DERİN ÖĞRENME KULLANILARAK TRAFİK KOŞULLARINA UYGUN OTONOM ARAÇ UYGULAMASI

Mahmut Esat SEÇKİN

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME KULLANILARAK TRAFİK KOŞULLARINA UYGUN OTONOM ARAÇ UYGULAMASI

Mahmut Esat SEÇKİN
0000-0001-5045-8528

Prof. Dr. Ali SÜRMEN
(Danışman)

Doç. Dr. Cemal HANİLÇİ
(İkinci Danışman)

YÜKSEK LİSANS TEZİ
OTOMOTİV MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2021

TEZ ONAYI

Mahmut Esat SEÇKİN tarafından hazırlanan "DERİN ÖĞRENME KULLANILARAK TRAFİK KOŞULLARINA UYGUN OTONOM ARAÇ UYGULAMASI" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Otomotiv Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Prof. Dr. Ali SÜRMEN
Yard. Danışman : Doç. Dr. Cemal HANİLÇİ

Başkan : Prof. Dr. Ali SÜRMEN — İmza
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi,
Otomotiv Mühendisliği Anabilim Dalı

Üye : Dr. Öğr. Gör. Barış ERKUŞ — İmza
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi,
Otomotiv Mühendisliği Anabilim Dalı

Üye : Prof. Dr. Hakan GÜRKAN — İmza
Bursa Teknik Üniversitesi,
Mühendislik ve Doğa Bilimleri Fakültesi,
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
../../….

ÖZET

Yüksek Lisans Tezi

DERİN ÖĞRENME KULLANILARAK TRAFİK KOŞULLARINA UYGUN OTONOM ARAÇ UYGULAMASI

Mahmut Esat SEÇKİN

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Otomotiv Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ali SÜRMEN
İkinci Danışman: Doç. Dr. Cemal HANİLÇİ

Teknolojinin geliştiği ve firmaların ve küreselleşme çabası içerisinde yatırımlarını arttırdığı bir dönemde, yeni fikirlerin ortaya çıkması ve rekabetin artması kaçınılmaz hale gelmiştir. Zaman içerisinde cihazların donanım kapasitelerinin de artması sonucu birçok sektörde otonom sistem çalışmaları hız kazanmıştır. Makine öğrenmesi ve derin öğrenme alanındaki gelişmeler karmaşık problemlerin çözümüne ve birçok farklı uygulamanın geliştirilmesine olanak sağlamıştır.

Otonom araçlar, insan müdahalesi gerektirmeyen ve kendi kendine hareket kabiliyetine sahip olan araçlardır. Son yıllarda otonom araçlar, askeri, ticari ve araştırma projeleri olmak üzere hem akademik alanda, hem savunma sanayi alanında hem de özel sektörde kendine yer edinmiştir. Otonom araçların gelişimi ve kullanımının yaygınlaşması ile sürücü kaynaklı trafik kazalarının engellenebileceği, yakıt tasarrufunun artacağı ve yoğun trafik sıkışıklığı problemini ortadan kaldıracağı öngörülmektedir. Bu bağlamda tez çalışmasında, yapay sinir ağlarının temel çalışma prensipleri, fonksiyonları ve barındırdıkları değişkenler incelenmiştir. Ardından, 0'dan 9'a kadar olan sayılardan oluşan MNIST veri setine dayalı olarak bir evrişimli sinir ağı modeli oluşturulmuştur. Tüm ağ katmanları detaylıca incelenmiş ve modelin başarı sonuçları grafikler ile değerlendirilmiştir. Son olarak sürücüsüz bir aracın özerk hareketi için derin öğrenme modeli oluşturularak, bir sinir ağı eğitimi gerçekleştirilmiştir. Çalışma esnasında Torch kütüphanesi kullanılmış ve GPU üzerinden çalışan bir derin öğrenme uygulaması gerçekleştirilmiştir. Çalışma kapsamında araç kiti, Jetson Nano geliştirme kartı, Raspberry Pi kamera modülü ve ultrasonik sensör kullanılmış, otonom sürüş için de bir parkur hazırlanmıştır. Eğitilen sinir ağı modeliyle beraber aracın otonom bir şekilde parkuru tamamlaması sağlanmış ve çalışmanın sonuçları grafiklerle sunulmuştur.

Anahtar Kelimeler: Derin öğrenme, yapay sinir ağları, otonom araç, görüntü işleme
2021, xiii + 85 sayfa.

ABSTRACT

MSc Thesis

AUTONOMOUS VEHICLE APPLICATION SUITABLE FOR TRAFFIC CONDITIONS USING DEEP LEARNING

Mahmut Esat SEÇKİN

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Automotive Engineering

Supervisor: Prof. Dr. Ali SÜRMEN
Second Supervisor: Assoc. Prof. Dr. Cemal HANİLÇİ

In a period when technology develops and firms increase their investments in the effort of globalization, it has become inevitable for new ideas to emerge and competition to increase. As a result of the increase in the hardware capacities of the devices over time, autonomous system studies have accelerated in many sectors. Advances in machine learning and deep learning have enabled the solution of complex problems and the development of many different applications.

Autonomous vehicles are vehicles that do not require human intervention and have the ability to move on their own. In recent years, autonomous vehicles have gained a place in both the academic field, the defense industry and the private sector, including military, commercial and research projects. It is predicted that with the development and widespread use of autonomous vehicles, driver-related traffic accidents can be prevented, fuel savings will be achieved and traffic congestion will be eliminated. In this context, the basic working principles, functions and variables of artificial neural networks were examined in the thesis study. Then, a convolutional neural network model was created based on the MNIST dataset consisting of numbers from 0 to 9. While creating the model, all layers were examined in detail and the success results of the model were evaluated with graphics. Finally, a deep learning model was created for the autonomous movement of a driverless vehicle and neural network training was carried out. During the study, the Torch library was used and a deep learning application running on the GPU was implemented. Within the scope of the study, a car kit, Jetson Nano development board, Raspberry Pi camera modüle and ultrasonic distance sensor were used, and a track was prepared for autonomous driving. With the trained neural network model, the vehicle was provided to complete the track autonomously and the results of the study were presented with graphics.

Key words: Deep learning, artificial neural networks, autonomous vehicle, image processing
2021, xiii + 83 pages.

TEŞEKKÜR

Tez çalışmam boyunca, kendisine her danıştığımda değerli bilgiler edindiğim, problemlere karşı elinden gelenin fazlasını sunan ve imkanlarını seferber eden, desteğini ve samimiyetini hiçbir zaman esirgemeyen değerli danışmanım Prof. Dr. Ali SÜRMEN hocama, tez kapsamında kendi verdiği eğitimlerden ve uzmanlık alanlarından faydalanmama imkan sağlayan, çalışmamla alakalı önemli ölçüde gelişim göstermeme katkı veren ve oluşturduğum proje hakkında her danıştığımda beni aydınlatan kıymetli ikinci danışmanım Doç. Dr. Cemal HANİLÇİ hocama teşekkürlerimi sunarım. Son olarak, tezin başlangıcından bitişine kadar her zaman beni destekleyen, motive eden ve yanımda olup çalışmama destek olan aileme, diğer hocalarıma ve arkadaşlarıma teşekkürlerini sunarım.

