskeler

TensorFlow Nesne Algılama API kullanılarak tekrardan eğitilen modellerin 10 nesneyi algılamadaki performansları değerlendirilmiştir ve bu çalışma için hem hız açısından hem de doğru tespitler gerçekleştirme açısından en iyi model seçilmeye çalışılmıştır. Modellerin eğitimi Intel Core i7 CPU işlemciye sahip bilgisayar üzerinde gerçekleştirilmiştir.

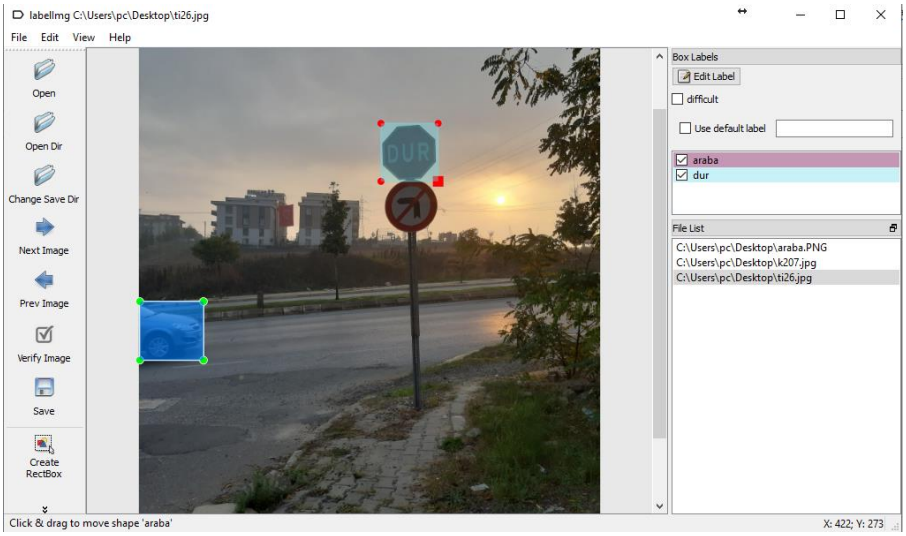


Şekil 3.12: Modeli eğitmede kullanılan temel akış diyagramı.

Modeli eğitmek için uygulanması gereken temel adımlar Şekil 3.12'de gösterildiği gibidir. Görüntülerden CSV dosyaları oluşturma adımında; veri setindeki tüm görüntülerin manuel olarak etiketlenmesi gerekmektedir. LabelImg programı kullanılarak her bir görüntüde bulunan nesneyi içine alacak şekilde bir kutu çizilip o nesnenin ismi belirtilerek etiketleme işlemi yapılmıştır. LabelImg etiket dosyalarını (.xml) popüler Pascal VOC biçiminde kaydeder. Görüntü etiketleme Şekil 3.13'te gösterilmiştir.



a) İnsan ve arabaların etiketlenmesi



b) Araba ve trafik işaretinin etiketlenmesi

Şekil 3.13: Görüntü etiketleme

Resimlerdeki nesneler etiketlenerek xml formatında dosyalar elde edilir. Xml uzantılı dosyalar daha sonra her bir resmin adının, yüksekliğinin, genişliğinin ve her bir nesnenin adının, nesnenin etrafına çizilen kutunun sol üst köşesinin (xmin,ymin), sağ alt köşesinin

(xmax,ymax) koordinat değerleri bilgilerinin yer aldığı CSV dosyalarına çevrilir. CSV her değerin virgülle ayrıldığı veri satırları içeren basit metin dosyalarıdır.

Tablo 3.4: Görüntüdeki nesnelerin bilgilerini içeren CSV dosyası.

Dosya adı	Genişlik	Yükseklik	Sınıf	xmin	ymin	xmax	ymax
a10.jpg	640	480	araba	1	82	121	209
k11.jpg	640	480	araba	114	212	218	295
k11.jpg	640	480	insan	286	191	323	299
k138.jpg	640	480	araba	259	159	318	191
k138.jpg	640	480	insan	341	148	359	185
k138.jpg	640	480	insan	448	164	478	239
k138.jpg	640	480	bisiklet	322	259	377	356
ti10.jpg	637	849	yol ver	56	385	73	400
ti10.jpg	637	849	doner kavsak	516	399	523	405
ti102.jpg	637	849	yaya gecidi	150	391	160	401
ti111.jpg	637	849	30 hiz siniri	316	288	383	359
ti143.jpg	637	849	20 hiz siniri	320	309	366	362
ti143.jpg	637	849	kavisli yol	316	259	368	306
ti165.jpg	1000	750	dur	177	47	809	694
k106.jpg	640	480	insan	247	45	415	401
k106.jpg	640	480	bisiklet	89	159	532	407

TFRecords oluşturma adımında; Tensorflow'un modeli eğitmek için gerekli olan kayıt dosyaları (TFRecords) CSV dosyalarından elde edilir. TFRecord, Tensorflow için tasarlanmış önemli bir veri formatıdır ve sıralı yapılandırılmış verileri ikili dizilerde tutar.

Modeli eğitme adımında; toplamda 517 görüntüden oluşan veri seti kullanılıp modeller eğitilerek 10 sınıftan oluşan nesne algılama çalışması yapılmıştır. Modeller sıfırdan eğitilmeye çalışılırsa veri eksikliğinden kaynaklı olarak iyi bir performans elde edilemeyebilir. Bu tez çalışmasında kullanılan modeller daha önce COCO veri setiyle eğitilmiştir. COCO veri setinde bulunan nesneler ile bu çalışmadaki nesneler benzerlik göstermektedir ve eğitilmiş modellerin eğitim sonucu elde ettiği özellikler, ağırlıklar bu çalışmadaki eğitimlerde kullanılmıştır. Önceden eğitilmiş modellerin son katmanı

kaldırılıp onun yerine yeni oluşturulan veri setinin sınıflarının sayısına karşılık gelen 10 sınıf sayısına sahip yeni bir katman eklenmiştir. Şekil 3.14'de gösterildiği gibi 10 sınıfın ismini ve ID değerlerini tanımlamak için pbtxt uzantılı etiket haritası oluşturulmuştur. Yeni katman daha sonra eğitimli modeller üzerinde tahmin etmek için eğitilebilir parametrelerle eğitilmiştir.

```
item {
  id: 1
  name: 'araba'
}

item {
  id: 2
  name: 'insan'
}

item {
  id: 3
  name: 'bisiklet'
}

item {
  id: 4
  name: 'dur'
}

item {
  id: 5
  name: 'yaya gecidi'
}

item {
  id: 6
  name: 'yol ver'
}

item {
  id: 7
  name: 'doner kavsak'
}

item {
  id: 8
  name: 'kavisli yol'
}

item {
  id: 9
  name: '30 hiz siniri'
}

item {
  id: 10
  name: '20 hiz siniri'
}
```