Mahmut Esat SEÇKİN
31/08/2021

İÇİNDEKİLER

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
a	Öğrenme Katsayısı (Learning Rate)
b	Yanlılık - Bias
L	Maliyet Fonksiyonu
w	Weight
x	Bağımsız Beğişken
Z	Sigmoid Fonksiyonu

Kısaltmalar	Açıklama
ADAS	Advanced Driver Assistance Systems
ANN	Artificial Neural Network
API	Application Programming Interface
ASELSAN	Askerî Elektronik Sanayi
BRIEF	Binary Robust Independent Elementary Feature
BRISK	Binary Robust Invariant Scalable Keypoints
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DARPA	The Defense Advanced Research Projects Agency
DC	Direct Current
FCL	Fully Connected Layer
FREAK	Fast Retina Keypoint
GNSS	Global Navigation Satellite System
GPIO	General Purpose İnput/Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
HOG	Histogram of Gradient
HVNet	Hybrid Voxel Network
IBM	International Business Machines
IP	Internet Protocol Address
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
MNIST	Modified National Institute of Standards and Technology database
OICA	International Organization of Motor Vehicle Manufacturers
ORB	Oriented Fast and Rotated BRIEF
PID	Proportional Integral Derivative
RC	Radio Controlled
RF	Radio Frequency
SIFT	Scale Invariant Feature Transform
SURF	Speeded up Robust Feature
TÜİK	Türkiye İstatistik Kurumu

ŞEKİLLER DİZİNİ

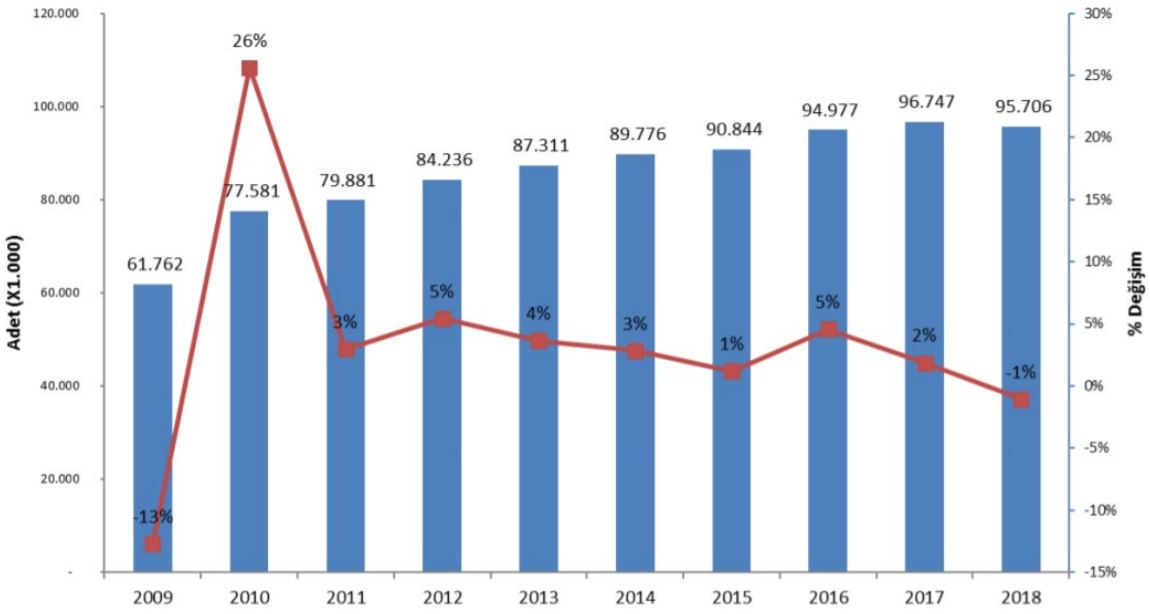
Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Teknolojinin geliştiği ve firmaların ve küreselleşme çabası içerisinde yatırımlarını arttırdığı bir dönemde, yeni fikirlerin ortaya atılması ve rekabetin artması kaçınılmaz hale gelmiştir. Firmaların bulundukları sektörde pazar paylarını arttırma hedefi içerisinde hareket etmeleri sonucu otomotiv sektöründe de birçok yeni yeniliğin de önü açılmıştır. Otomotiv sektörü aynı zamanda farklı sektörlerin ortak bir paydada buluşmasına ve sanayinin farklı alanlarının gelişmesine de katkı sağlamaktadır. Otomotiv sektöründe kullanılan demir-çelik, plastik, tekstil, kompozit gibi birçok sektör otomotiv pazarına ürün tedarik etmekte ve böylece otomotiv sektörü, kendisiyle ortak diğer sektörlerin gelişimini tetiklemektedir. Otomotiv ve bağlı bulunduğu diğer sektörler bazı dönemler geri çekilmeler yaşasa dahi her geçen yıl üretim adedini arttırmaktadır. Şekil 1.1'deki OICA verileri incelendiğinde, küresel çapta otomotiv üretiminin genel olarak artış gösterdiği görülmüştür.



Şekil 1. 1. 2009-2018 dünya otomotiv üretimi (OICA, 2019)

Sektörel çalışmalar incelendiğinde firmalar arasındaki rekabet vasıtası ile otonom/yarı otonom araçlar ve otomobillerdeki karar verme mekanizmaları ile ilgili çalışmaların sürekli olarak arttığı gözlemlenmektedir. Yarı-otonom sistemler son dönemlerde sıkça

otomotiv sektöründe kullanılmakta ve sürüş asistanı gibi kullanıcıya yardımcı teknolojiler şeklinde sunulmaktadır (Bayraktar, 2013). Aynı zamanda konfor anlayışının değişmesi, insan hayatına verilen önemin artması, trafik kazalarında yaşanan artışlar ve kaza nedenleri de yapılan çalışmalar üzerinde etkili olmuştur. Çizelge 1.1'de verildiği üzere, her yıl ne kadar trafik düzenlemeleri uygulanırsa uygulansın meydana gelen trafik kazalarının genel olarak yüksek olduğu ve bir azalış eğiliminde olmadığı görülmektedir.