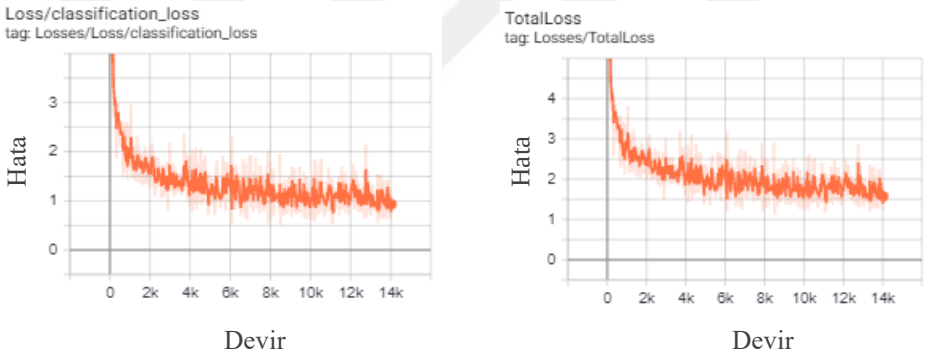
Şekil 3.14: 10 sınıfın isim ve ID bilgilerini tutan etiket haritası.

Çalışmada, veri bilimi vb. bilimsel uygulamalar için birçok paket programı içeren tümleşik bir python dağıtımı olan Anaconda kullanılmıştır. Anaconda üzerinde oluşturulan ortamda; Tensorflow, OpenCV, Pillow, Matplotlib ve Pandas kütüphaneleri kullanılmıştır.

Önceden eğitilmiş modellerin konfigürasyon dosyaları üzerinde değişiklikler yapılarak modeller yeniden eğitilmişlerdir. Her dört modelde sınıflandırıcı olarak softmax fonksiyonu kullanılarak nesne sınıflarının olasılıkları elde edilmiştir. Faster R-CNN Inception V2 modelinde optimizasyon yöntemi olarak 'momentum' kullanılmış, ilk öğrenim değeri (learning rate) 0.0002, batch değeri (batch size) 1 ve momentum değeri 0.9 olarak belirlenmiştir. Eğitim 14000 devir ile gerçekleştirilerek TensorBoard üzerinde elde edilen hata değerlerinin değişimi Şekil 3.15'te gösterilmiştir. Tensorflow tarafından sağlanılan TensorBoard; yapılan çalışmaların grafiklerini görselleştirmeye, grafiklerin çalışmasıyla ilgili nicel ölçümleri çizdirmeye yarayan görselleştirme aracıdır.

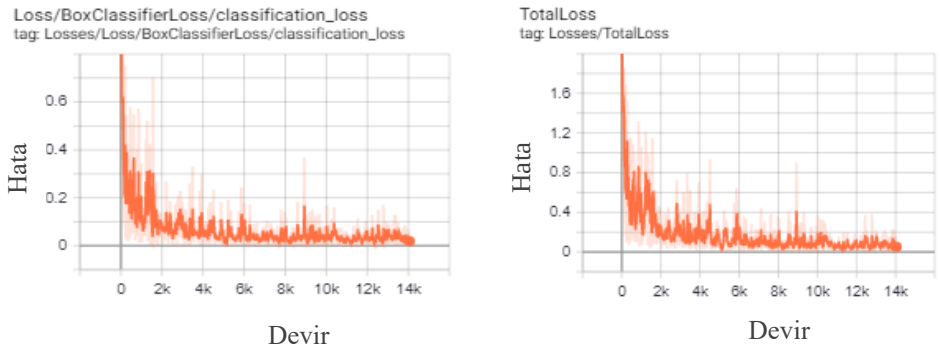
Şekil 3.15: Faster R-CNN Inception V2 modelinin 14000 eğitiminde hata değerinin değişimi.

SSD Inception V2 modelinin konfigürasyon dosyasında optimizasyon yöntemi olarak 'RMS Prob' kullanılmıştır. İlk öğrenme değeri 0.004, batch size değeri 24 ve momentum değeri 0.9 olarak belirlenmiştir. Bu model giriş görüntülerini 300x300 boyutuna getirerek işlemleri gerçekleştirir. Model 14000 devir (epoch) ile eğitilmiş ve eğitim sürecinde hata değerinin değişimi Şekil 3.16'da gösterilmiştir. Eğitim sürecinde devir sayısı artarken hata değerindeki azalım diğer modellere oranla yavaş olmaktadır.



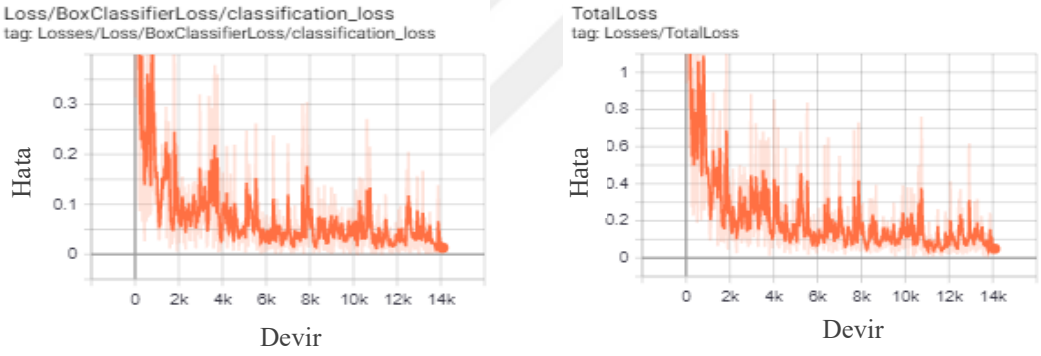
Şekil 3.16: SSD Inception V2 modelinin 14000 devir eğitiminde hata değerinin değişimi.

Optimizasyon yöntemi olarak 'momentum' kullanan ve ilk öğrenme değeri 0.0003, batch size değeri 1, momentum değeri 0.9 olarak belirlenen Faster R-CNN Resnet 101 modelinin 14000 devir sonucunda elde edilen hata değeri ve değişimi Şekil 3.17'de gösterilmiştir.



Şekil 3.17: Faster R –CNN Resnet 101 modelinin 14000 devir eğitiminde hata değerinin değişimi.

Özellik çıkarıcı olarak Resnet 50 kullanan Faster R-CNN Resnet 50 modelinin konfigürasyon dosyasında ilk öğrenme oranı 0.0003, batch size 1, momentum değeri 0.9 olarak belirlenmiş ve eğitim 14000 devirde gerçekleştirilmiştir. Optimizasyon yöntemi olarak 'momentum' kullanan modelin eğitim sürecinde hata değerindeki değişimler Şekil 3.18'de gösterilmiştir.