Çizelge 1. 1. Yıllara göre trafik kaza istatistikleri (TÜİK, 2019)

Yıl	Toplam kaza sayısı	Ölümlü yaralanmalı kaza sayısı	Maddi hasarlı kaza sayısı	Ölü sayısı			Yaralı sayısı
				Toplam	Kaza yerinde	Kaza sonrası[1]	
2009	**1 053 346**	111 121	942 225	**4 324**	4 324	-	201 380
2010	**1 106 201**	116 804	989 397	**4 045**	4 045	-	211 496
2011	**1 228 928**	131 845	1 097 083	**3 835**	3 835	-	238 074
2012	**1 296 634**	153 552	1 143 082	**3 750**	3 750	-	268 079
2013	**1 207 354**	161 306	1 046 048	**3 685**	3 685	-	274 829
2014	**1 199 010**	168 512	1 030 498	**3 524**	3 524	-	285 059
2015	**1 313 359**	183 011	1 130 348	**7 530**	3 831	3 699	304 421
2016	**1 182 491**	185 128	997 363	**7 300**	3 493	3 807	303 812
2017	**1 202 716**	182 669	1 020 047	**7 427**	3 534	3 893	300 383
2018	**1 229 364**	186 532	1 042 832	**6 675**	3 368	3 307	307 071
2019	**1 168 144**	174 896	993 248	**5 473**	2 524	2 949	283 234

Aynı zamanda TÜİK verilerine göre trafik kazalarının sebepleri incelendiğinde; yolcu kusuru, yaya kusuru, yol kusuru ve araç kusuru gibi etkenler bulunsa dahi asıl büyük etkenin sürücü kusuru olduğu tespit edilmektedir. Bununla alakalı istatistikler Çizelge 1.2'de verilmiştir. Verilen istatistikler dikkate alındığında, araçların otonom hale gelmesi durumunda trafik kazalarının önemli ölçüde azalacağı öngörülmektedir.

Çizelge 1. 2. Trafik kazalarına neden olan kusurlar (TUİK, 2020)

Yıl	Sürücü Kusuru (%)	Yolcu Kusuru (%)	Yaya Kusuru (%)	Yol Kusuru (%)	Araç Kusuru (%)
2010	89,72	0,39	9,86	0,69	0,36
2011	90,2	0,39	8,51	0,6	0,3
2012	88,86	0,44	9,75	0,62	0,33
2013	88,69	0,42	8,99	1,05	0,85
2014	88,62	0,47	9,38	0,95	0,58
2015	89,3	0,43	8,8	0,91	0,55
2016	89,59	0,41	8,73	0,81	0,47
2017	89,87	0,37	8,48	0,7	0,52
2018	89,46	0,88	8,44	0,6	0,62
2019	88,00	1,30	8,20	0,5	0,20
2020	88,30	1,40	7,00	0,5	2,70

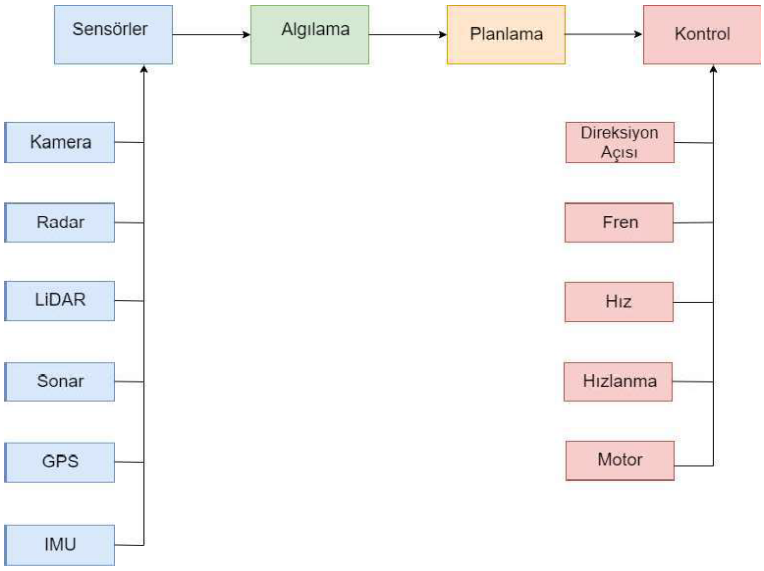
Çizelge 1.2'de bahsedilen durumlara ek olarak otonom araçlar önemli ölçüde yakıt tasarrufu sağlamaktadırlar. Bu sayede harcanan yakıtın azalması kullanıcıya ekonomik anlamda pozitif yansırken, beraberinde doğaya salınan sera gazları da azalmaktadır. Bunlar dışında, yapılan çalışmalara göre ilerleyen yıllarda tam otonom araçların yaygınlaşması sonucu nesnelerin interneti ile akıllı trafik uygulamaları da mümkün hale gelebilmekte ve bu sayede şehirlerin trafik sıkıntısının minimum düzeye indirilebileceği öngörülmektedir.

1.1. Otonom Araç

Robotik sistemler gün geçtikçe insan gücünün kaçınılmaz bir ihtiyaç olduğu düşünülen birçok sektörde kendine yer bulmuş ve bulmaya devam etmektedir. Bu sistemlerden biri de otonom araçlardır. Otonom araçlar, belli bir konumdan farklı bir konuma insan müdahalesine ihtiyaç duymadan güvenli bir biçimde seyir sağlayan ve kendisine verilen görevleri yerine getiren otonom sistemlerdir.

Otonom hareket sistemleri genel olarak dış dünyayı sisteme dâhil etmeyi sağlayan sensörler, sensör verileriyle edinilen bilgiler doğrultusunda gerçekleştirilecek eyleme karar veren bir denetleyici devresi ve otonom hareket eylemini yönlendiren kontrol yazılımından oluşmaktadır (Özgüner, Acarman, & Redmill, 2011). Otonom araçlar temel

olarak incelendiğinde bünyesinde dört adet ana modül barındırdığı tespit edilmektedir. Bunlar, sensör, algılama biçimi, planlama şekli ve kontrol mekanizmasıdır (Aki, Derin Öğrenme Tabanlı Sürücüsüz Araç Sistemleri, 2019). Bu modüller ve modülleri oluşturan etmenler Şekil 1.2'de diyagram halinde gösterilmiştir.



Şekil 1. 2. Sürücüsüz araç alt sistemleri (Aki, 2019)

Genel olarak dış dünyayı sisteme dahil etmeyi hedefleyen sensörler, büyük miktarlarda veri üretmekte ve üretilen bu veriler algılama modülüne iletilmektedir. Sensör verileri, algılama modülü içerisinde anlamlı birer bilgiye dönüştürülür. Ardından bu bilgiler aracın kendisine belirtilen hedef doğrultusunda davranış ve yol planlaması için kullanılır. Plan dahilinde oluşturulan yeni veriler kontrol modülüne aktarılır ve yeni veriler ışığında aracın yolu güvenli bir biçimde takip etmesi sağlanır.

Otonom araç çalışmaları ile bu araçların kullanımının yaygınlaşmasının sonucunda, otonom trafik akışı ile trafik planlama, park problemlerine yardımcı olma, sürüş özelliklerini geliştirme ve yakıt tasarrufu sağlayarak şehirlerdeki kirlenmeyi azaltmaya yardımcı olması öngörülmektedir. Gelişen bu uygulamaların birçok çevresel getiri sağlayacağı, şu ana kadar gerçekleştirilen projelerden anlaşılmakta ve gelişen teknoloji ile planlanan otonom araç tasarımlarının muhtemel faydaları tahmin edilebilmektedir (Aki, Derin Öğrenme Tabanlı Sürücüsüz Araç Sistemleri, 2019).