Şekil 3.18: Faster R-CNN Resnet 50 modelinin 14000 devir eğitiminde hata değerinin değişimi.

Modellerin eğitimi yapıldıktan sonra değerlendirme aşamasında 106 görüntüde bulunan nesne veya nesnelerin tespit etme başarımına ilişkin verilerin hesaplamasında, Tensorflow Şekil 3.19'da gösterilen karışıklık (confusion) matrisinden yararlanır.

Gerçek Pozitif (True Positive - TP): Gerçekte pozitif olan ve pozitif olarak sınıflandırılan nesneleri belirtir.

Yanlış Pozitif (False Positive - FP): Gerçekte negatif olan ve pozitif olarak sınıflandırılan nesneleri belirtir.

Gerçek Negatif (True Negative - TN): Gerçekte negatif olan ve negatif olarak sınıflandırılan nesneleri belirtir.

Yanlış Negatif (False Negative - FN): Gerçekte pozitif olan ve negatif olarak sınıflandırılan nesneleri belirtir.

		Tahmin Edilen Sınıf	
		Pozitif (P)	Negatif (N)
Gerçek Sınıf	Pozitif (P)	Gerçek Pozitif (TP)	Yanlış Negatif (FN)
	Negatif (N)	Yanlış Pozitif (FP)	Gerçek Negatif (TN)

Şekil 3.19: Modellerin değerlendirilmesinde kullanılan karışıklık matrisi.

Modeller değerlendirilirken karışıklık matrisinden yararlanılır ve bu matristen türetilen metrik değerleri kullanılır. Matristen elde edilen veriler kullanılarak elde edilen hassasiyet, özgünlük, doğruluk ve kesinlik gibi metrikler aşağıda belirtildiği gibi hesaplanmaktadır.

Doğruluk (accuracy):

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

Özgünlük (specificity):

$$\frac{TN}{TN+FP} \tag{3.2}$$

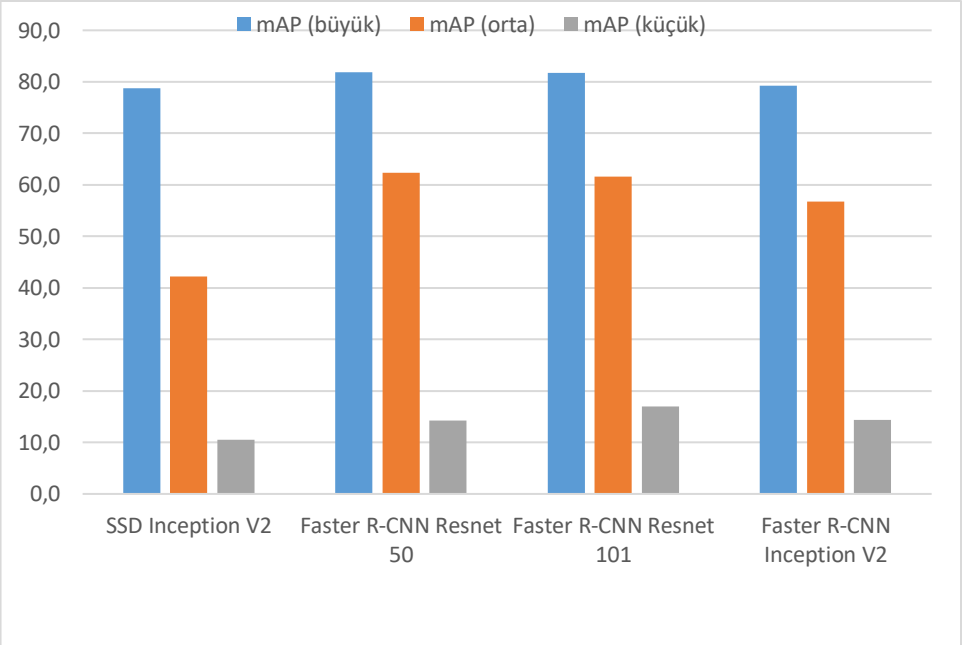
Hassasiyet (recall):

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

Kesinlik (precision):

$$\frac{TP}{TP+FP} \tag{3.4}$$

Kesinlik (precision) metriği ‘tahmin edilen örnekler arasında gerçekten kaç tanesi doğrudur’ sorusuna cevap verir. Hassasiyet (recall) metriği ‘pozitif sınıfa ait örneklerden kaç tanesi doğru tahmin edildi’ sorusuna cevap verir. Özgünlük metriği yanlış pozitif oranıdır. mAP (Mean Average Precision) nesne algılayıcıların doğruluğunu ölçen ölçüttür. mAP(small) 32x32 pikselden küçük olan nesneleri algılama başarısını, mAP(medium) 32x32 ile 96x96 piksel arasında olan nesneleri yakalama başarısını ve mAP(large) 96x96 ile 10000x10000 piksel arasında olan nesneleri yakalama başarısını belirtir.



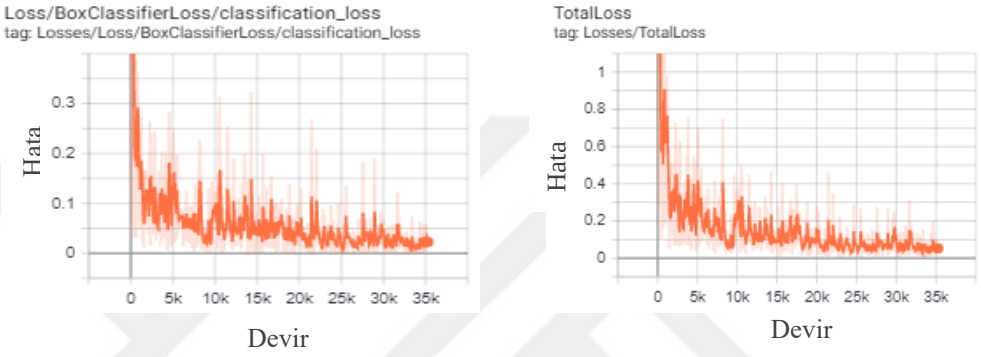
Şekil 3.20: 4 modellin mAP ile hesaplanan doğruluklarının karşılaştırılması.

14000 devirde eğitilen modellerin birbirlerine yakın değerlerde doğruluk değerleri verdikleri tespit edilmiştir ve Şekil 3.20 ‘de gösterilmiştir. Video görüntüleri üzerinde test edilen 4 modelden SSD Inception V2’nin nesne tespit etmede hızlı olmasına rağmen tespit ettiği nesne sayısının az olduğu gözlemlenmiştir. Daha iyi performans elde edebilmek için Faster R-CNN Inception V2, Faster R-CNN Resnet 50 ve Faster R-CNN Resnet 101 modelleri 35000 devirde yeniden eğitilmişlerdir. Yeniden eğitilen modellerin eğitim parametreleri Tablo 3.5’te belirtilmiştir.