1.2. Otonom Aracın Gelişimi

Makine öğrenmesi ve derin öğrenme, otomotiv endüstrisinde özellikle sürücüsüz araçların geliştirilmesinde çok büyük etkiye sahiptir. Nesne algılama, şerit takibi, hız sınırlayıcı, yol planlaması ve haritalama gibi birçok özellik son yıllarda geliştirilen yarı otonom araçlarda dahi halihazırda kullanılmaktadır.

Otonom araçlar ile ilgili ilk fikirler 1939 yılında General Motors öncülüğünde "Futurama Dünya Fuarı"nda geleceği öngören bir yaklaşım ile ele alınmıştır. Fiili olarak sonuç alınan ilk çalışmalar ise 1980'li yıllarda Alman Bundeswehr Üniversitesi'nde gerçekleşmiştir. Çalışma, Ernst Dickmanns ve arkadaşları ile yapılmış olup Mercedes-Benz araç kullanılmıştır. Bu araca kamera ve sensörler eklenerek otonom bir şekilde 95 km/h hıza ulaşılması sağlanmıştır. İlk denemeler güvenlik nedeniyle trafiğe kapalı alanlarda gerçekleştirilmiştir (Özgüner, Acarman, & Redmill, 2011).

2000'li yıllardan itibaren sürücüsüz araç gelişimi oldukça hızlanmıştır. 2002 yılında DARPA ajansı tarafından "Grand Challenge" adı altında otonom araç yarışmaları düzenlenmeye başlanmıştır. Müsabakalarda yarışmacılardan, araçlarının belirlenmiş hedefe insan müdahalesi olmadan kendi kendine gitmesini sağlayabilecek bir sistem geliştirilmesi istenmiştir. Yarışmanın ilk yıllarında her ne kadar rota tamamlanamasa da, çalışmaların ilerlemesi ışığında 2005 yılında beş takım 217 km'lik yarışma rotasını tamamlamayı başarmıştır. 2007 yılında ise arazide yapılan DARPA "Grand Challenge" yarışması yerini DARPA "Urban Challenge"a bırakmıştır. Bu yarışmada ise araçların şehir içi sürüşlerinin geliştirilmesi hedeflenmiştir. Ortalama bir şehrin içerisindeki dinamikleri simule eden bir parkurda, otonom araçlar trafik kurallarına uyarak diğer araçlara, yayalara ve çarpılmaması gereken materyallere çarpmadan yaklaşık 100 km parkuru 6 saatin altında bitirmeye çalışmışlardır (Cardinal, 2011).

Şekil 1. 3. DARPA yarışmalarına katılan otonom bir araç (Wikipedia, 2021)

Bu alandaki çalışmalardan nitelikli sonuçlar elde edilmeye başlanması ile araştırma enstitüleri de bu alanda büyük miktarlarda finansal yatırımlar yapmıştır. Bu çalışmaların sektörel bazda ticarileşmesini sağlayacak diğer gelişmeler ise, yalnızca araştırma kurumlarının değil; birçok otomotiv ve teknoloji firmasının da otonom araç modelleri üzerinde çalışmalar yürütmesidir. Örneğin Google'ın geliştirmekte olduğu Waymo isimli aracı 2009 yılında geliştirilmeye başlanmış ve araç günümüze kadar gerçek yaşam alanlarında yaklaşık 32 milyon kilometre otonom sürüş gerçekleştirmiştir. Gerçekleştirilen bu sürüşler dünyanın pek çok yerinde test amaçlı olarak devam etmektedir (TheVerge, 2020).

Şekil 1. 4. Google'ın geliştirmiş olduğu otonom araç Waymo (Datta, 2021)

Yarı otonom/tam otonom ve elektrikli araçlar arasında yenilikleri ve girişimleri ile öne çıkan bir diğer araç firması Tesla, ürettiği otomobillerde otomatik fren, hızlanma ve şerit takibi ile aracı yönlendirebilen bir sürücü asistanı özelliğini halihazırda standart olarak kullanıcılarına sunmaktadır. Ayrıca Tesla otonom araç filosu gerçek yaşam alanlarında 5,3 milyar kilometreden fazla otonom sürüş gerçekleştirmiş, rakibi olan diğer otonom araçlardan çok daha büyük bir veri kapasitesine ulaşmıştır (Bouchard, 2019). Bu açıdan Tesla, filosuna tam otonom sürüş özelliği kazandırmak için tüm ortamlardan topladığı veriler ile araçlarını diğer yol şartlarına göre de geliştirmektedir. 2021 yılında, var olan özelliklerinin yanı sıra otonom park, şerit değiştirme ve tabela ile trafik işaretlerini algılama özelliklerini barındıran bir paketi de kullanıcılarına sunacaklarını bildirmişlerdir.

General Motors tarafından 2013 yılında kurulmuş olan Cruise otonom araç şirketi ise, kendi bünyesinde geliştirilen araçların 2017'den bu yana gerçek yaşam alanlarında sürüş testlerini gerçekleştirmektedir. Ford Motor Şirketi ve Volkswagen ortaklığıyla kurulan bir başka otonom araç şirketi ise Argo AI teknoloji şirketidir. 2016 yılında kurulmuş olan bu şirket yapmış olduğu otonom araç çalışmaları ile ortak firmalarının araçlarına otonom sürüş hizmeti sağlamayı amaçlamaktadır.

Şekil 1. 5. Cruise ve Argo şirketlerinin geliştirdiği otonom araçlar (Anonim, 2016)

Yine dünyanın önemli teknoloji şirketlerinden Rusya merkezli Yandex, 2017'den bu yana otonom araç çalışmaları gerçekleştirmektedir. Avrupa'daki ilk robotaksi faaliyetlerini başlatan bu firma, farklı markalardan oluşan araç filosu ile toplamda 6,4 milyon kilometre otonom sürüş gerçekleştirmiştir.

Şekil 1. 6. Yandex'in geliştirmiş olduğu Robotaksi (Yandex, 2017)

Otonom araç teknolojisinin gelişimi ile ilgili hükümetler de konu kapsamında yasalar çıkarmakta, düzenlemeler oluşturmakta ve oluşturmaya devam etmektedirler. Örneğin California'da 2012 yılından beri sürücüsüz araç çalışmalarının geliştirilmesi ve desteklenmesi adına otonom araçların trafiğe çıkmalarına izin verilmiştir.

Türkiye'de otonom araç gelişmeleri incelendiğinde ise genellikle askeri amaçlı çalışmalar yürüten ve insansız araçlar üzerinde çalışmaları da bulunan, başarı sağlamış bir kurum olarak ASELSAN örnek verilebilir. ASELSAN askeri amaçlı insansız kara, hava, deniz araçları geliştirmiş ve geliştirmeye devam etmektedir.

ASELSAN bünyesinde geliştirilen araçlar (Yumruktay, 2015);
- İZCİ (insansız kara aracı)
- GEZGİN (çok amaçlı insansız kara aracı)
- DENİZCİ (insansız su üstü aracı)
- AUV (insansız su altı aracı)
- KÂŞİF (balonlu keşif gözetleme sistemi)
- DALKILIÇ (çok maksatlı muharebe robotu) örnek olarak verilebilir.