Tablo 3.5: 35000 devirde eğitilecek modellerin parametreleri.

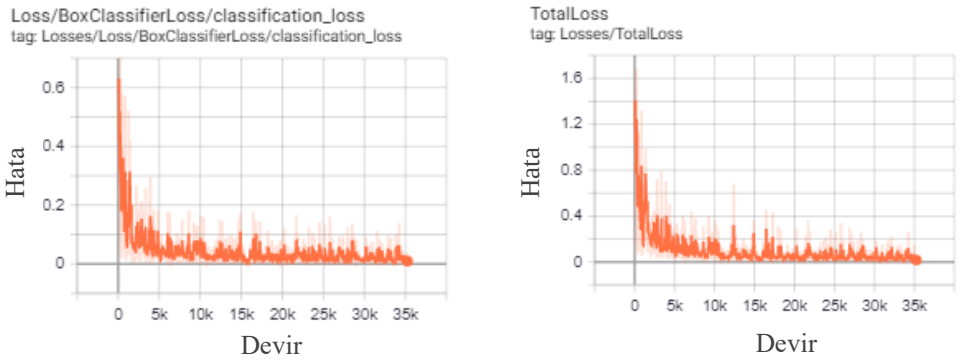
Model İsmi	Optimizasyon Yöntemi	Öğrenme Değeri	Batch Değeri	Momentum Değeri
Faster R-CNN Inception V2	Momentum	0.0002	1	0.9
Faster R-CNN Resnet 50	Momentum	0.0003	1	0.9
Faster R-CNN Resnet 101	Momentum	0.0003	1	0.9

Faster R-CNN Inception V2 modelinin 35000 devirde gerçekleşen eğitimindeki hata değişim grafiği Şekil 3.21'de gösterilmiştir.



Şekil 3.21: Faster R-CNN Inception V2 modelinin 35000 devir eğitiminde hata değerinin değişimi.

Faster R-CNN Resnet 101 modelinin 35000 devir eğitim sürecinde hata değerinin değişimi Şekil 3.22'de gösterilmiştir.

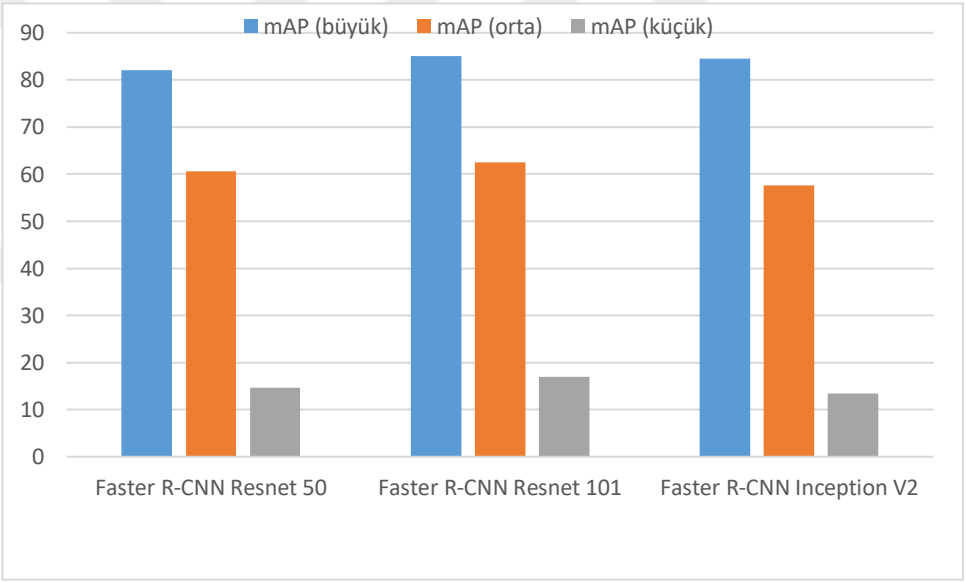


Şekil 3.22: Faster R-CNN Resnet 101 modelinin 35000 devir eğitiminde hata değerinin değişimi.

35000 devirde eğitilen Faster R-CNN Resnet 50 modelindeki hata değerinin değişimi Şekil 3.23'te gösterilmiştir.

Şekil 3.23: Faster R-CNN Resnet 50 modelinin 35000 devir eğitiminde hata değerinin değişimi.

35000 devirde gerçekleştirilen eğitimde performansta iyileşme gösteren modellerden en yüksek doğruluk değerini Faster R-CNN Resnet 101 göstermiştir. Şekil 3.24'de 35000 devirde eğitilen modellerin doğruluk değerlerinin karşılaştırılması gösterilmiştir.



Şekil 3.24: 3 modelin mAP ile hesaplanan doğruluklarının karşılaştırılması.

Faster R-CNN Resnet 101 modelinin hem görüntüler üzerinde hem de videolar üzerinden nesne tanıma işlemleri gerçekleştirilmiştir. Yapılan çalışmalarda araçtan yaklaşık 50 metre uzaktaki işaretler doğru bir şekilde algılanabilmiştir. Veri seti oluşturulduğunda nesnelerin bir kısmı kapatılmış durumdaki görüntüler de kullanılmıştır. Uygulama aşamasında bir kısmı kapatılmış nesneleri tanımada bazı durumlarda trafik işretlerinde sorun yaşanmıştır. Bu durumun ağı eğitmede kullanılan trafik işaretlerinin sayısının diğer nesnelere oranla az olmasından kaynaklı olduğu düşünülmüştür. Şerit algılama çalışmasında günün saatlerine göre değişen ışıklandırma koşulları altında şeritlerin

tespitinde farklılıklar yaşanmıştır. Görüntüde nesne tanıma çalışması Şekil 3.25 ’te, videoda nesne tanıma ise Şekil 3.26 ’da gösterilmektedir.