Askeri amaçlı olarak üretilen bu araçlar, afet durumlarında insanların erişemeyeceği konumlara ulaşabilmeleri ile afetzedelerin tespiti ve kurtarılması sağlamakta; terör eylemlerinde konvansiyonel silahlarla ve bomba imha operasyonlarında insan faktörünü ortadan kaldırarak minimum riskle faaliyet gerçekleştirmektedirler. Şekil 1.7'de

ASELSAN bünyesinde üretilmiş ve Türk Silahlı Kuvvetleri envanterine yeni eklenmiş olan bir insansız kara aracı verilmiştir.

Şekil 1. 7. ASELSAN Kaplan insansız kara aracı (Yumruktay, 2015)

2. KURAMSAL TEMELLER VE KAYNAK ARAŞTIRMASI

Yapay zeka, ilk kez 1956 yılında Dartmouth Konferansı'nda Prof. John McCarthy tarafından kullanılmıştır. Temel olarak görevleri yerine getirmek için insan zekasını taklit eden ve topladıkları bilgilere göre yinelemeli olarak kendilerini iyileştirebilen sistem veya makineler anlamına gelir.

Yapay zeka çalışmalarının araştırma ve teknoloji bakımından dikkat çekici gelişmeleri 1990'lı yıllara dayanmaktadır. IBM şirketi bünyesinde geliştirilen "Deep Blue" adını verdikleri yapay zekanın 1997 yılında dünya satranç şampiyonu Garry Kasparov'u yenmesi araştırmacıları yapay zekaya bakış açısını değiştirmiştir. 2011 yılında ise yine IBM'in kendi bünyesinde geliştirmiş olduğu "Watson" adındaki yapay zekanın bir yarışma programında rakiplerini yenmesi, yapay zeka oluşumlarının yalnızca matematiksel konularda değil insanlarla mücadele ve yarışma konusunda da başarılı olduğunun bir ispatı olmuştur. 2016 yılında Google'ın yapmış olduğu çalışmada, oyunların insanlar tarafından nasıl oynandığını incelenmiş elde edilen veriler üzerinden "Google Deepmind" adında sürekli öğrenen bir yapay sinir ağı geliştirilmiştir. Yine "Google Deepmind" ile geliştirilen "AlphaGo" adlı program dünya Go şampiyonu Lee Sedol'ü yenmiştir. Böylece yapay zekanın satrançtan çok daha karışık oyunlarda da başarılı olabileceği kesinleşmiştir (Anonim, Yapay Zeka Nedir? Uygulama Alanları Nelerdir?, 2021).

Makine öğrenmesi (machine learning) ise, 1959 yılında bilgisayar biliminin yapay zekada sayısal öğrenme ve model tanıma çalışmalarından türetilmiş bir daldır. Temel olarak yapılan şeyşu şekilde tanımlanabilir. Matematiksel ve istatistiksel problem çözümleri ile veri kümeleri süzülerek karar sonuçları çıkarılmakta ve oluşturulan sistemler bilgisayar yazılımı üzerinden modellenmektedir. Yapısal işlevi, öğrenebilen ve veriler üzerinden hareket edebilen algoritmaların çalışma biçimlerini incelemek ve bu yapıları geliştirmek üzerine kuruludur.

Yapılan akademik çalışmalar incelendiğinde; Pomerleau D. A. (1989), ABD'de DARPA'nın destek verdiği otonom sürüş kabiliyetine sahip bir yer aracı olan ALVINN'i ilk insansız sürüş testine tabi tutmuştur. Aracın üzerinde dış kaynaktan veri alabilmesi için kamera ve radar mekanizmaları kullanılmıştır. Bu sayede elde edilen veriler aktarılarak algoritma içerisinde kullanılmış ve aracın hızı saatte 30 kilometreye ulaşmıştır. Sonraki yıllarda ise belirlenen bir arazide yalnızca sensör bazlı sürüş gerçekleştirmiştir. Gerçekleştirilen çalışma için seçilen parkur 600 metrelik bit yoldur. Yol boyunca farklı arazi koşulları bir arada bulundurulmuş ve bu sayede karmaşık bir güzergah elde edilmiştir. Ayrıca parkur kapsamında zor bir yol tutuşunun olması için farklı hava şartlarında da çalışmalar gerçekleştirilmiştir.

Dickmanns D. ve diğerleri (1995), Mercedes-Benz marka bir aracın S sınıfı modelini çalışmalarında kullanarak Münih'ten Kopenhag'a belirle bir rota üzerinde aracı götürüp getirmeyi başarmışlardır. Yaklaşık olarak 1600 kilometrelik mesafeye sahip olan bu güzergahta 175 kilometre/saatlik bir sürate çıkan araç, başarı oranı %95 olacak şekilde otonom bir sürüş gerçekleştirebilmiştir.

Bertozzi M. ve Broggi A. (1998), Parma Üniversitesi'nde otoban üzerindeki trafik şeritlerini takip edebilen ve hareket esnasında bu şeritler arasında kalabilen bir araç geliştirmişlerdir. Geliştirdikleri Lancia Thema markalı araç, ortalama 90 kilometre hız ile İtalya'daki 2000 kilometre mesafeye sahip yol, çalışma senasında toplamda 6 gün sürmüş ve sonunda parkur tamamlanmıştır. Araç, yolun %94'lük önemli bir kesimini insan müdahalesi olmadan tamamlamıştır.

Güner Ş. (2012), yaptığı çalışmasında sürücüsüz araçlarda kullanılması için bir sürat kontrolörü geliştirmiş ve geliştirilen kontrolör simülasyon ortamında test edilmiştir. Ardından kontrolörün yol testleri de gerçekleştirilerek somut bir çalışma ortaya konulmuştur. Oluşturulan sürat kontrolörünün ana fonksiyonu, elektrik sinyali şeklinde gerçekleştirilen referans hızında, parkurun dış faktörlerinden biri olan eğim gibi bozucu etkenlerin de incelenmesi suretiyle aracın hareket etmesi mümkün hale getirilmiştir. Matlab yazılımı ile, hız bilgileri değişken olarak girilmiş ve sürüşü bozucu olarak

nitelenen eğim gibi dış etkenlerle birlikte simule edilen sistem incelenmiş, sonuçları grafiklerle sunulmuştur.

Manyika J. ve diğerleri (2013), yaptıkları çalışmada diğer çalışmalardan farklı olarak üç boyutlu kamera, lazer, LIDAR, GPS gibi diğer sensörleri de kullanmış, ve bu cihazlar ışığında hareket kabiliyetine sahip olan bir otonom araç geliştirmişlerdir. Araca yerleştirilen bu alıcılar dış ortamdaki kullanılabilir verileri toplamakta, geliştirilen algoritma vasıtasıyla araca hareket kabiliyeti verilmektedir. Bu kapsamda, farklı veri kümeleri sağlayan cihazların bir arada kullanılması ile trafik kurallarına uygun bir biçimde aracın nasıl hareket etmesi gerektiğine karar veren bir yapay zeka yazılımı geliştirilerek kapsamlı bir çalışmaya imza atılmıştır.