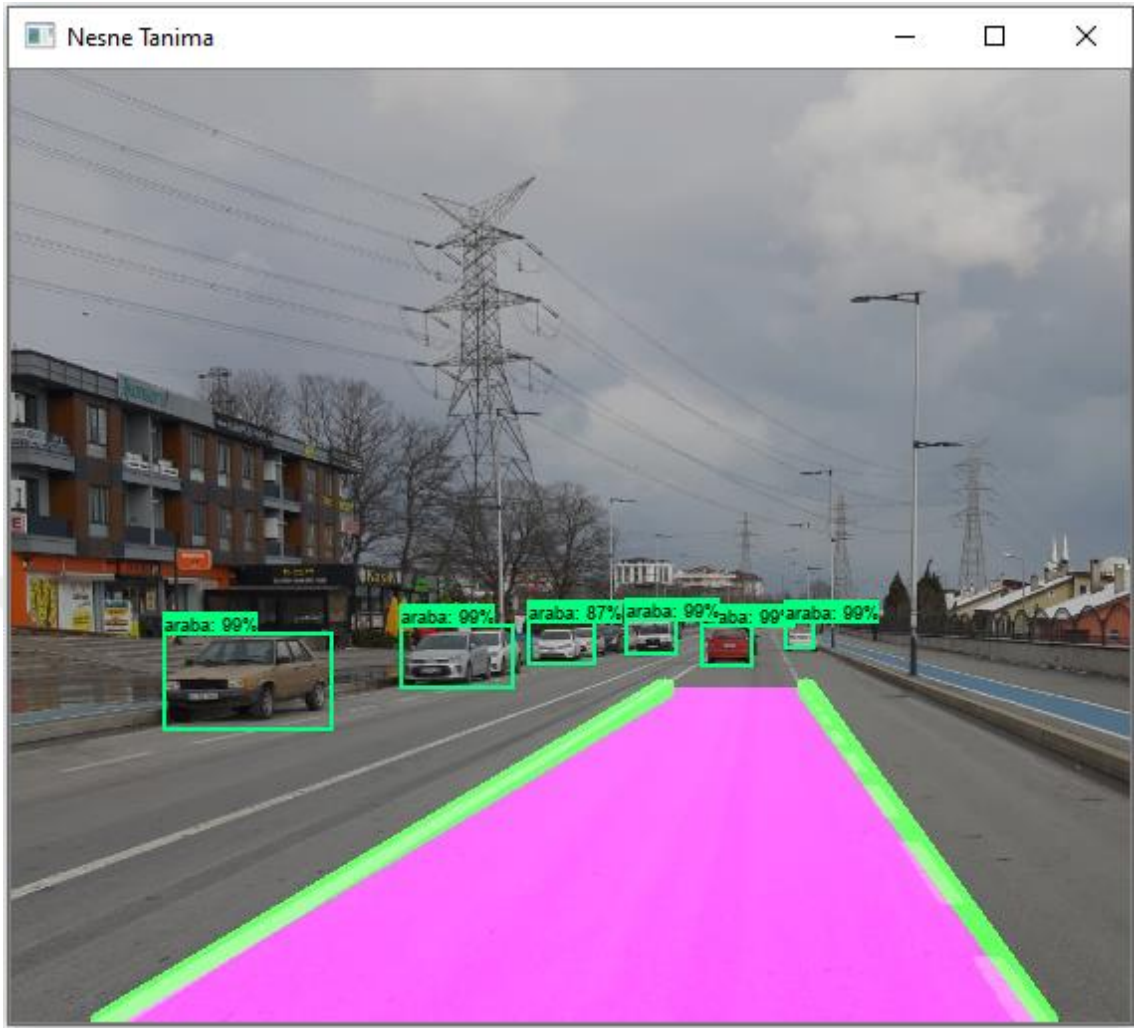
Şekil 3.25: Görüntü üzerinde nesne tanıma.

Şekil 3.26: Videoda nesne tanıma.

Şekil 3.27’de yağmurlu havada gerçekleştirilen nesne tanıma gösterilmiştir. Araba sileceğinin küçük boyutlu trafik işaretlerini kısmen kapatmasından dolayı kısa süreliğine tanıma işlemlerinde düşüş yaşanmıştır.

Şekil 3.27: Yağmurlu havada nesne tanıma.

Video görüntüleri üzerinde gerçekleştirilen nesne algılama ve şerit algılama çalışmaları birleştirilerek elde edilen çıktı Şekil 3.28’de gösterilmiştir. Çalışma, saatte 30-70 km hızları arasında giden bir taşıt içerisinden cep telefonu kamerasıyla kaydedilmiş video görüntüleri üzerinde test edilmiştir.



Şekil 3.28: Nesne ve şerit algılama.

BÖLÜM 4. SONUÇ VE ÖNERİLER

4.1. Sonuçlar

Bu tez çalışması kapsamında yoldaki nesneleri çeşitli koşullarda tanıyıp çevresindeki durumu sürücülere bildiren otonom arabaların bir parçası olabilecek bir uygulama gerçekleştirilmiştir. Arabanın sürüş esnasında içinde bulunduğu şeritleri algılamasını sağlamak amacıyla kameradan alınan görüntülere görüntü işleme tekniklerinden Canny kenar algılama ve HOG dönüşümü uygulanarak algılama işlemi gerçekleştirilmiştir. Yapılan şerit çizgisi tespiti, çoğunlukla gündüz alınan yol görüntüleri üzerinde yapılmıştır ve şerit renkleri de görüntülerin çoğunda açıkça görülmektedir.

Bu tez çalışması kapsamında yapılan ikinci bir işlem ise nesne algılama olmuştur. Makine öğrenmesi yöntemlerinde nesneye ait özellikleri elde etmek için özellik çıkarma algoritmalarının uygulanması gerekmektedir. Özellik çıkarmada kullanılan geleneksel algoritmalar, kısıtlamaları nedeniyle nesneleri verimli olarak tanıyamadığından son zamanlarda doğruluk oranı ve çalışma hızı ile gündemde olan derin evrişimsel sinir ağları kullanılmıştır. Derin öğrenmede özellik çıkarma evrişimsel sinir ağlarının konvolüsyon katmanlarında gerçekleşmektedir. Çıkarılan özellikler kullanılarak sınıflandırma işlemi gerçekleştirilir. Sınıflandırmada iyi sonuçların elde edilmesi için çıkarılan özelliklerin kalitesi ve verimliliği önemlidir, bunu sağlamanın bir adımı da ağı çok veri ile beslemektir. Veri çokluğunun evrişimsel sinir ağlarının iyi performans göstermelerini sağladığından yola çıkılarak içerisinde bir veya daha fazla nesneyi bulunduran 517 görüntünün 10 nesneyi sınıflandırmada yetersiz olacağı kararına varılmış ve bu durumun önüne geçmek için transfer öğrenimi yöntemi kullanılmıştır. Transfer öğrenimi ile daha önce başka veri setiyle eğitilen modellerin elde ettiği özellikler ve ağırlıklar, yeni veri setinin eğitiminde kullanılarak eğitimin kısa sürede iyi sonuçlar elde etmesi sağlanmıştır. Microsoft COCO veri seti üzerinde eğitilmiş ve nesne algılama çalışmalarında başarılı sonuçlar vermiş Faster R-CNN Inception V2, SSD Inception V2, Faster R-CNN Resnet

50 ve Faster R-CNN Resnet 101 modelleri kullanılarak nesne algılamadaki doğruluk (mAP) ve hız değerlerinin karşılaştırılması yapılmıştır. Nesne algılamada kullanılan modellerde mAP değerinin yüksek olması o modelin daha fazla doğruluk gösterdiğini ifade etmektedir.