Yumruktay M. K. (2015), yaptığı çalışmada kullandığı araca, etrafını algılayabilmesi için farklı algılayıcılar yerleştirmiş ve bunun üzerinden insansız bir araç tasarlamıştır. Kullanılan modüller sayesinde araç uzaktan kontrol edilebilir hale getirilmiştir. Çalışmada, belirlenen hedef koordinatlar uzaktan kontrol modülü vasıtasıyla araca iletilmekte ve araçta bünyesinde bir başka modül olarak bulunan GNSS konum sinyal sistemi ile aracın kendisine iletilen hedef noktaya ulaşımı sağlanmıştır. Çalışma kapsamında GPS modülü kullanılarak aracın konumu belirlenmiş, elektromanyetik pusula ve jiroskop sayesinde de aracın yön belirleme işlemi gerçekleştirilmiştir. Aracın parkur esnasında karşılaşacağı engellerin tespit edilmesi için ise araca ultrasonik sensörler ilave edilmiş, RF ve Bluetooth modülleri ile araca kablosuz haberleşme imkanı sağlanmıştır.

Kurakin A. ve diğerleri (2016), gerçekleştirdikleri çalışmada Pilot Net sisteminin nesneleri tanımadaki başarısından yola çıkarak dış dinamiklerin arttırılması durumunda sistemin vereceği çıktıların incelenmesi ve iyileştirilmesi üzerine çalışmalar yapmışlardır. Yol içerisindeki işaretler, yol kenarındaki sabit etkenler gibi dış ortam özelliklerine odaklı yeni bir öğrenme yapısı geliştirmişlerdir. Sistem bu sayede gerçek trafik koşullarına daha yakın, öngörünün azaldığı kaotik bir ortamda daha detaylı özellikleri algılama konusunda başarı kat etmiştir.

Oh S. ve Kang H. B. (2016), yaptıkları çalışmada aracın nerede olduğunun tespitinin sağlanması için LIDAR ve araç üzerinden girdi görüntüsü alınması için stereo bir kamera kullanmışlardır. Stereo görüntü alıcılardan çıkarılan ışık algılama, yoğun derinlik haritaları kullanılarak ölçülen üç boyutlu parçacıklara "grid cell cluster" (GCC) adını verdikleri süper piksel tabanlı bir kümeleme yöntemi oluşturmuşlardır. Kullandıkları bu yöntem sayesinde tahmin matrisleri üzerinden hesaplama sürelerini ve hesaplama hatalarının miktarını azaltmışlardır. Geliştirtikleri yöntem KITTI kıyaslama veri setlerinde değerlendirilmiştir. Geliştirdikleri model sonuç olarak Mekahanachas yöntemine göre yaklaşık %38,9 daha hızlı ve yaygın olarak kullanılan diğer tekniklere kıyasla yaklaşık %12 daha doğru sonuçlar verdiği gözlemlenmiştir.

William B. ve Edvin V.O. (2016), yaptıkları çalışmada otonom araç için bir kontrol ünitesi geliştirmiş ve bunu yapay sinir ağı metodu ile gerçekleştirmişlerdir. Dış kaynaklardan veri alınması için tek girişli bir kamera kullanılmış ve alınan görüntüler, tasarlanan algoritma üzerinden işlenmiştir. Yapılan çalışmada kamera olarak Raspberry Pi cam v2 kullanılmıştır. Görüntü işleme kısmında ise ANNA Drive tercih edilmiş ve Python programlama dili kullanılarak PyGame oyun motoru kütüphanesi üzerinde uygulama geliştirilmiştir. Bu sayede aracı klavye üzerinden kontrol edebilecekleri bir arayüz oluşturmuşlardır. Dış kaynaktan üç farklı görüntü çözünürlüğüne sahip (25x25, 50x50 ve 100x100 piksel) toplam 900 veri kümesi bir araya getirilmiş ve farklı piksel seviyelerinde elde edilen başarı düzeyleri kıyaslanmıştır. Çalışmanın sonucunda 100x100 piksel çözünürlüğw sahip görüntü kümesi oluşturulmuş, oluşturulan görüntü kümesi üzerinden %78'lik bir başarı elde edilmiştir.

Doruk A. (2016), yaptığı çalışmada otonom sürüş için bir parkur oluşturmuş ve bir otonom araç projesi gerçekleştirmiştir. Proje kapsamında araç şerit takibi gerçekleştirebilmekte ve yoldaki sembollere göre hız ayarı yapabilmektedir. Oluşturulan yapıda mikroişlemci olarak Arduino Uno, yol takibinin sağlanması için TCRT5000 algılayıcı kullanılmış, parkurdaki renklerin ayrımının içinse TCS3200 renk algılama modülü kullanılmıştır. Ayrıca aracın önündeki engelleri algılayabilmesi için HC-SR04 mesafe sensörü ve cihaza kablosuz bağlantı için HC05 bluetooth modülü kullanılmıştır.

Android işletim sistemine sahip bir cihaz vasıtası ile yine Android tabanlı oluşturulmuş bir yazılım kullanılarak Arduino'ya kablosuz bağlantı sağlanmıştır. Araç, uzaktan kontrol mekanizması ile hareket etmeye başladığında yol üzerinde kullanılan her bir rengin kendi özelinde kodlanmış olan hız değeri araç tarafından algılanarak belirlenen hız seviyelerinde aracın parkuru tamamlaması sağlanmıştır.

Bojarski M. ve diğerleri (2016), Nvidia şirketinin otonom araçlar için görüntü işleme, yapay zeka gibi konularda araştırma-geliştirme çalışmalarına destek olması amacıyla oluşturmuş olduğu CUDA sisteminden faydalanmış, dış kaynaktan gelecek verileri işlemesi amacıyla donanım altyapısı olarak GPU (grafik işletim birimi) kullanmıştır. Yapılan çalışmada aracın önüne yerleştirilen bir kamera vasıtasıyla standart bir sürüş esnasında görüntüler toplamış ve bu görüntüler ile evrişimli sinir ağlarını eğitmişlerdir. Sinir ağının öğrenme işleminin yeterli düzeye ulaşması sonucu eğitilen algotima, yol takibi için şerit çizgileri olan-olmayan tüm yollarda ve otoban trafiğinde otonom sürüş kabiliyetine sahip olmuştur. Algoritma eğitimi esnasında geliştirme kiti olarak Nvidia DevBox kullanılmış ve yazılım konusunda Torch tercih edilmiştir. Ayrıca aracın nereye gideceğini belirlemek için Torch üzerinden çalıştıran ve otonom sistem geliştirme platformu olan Nvidia Drive araç bilgisayarı kullanılmıştır. Sistem saniyede 30 kare hızında (FPS) çalışabilmiş ve komutları eş zamanlı gerçekleştirebilmiştir.

Bojarski M. ve diğerleri (2016), ilk oluşturdukları modeldeki eksiklikleri tespit etmiş ve bunları gidermek amacıyla 2017 yılındatekrar bir çalışma gerçekleştirmişlerdir. Bu çalışmada kullanılan yol üzerinden toplanan kamera verileri ile aracın direksiyon kontrolünü sağlayan PilotNet ağ mimarisi kullanılmış ve bir otonom araç sistemi ortaya konmuştur. PilotNet kullanımında eğitim öncesi kamera üzerinden fotoğraf kareleri toplanmış, aynı zamanda parkur boyunca fotoğraf alınması esnasında aracın direksiyon açıları da kayda geçilmiştir. Yapılan testlerin sonucunda, PilotNet ağ yapısının yolda şerit çizgileri olmasa dahi aracın kendi yolunu başarılı bir şekilde takip ettiği gözlemlenmiştir. Ayrıca 2017 yılında yaptıkları diğer bir çalışmada görüntü işleme ve evrişimli sinir ağları konusunda PyTorch üzerinden "Visual Back Prop" adlı bir yöntem kullanmışlardır. Bu yöntem, bir evrişimli sinir ağı modeli

üzerinde işlenecek olan görüntülerin hangilerinin çıktı tahminine en fazla katkıda bulunduğunun tespit edilmesini sağlamıştır. Aynı zamanda bu yöntem sayesinde işe yaramayacak olan bilgi kümeleri sistemden çıkarılmakta ve algorimanın eğitimi esansındaki gereken zamandan kazanç sağlanmıştır.

Şanlı E. (2018), yapmış olduğu çalışmada radyo kontrollü bir otonom araç projesi gerçekleştirmiştir. Çalışmada mikroişlemci olarak Raspberry Pi kullanılmış, kamera modülü üzerinden alınan fotoğraf kareleri 120x160 piksel çözünürlüğüne dönüştürülmüş ve eğitim verisi haline getirilmiştir. Programlama dili olarak Python kullanılmış, TensorFlow'un görüntü işleme yöntemleri ve kütüphaneleri kullanılarak yapay sinir ağı modeli oluşturulmuş ve aracın otonom sürüşü gerçekleştirilmiştir. Yapılan çalışmada, otonom aracın konum verisi için GPS, mesafe ölçümü için ultrasonik sensör kullanılmıştır. Çalışma sonunda otonom araçların özellikle toplu taşımada, lojistik birimlerinde ve uzun yol araçları gibi birçok farklı alanda faaliyet gösterebileceği öngörülmüş ve bu konu hakkında değerlendirmeler yapılmıştır.

Kayaduman A. ve diğerleri (2018), yaptıkları çalışmada otonom kara aracı üretmeye odaklanmışlardır. Kullandıkları konum sensörü ve hız sensörleri ile aracın hızı ve konumu kontrol edilebilmektedir. Kamera modülü ve çeşitli sensörlerden gelen veriler yazılım kütüphaneleri yardımıyla mikro kontrolörlerden teşkil edilmiş bir ağ üzerinde işlenmiş ve yorumlanması uygulamalı olarak gerçekleştirilmiştir. Kamera modülü ve görüntü işleme algoritmaları vasıtasıyla şeritler algılanmakta, elde edilen görüntüler ve diğer sensörlerden gelen veriler ise gömülü sistem üzerinde oluşturulan bir merkezi yazılım ile aracın hareketi için gerekli komutları üretmektedir.

Chen Y. ve diğerleri (2018), yaptıkları çalışmada Venodyle-HDL32E lazer ve kamera sistemi kullanarak aracın takip ettiği yolun kamera görüntüsünü sürücünün davranışlarıyla birlikte kaydetmişlerdir. Bu doğrultuda, lazerle birlikte gerçekleştirilen çalışmanın, sadece kameradan edilen video karelerinin kullanılıp sinir ağı ile analiz edilmesine dayanan çalışmaya kıyasla tahmin doğruluğunu büyük ölçüde arttırdığı gözlemlenmiştir.

Xu S. ve diğerleri (2020), yaptıkları çalışmada otonom araç üzerine yerleştirdikleri donanımlar sayesinde araç çevresinden üç boyutlu görüntüler almışve bu görüntüler üzerinden obje tanıma sistemi oluşturmuşlardır. Oluşturdukları sistemde görüntü için kamera ve tarama için LIDAR donanımı kullanılmıştır. 2020'de gerçekleştirdikleri bir diğer çalışmada ise yine otonom sürüşte kullanılması adına nokta bulutu tabanlı üç boyutlu nesne algılama için yeni ve tek aşamalı birleşik ağ bir olan HVNet'i kullanmışlardır. Pikselin üç boyutlu karşılığı olan voksel üzerinden hareket ederek oluşturdukları sistemde aynı alanı daha büyük voksellerle oluşturdukları özellik haritasında küçük voksellere kıyasla işlemlerin daha hızlı gerçekleştirildiğini, ancak küçük nesneler için karmaşık özellikleri ve doğru konumu yakalayamadıklarını tespit etmişlerdir. Yapılan çalışmada bu sorunu çözen hibrit bir voksel ağı oluşturulmuş, KITTI kıyaslaması üzerindeki deneylerlerde gerçek zamanlı çıkarım hızı açısından tüm mevcut yöntemler arasında en iyi mAP'ye ulaşıldığı belirtilmiştir.

Cho M. (2019), gerçekleştirdiği çalışmada sürücüsüz araçların otonom sürüşleri esnasında etrafından geçen insan, nesne ve araçları algılayabilmeleri için kamera, radar ve LIDAR 'ı bir arada kullanmışlardır. Yayaların hatasız bir biçimde tanımlanabilmesi için ısı ve LIDAR sensörleri ve kızılötesi kamera kullanılarak yeni bir engel tanıma yöntemi önermişlerdir. Engel tanımanın yanısıra çarpışmadan kaçınma, çarpışmalar öncesinde doğru, zamanında ve güvenilir uyarılar vermeyi amaçlayan ADAS kullanılmıştır. LIDAR ve radar kullanan geleneksel engel tanıma teknolojilerinin, kötü hava koşullarında ve geceleri nesneleri doğru bir şekilde tanımlayamadıkları ve kazalara sebebiyet verdiği üzerinde durulmuş, yapılan çalışmada ise öndeki araç ve etraftaki yayaların önceden tanınması için ısı algılayabilen bir kızılötesi kamera ek olarak sisteme ilave edilmiştir.

Bingöl M. S. ve diğerleri (2019), yaptıkları çalışmada otonom araçlar için algılama sistemi geliştirmişlerdir. İşletim sistemi olarak Linux kullanılmış ve ilk çalışma olarak MNIST veri kümesi üzerinden rakam tanıma yapılmıştır. Gerçekleştirilen otonom araç uygulamasında Nvidia TK1 ile Nvidia TX1 kartları kullanılmış ve derin öğrenme yöntemlerini kullanan otonom bir araç tasarlanmıştır. Aracın üzerine yerleştirilen çeşitli sensörler sayesinde araç tasarlanan parkur üzerinde test edilmiştir.