Modellerin son katmanları değiştirilerek insanları, arabaları, bisikletleri ve 7 trafik işaretini algılaması için GRAZ-01, GRAZ-02 ve dışarıdan yapılan çekimlerle elde edilen görüntüler ile oluşturulan veri seti üzerinde tekrardan eğitilmiştir. Eğitimde kullanılan veri setinin %20 'si test için kullanılmıştır. Trafik işaretleri olarak dur, yol ver, döner kavşak, yaya geçidi, kavisli yol, 20 hız sınırı ve 30 hız sınırı işaretleri kullanılmıştır.

Intel Core i7 CPU işlemciye sahip bilgisayarda; 14000 devirde gerçekleştirilen eğitimde modellerin birbirine yakın değerde doğrulukta oldukları tespit edilmiştir. Eğitilen modeller görüntü ve videoya uygulandığında SSD Inception V2 modelinin hızda iyi performansa sahip olmasına rağmen doğrulukta diğer modellerden geride kaldığı gözlemlenmiş ve özellikle küçük nesneleri tespit etmede çok zayıf kaldığı görülmüştür.

Faster R-CNN Resnet 50, Faster R-CNN Resnet 101 ve Faster R-CNN Inception V2 modellerinin 35000 devirde gerçekleştirilen eğitimlerinde doğruluk değeri olarak iyi sonucun Faster R-CNN Resnet 101 modelinin verdiği tespit edilmiştir. Yapılan eğitimler sonucunda genel olarak, dört modelin de büyük boyuta sahip nesnelerin tespitinde iyi sonuçlar verdiği gözlemlenmiştir (mAP > %75).

Bu tez çalışması kapsamında durağan veya hareket halinde olan nesneleri algılamada en iyi performansı %85,1 doğrulukla Faster R-CNN Resnet 101 modelinin sağladığı bulunmuştur. Hız açısından ise en iyi performansı SSD Inception V2 modelinin sağladığı tespit edilmiştir.

Otonom arabanın koordineli bir şekilde çalıştığı ADAS sistemlerinin en kısa zamanda en doğru kararları alabilmesi için nesneleri ve işaretleri tespit etmede iyi performans göstermesi önemli bir kriterdir ve bu nedenle hız ve doğrulukta dengeyi kuran modelin seçilmesi gerekir. Bu tez çalışmasında doğrulukta iyi performans elde eden model tespit edilmiştir. Fakat hız açısından daha iyi performansın elde edilebilmesi için CPU yerine resim veya video karelerini gerçek zamanlı olarak işleyen GPU 'nun kullanılması önerilir.

FPGA’lerin tasarımlarda esneklik sağlaması, paralel işlem yapabilme yeteneğinden dolayı hızlı olması ve çalışmalarda az kaynak kullanması özellikle görüntü işleme tekniklerinde tercih edilmesini sağlamaktadır. Yazılım tabanlı çalışan CNN modellerinin donanım tabanlı olarak yüksek işlem gücüne sahip FPGA’lerde gerçekleştirilmesi algılama işlemlerinde hız ve doğrulukta iyi sonuç sağlayabilir.

Gelecekteki çalışmalarda, nesneleri iyi algılamak veya sınıflandırmak için yeni çıkan modeller araştırılarak iyi çalıştığı kanıtlanmış olan derin öğrenme mimarileri kullanılabilir ve otonom arabanın engel tanıma gibi farklı görevleri de bu çalışmaya eklenebilir. Ağı besleyecek veri çoğaltılarak şerit algılama; nesne algılama ile bir bütün olarak derin evrişimsel sinir ağları aracılığıyla gerçekleştirilebilir.

KAYNAKLAR

[1] Jochem, T., Pomerleau, D., Kumar, B., & Armstrong, J. (1995, September). PANS: A portable navigation platform. In *Proceedings of the Intelligent Vehicles' 95. Symposium* (pp. 107-112). IEEE.

[2] Dickmanns, E. D. (2002, June). The development of machine vision for road vehicles in the last decade. In *Intelligent Vehicle Symposium, 2002. IEEE* (Vol. 1, pp. 268-281). IEEE.

[3] Taşhan, B., (2017). *Road Lane Detectıon System With Convolutional Neural Network.* Bahçeşehir University, Istanbul.

[4] Joseph, A., Vineetha, K. V., Kurup, D. G., & Mini, R. (2019, June). A Neural Network Based Overvoltage Prediction System for Long Cable Issue. In *International Conference on Intelligent Computing, Information and Control Systems* (pp. 357-364). Springer, Cham.

[5] Borkar, A., Hayes, M., Smith, M. T., & Pankanti, S. (2009, March). A layered approach to robust lane detection at night. In *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems* (pp. 51-57). IEEE.

[6] Ingale, P. V., & Bhagat, K. S. (2016). Comparative Study of Lane Detection Techniques. *International Journal on Recent and Innovation Trends in Computing and Communication*, *4*(5), 381-390.

[7] Mistry, V. H., & Makwana, R. (2014). Survey: Vision based road detection techniques. *Int. J. Comput. Sci. Inf. Technol*, *5*, 4741-4747.

[8] Pomerleau, D., & Jochem, T. (1996). Rapidly adapting machine vision for automated vehicle steering. *IEEE expert*, *11*(2), 19-27.

[9] Bertozzi, M., & Broggi, A. (1998). GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE transactions on image processing*, *7*(1), 62-81.

[10] Wang, Y., Teoh, E. K., & Shen, D. (2004). Lane detection and tracking using B-Snake. *Image and Vision computing*, *22*(4), 269-280.

[11] Jung, C. R., & Kelber, C. R. (2005). Lane following and lane departure using a linear-parabolic model. *Image and Vision Computing*, *23*(13), 1192-1202.

[12] Kim, Z. (2008). Robust lane detection and tracking in challenging scenarios.

[13] Khalifa, O. O., Hashim, A. H. A., & Assidiq, A. A. (2009, September). Vision-based lane detection for autonomous artificial intelligent vehicles. In *2009 IEEE International Conference on Semantic Computing* (pp. 636-641). IEEE.

[14] Măriut, F., Foşalău, C., & Petrisor, D. (2012, October). Lane mark detection using Hough Transform. In *2012 International Conference and Exposition on Electrical and Power Engineering* (pp. 871-875). IEEE.

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

[16] Arcos-Garcia, A., Alvarez-Garcia, J. A., & Soria-Morillo, L. M. (2018). Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing*, *316*, 332-344.

[17] Wali, S. B., Hannan, M. A., Hussain, A., & Samad, S. A. (2015). An automatic traffic sign detection and recognition system based on colour segmentation, shape matching, and svm. *Mathematical Problems in Engineering*, *2015*.

[18] Paclık, P., Novovičová, J., Pudil, P., & Somol, P. (2000). Road sign classification using Laplace kernel classifier. *Pattern Recognition Letters*, *21*(13-14), 1165-1173.

[19] Maldonado-Bascón, S., Lafuente-Arroyo, S., Gil-Jimenez, P., Gómez-Moreno, H., & López-Ferreras, F. (2007). Road-sign detection and recognition based on support vector machines. *IEEE transactions on intelligent transportation systems*, *8*(2), 264-278.

[20] Aghdam, H. H., Heravi, E. J., & Puig, D. (2016). A practical approach for detection and classification of traffic signs using convolutional neural networks. *Robotics and autonomous systems*, *84*, 97-112.

[21] Fleyeh, H., & Davami, E. (2011). Eigen-based traffic sign recognition. *IET Intelligent Transport Systems*, *5*(3), 190-196.

[22] CireşAn, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural networks*, *32*, 333-338.

[23] Wang, G., Ren, G., Wu, Z., Zhao, Y., & Jiang, L. (2013, August). A hierarchical method for traffic sign classification with support vector machines. In *The 2013 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-6). IEEE.

[24] Shustanov, A., & Yakimov, P. (2017). CNN design for real-time traffic sign recognition. *Procedia engineering*, *201*, 718-725.

[25]Url-1< *https://towardsdatascience.com/mccullochpitts-model-5fdf65ac5dd1*>, erişim tarihi 16.04.2019

[26] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133.

[27] Minsky, M., & Papert, S. (1969). Perceptrons Cambridge. *MA: MIT*

[28] Demir, U., & Unal, G. (2017). Deep Stacked Networks with Residual Polishing for Image Inpainting. *arXiv preprint arXiv:1801.00289*.

[29] Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). San Francisco, CA, USA:: Determination press.

[30] Özkan, İ. N. İ. K., & Ülker, E. Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri. *Gaziosmanpaşa Bilimsel Araştırma Dergisi*, *6*(3), 85-104.

[31] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798-1828.

[32] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

[33] Url-2 <*https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*>, erişim tarihi 07.12.2019

[34] Mercan, C. A., & Celebi, M. S. (2013). An approach for chest tube detection in chest radiographs. *IET Image Processing*, *8*(2), 122-129.

[35] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[36] 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.

[37] Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

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

[39] 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).

[40] 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).

[41] Xie, S., Girshick, R. B., Dollár, P., Tu, Z., & He, K. (2016). Aggregated residual transformations for deep neural networks. CoRR abs/1611.05431 (2016).

[42] Url-3 <*http://cs231n.github.io/convolutional-networks*>, erişm tarihi 10.12.2019.

[43] Shetty, J., & Jogi, P. S. (2018, May). Study on Different Region-Based Object Detection Models Applied to Live Video Stream and Images Using Deep Learning. In *International Conference on ISMAC in Computational Vision and Bio-Engineering* (pp. 51-60). Springer, Cham.

[44] 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.

[45] 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).

[46] 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).

[47] 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.

[48] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

[49] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[50] Çörekcioğlu, B., (2017). *Visual Object Recognition and Detection Using Deep Learning*, Master's Thesis, Istanbul Technical University, Istanbul.

[51] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

[52] Kim, H., Lee, Y., Yim, B., Park, E., & Kim, H. (2016, October). On-road object detection using deep neural network. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)* (pp. 1-4). IEEE.

[53] Thrun, S., & Pratt, L. (1998). Learning to learn: Introduction and overview. In *Learning to learn* (pp. 3-17). Springer, Boston, MA.

[54] Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345-1359.

[55]Url-4<*https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a*> erişim tarihi 09.12.2019

[56] Şeker, A. (2018, September). Evaluation of Fabric Defect Detection Based on Transfer Learning with Pre-trained AlexNet. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)* (pp. 1-4). IEEE.

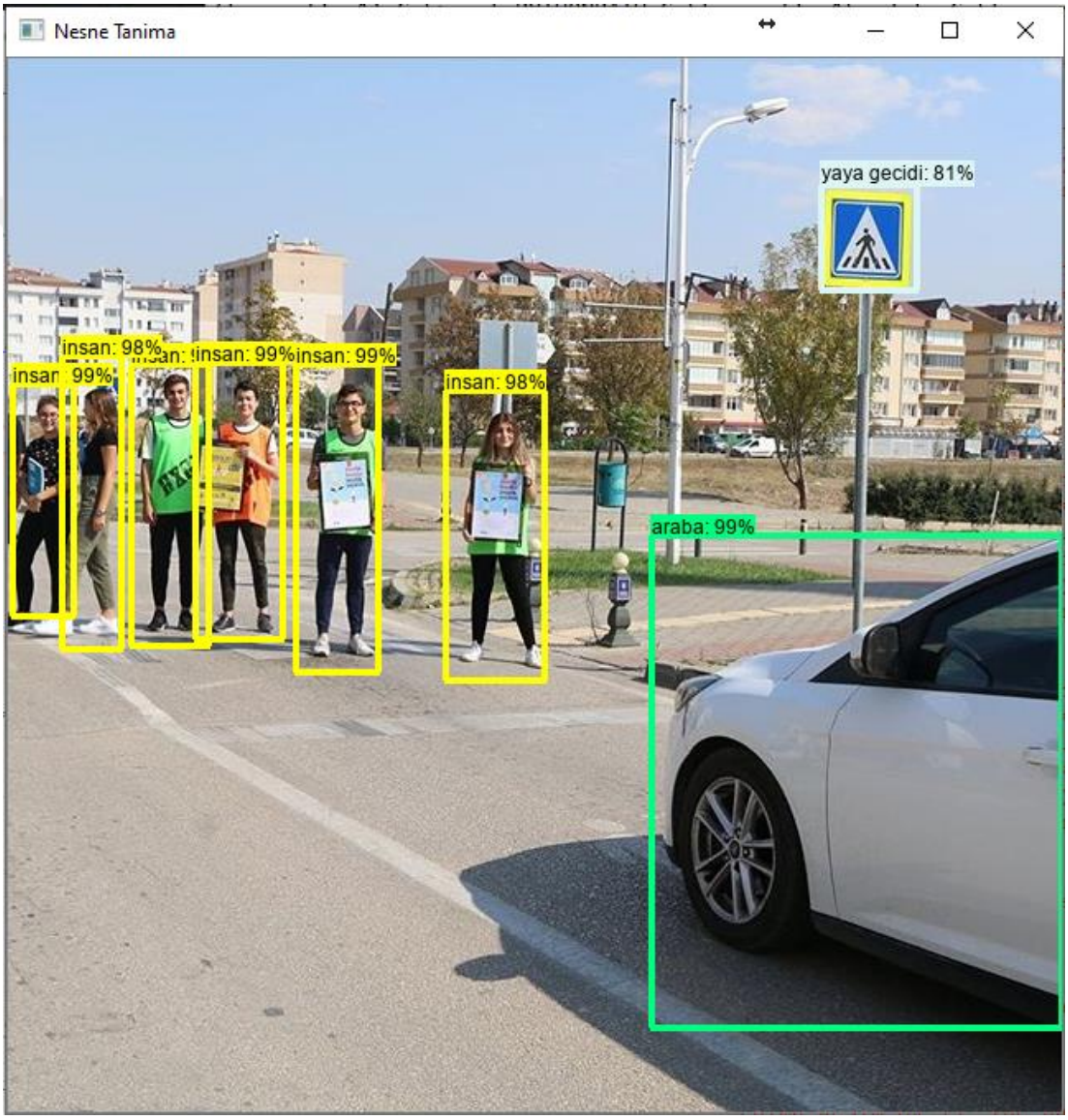
[57] Url-5 < *http://www-old.emt.tugraz.at/~pinz/data/*>, erişim tarihi 10.10.2019

[58] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

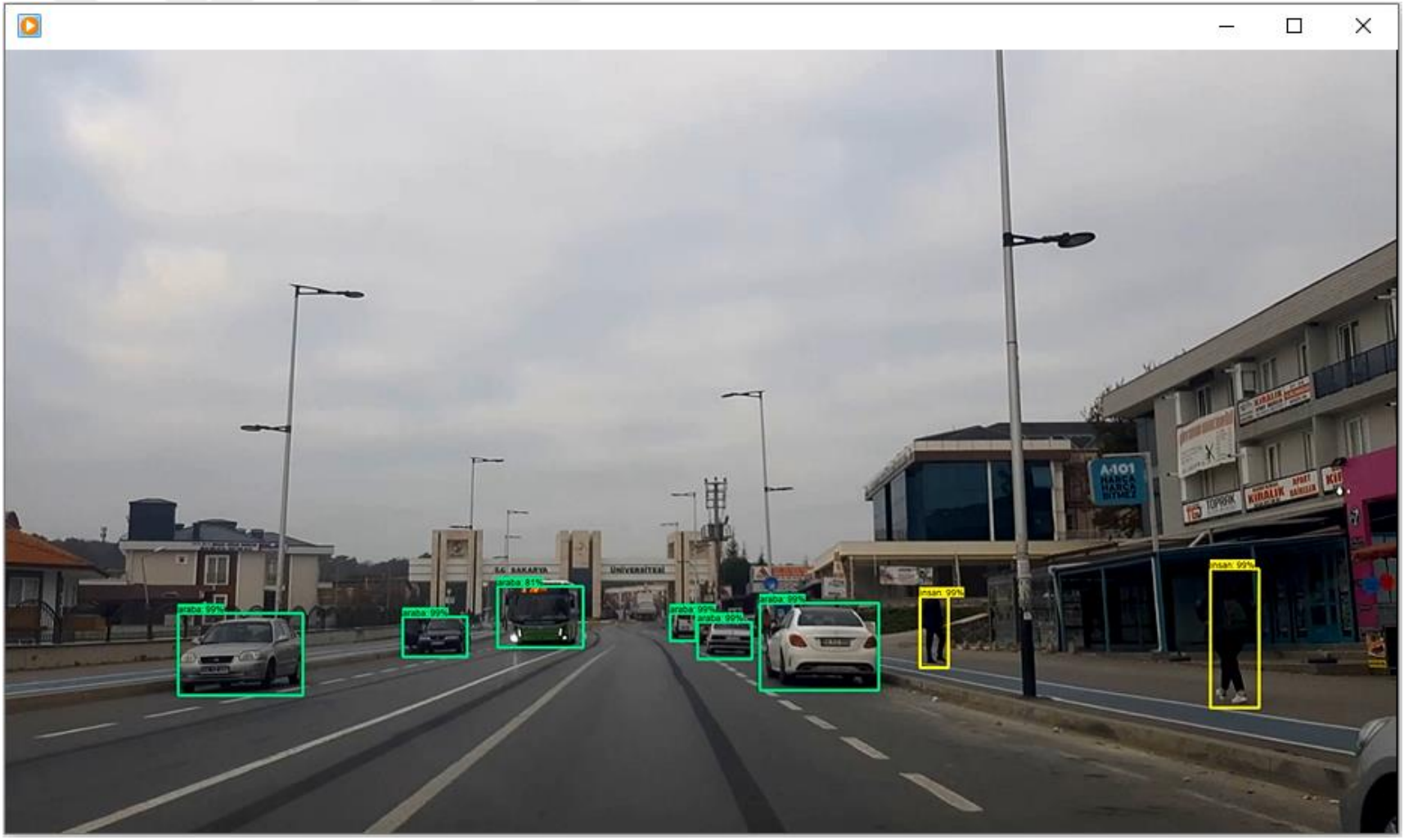
[59] Url-6 < *https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md* >, erişim tarihi 15.07.2019

EKLER

EK A: Görüntüde ve videoda nesne algılama



a) İnsan, araba ve trafik işareti algılama



b) Videoda insan ve araba algılama

c) Yağmurlu havada nesne algılama

d) Görüntüde nesne algılama

ÖZGEÇMİŞ

Ad-Soyad **: Gülyeter ÖZTÜRK**

Doğum Tarihi ve Yeri **: 04.06.1990**

E-posta **: gulyeter.ozturk@gmail.com**

ÖĞRENİM DURUMU:

- **Lisans** **:** 2015, Sakarya Üniversitesi, Teknoloji Fak., Mekatronik Müh.
- **Lisans** **:** 2016, Sakarya Üniversitesi, Bilgisayar ve Bilişim Bilimleri Fakültesi, Bilgisayar Mühendisliği (ÇAP)
- **Yüksek Lisans** **:** 2020, SUBÜ, Mekatronik Müh. ABD, Mekatronik Müh.

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2016-2017 yılları arasında Serdivan Halk Eğitim Merkezi'nde usta öğrenci olarak çalıştı.
- 2017-2018 yılları arasında Sakarya Yaz-Kar Klima ve Soğutma A.Ş.'de Mekatronik Mühendisi olarak çalıştı.
- 2018-2019 yılları arasında Nişantaşı Üniversitesi'nde Mekatronik Mühendisliği Bölümü'nde Araştırma Görevlisi olarak görev yaptı.
- 2019 yılından itibaren Sakarya Uygulamalı Bilimler Üniversitesi Mekatronik Mühendisliği Bölümü'nde Araştırma Görevlisi olarak görev yapmaktadır.