Aki K. ve Dirik A. E. (2020), yaptıkları çalışmada makine öğrenmesi ve derin öğrenme alanındaki gelişmeler ve karmaşık problemlerin çözümü hakkında literatür incelemesi gerçekleştirmişlerdir. Ardından PID kontrol ünitesi üzerinden ve derin öğrenme yöntemi üzerinden olmak üzere iki farklı yöntem kullanılarak bir RC aracın ayrı ayrı otonom sürüşü sağlanmış, PID kontrol tabanlı ve derin öğrenme tabanlı iki farklı sistemin araç üzerindeki otonom sürüş performansları kıyaslanmıştır. Sonuç olarak PID ile de başarılı sonuçlar elde edilse de derin öğrenme tabanlı modelin daha başarılı sonuçlar verdiği ortaya konmuştur.

Aisha A. M. ve diğerleri (2021), yaptıkları çalışmada engellerden kaçınma sistemine sahip bir RC otonom elektrikli araç prototipi üretmişlerdir. Uygulamada denetleyici olarak Arduino kullanılmış ve dış kaynaktan veri alımı konusunda ultrasonik sensörlere başvurulmuştur. Ultrasonik sensör verilerine göre araç engellerden kaçınabilir hale gelmiştir. 2-200 cm nesne algılama aralığı bulunan araçta güç kaynağı olarak güneş paneli kullanılmıştır. Bu sayede aracın harici güç kaynağına bağımlılığı azaltılmış ve kullanım süresi açısından süre kısıtlaması ortada kaldırılmıştır.

Literatür araştırması sonucunda otonom sistemlerin, akademik çalışmalarda önemli bir yer edinmekte olduğu ve sürekli olarak geliştiği görülmüştür. Özellikle otonom araçlar, kullanıldıkları ortam gereği birçok farklı dinamik unsuru içinde barındırmaktadır. Bu unsurlar farklı şehirlerin farklı yol yapıları, zorlayıcı, görüş engelleyici hava koşulları, hayvanlar, yayalar ve diğer araçların hareketleri gibi birçok etmeni barındırmakla birlikte, dünyanın farklı yerlerinde farklı faktörler de ortaya çıkabilmektedir. Literatürdeki çalışmalarda radyo kontrollü prototip otonom araç üzerinden çalışmalar halihazırda devam etmekle beraber, bazı çalışmalarda gerçek araçlar üzerinden gerçek trafik koşulları ile sinir ağlarının eğitildiği ve farklı modellerin geliştirildiği de gözlemlenmiştir. Tüm bu etmenler ve çalışmalardaki çeşitlilik sayesinde otonom sürüşü sağlamak amacıyla üretilen sistemler amaca uygun birçok çeşitlilik sunmakta, birçok farklı metot ve yöntem birbirleriyle kıyaslanarak sürekli güncellenebilmektedir.

Kaynak araştırmasında görüleceği üzere otonom araç uygulamalarında mikroişlemciler, geliştirme kartları, kızılötesi sensörler ve kameralar gibi birçok farklı donanım ve birçok farklı algoritma ile öğrenme metotları kullanılmıştır. Yapılan çalışmalar dikkate alındığında, derin öğrenme algoritmalarının daha verimli çalışması için, kullanılan geliştirme kartlarının sürekli olarak RAM değerlerinin arttırıldığı ve işlem birimi olarak GPU'ların tercih edildiği gözlemlenmiştir. Bu yaklaşımlar sayesindedir ki otonom araçlar yol tutuşları ve kendilerinden istenen görevler konusunda daha başarılı sonuçlar ortaya koyabilmektedir. Bu doğrultuda, bu tez çalışmasında sürücüsüz bir aracın otonom hale getirilmesi için derin öğrenme metodu oluşturulmuş ve evrişimli sinir ağı üzerinden model eğitimi yapılmıştır. Küçük bir yer aracı temel alınarak, uygulama esnasında birçok yazılım kütüphanesiyle beraber PyTorch kütüphanesinin yoğunlukta kullanımı ile derin öğrenme yöntemi oluşturulmuştur. Donanımsal olarak Nvidia Jetson Nano geliştirme kartı, Raspberry Pi geniş açı kamera modülü ve ultrasonik mesafe sensörü kullanılmış, araç ve aracın üzerinde hareket edebileceği bir parkur hazırlanmıştır. Sinir ağının eğitilebilmesi için araç üzerindeki kamera vasıtasıyla parkur üzerinden veri setleri alınmış ve alınan görüntüler üzerinden eğitim gerçekleştirilmiştir. Ardından sürüş esnasında elde edilen test verileri, eğitilen sinir ağı modeliyle işlenerek aracın otonom bir şekilde parkuru tamamlaması sağlanmıştır. PyTorch kütüphanesi yardımıyla derin öğrenme metodu kullanılarak araç şerit takibini gerçekleştirmiştir. Aynı zamanda aracın üzerine Arduino tarafından kontrolü gerçekleştirilen ultrasonik mesafe sensörü takılmıştır. Bu sensör vasıtasıyla aracın yakınındaki nesneler tespit edilmekte ve belli bir mesafe sonrasında çarpmanın engellenmesi amacıyla araç durmaktadır. Kamera modülü vasıtasıyla parkur üzerindeki yol takibi sağlanmakta, kameradan üretilen görüntü verileri geliştirme kartı ve üzerindeki algoritma sayesinde aracın hareketi için gerekli komutları üretmektedir. Bu sayede oluşturulan araç hem kamera hem de ultrasonik sensör üzerinden çevresini algılayarak otonom hareket kabiliyetine sahip olmaktadır.

Makine öğrenmesi, yapay zekanın bir alt bilim dalı olarak incelenebilir. Kendi bünyesinde birçok alt uygulama biçimini ve farklı öğrenme yapılarını bulundurmaktadır. Makine öğrenmesinin alt başlıklarından biri de derin öğrenmedir. Derin öğrenme (deep learning), dünya üzerinde veri miktarının büyük boyutlara ulaşması ve işlenmesi gereken veri miktarının artması ile hızlı bir gelişim süreci yaşamıştır. Bu öğrenme yöntemi, kendi

kendine öğrenebilen çok katmanlı yapay sinir ağlarından oluşan bir öğrenme yöntemidir. Şekil 2.1'de yapay zeka, makine öğrenmesi ve derin öğrenmenin kapsam haritası verilmiştir.

Şekil 2. 1. Yapay zeka, makine öğrenmesi ve derin öğrenme kapsam haritası (Aki, 2019)

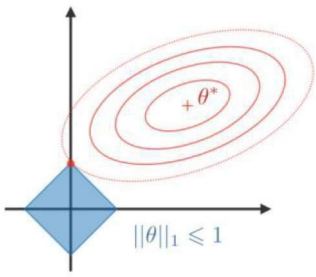
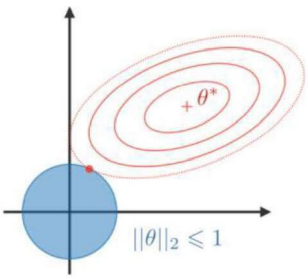
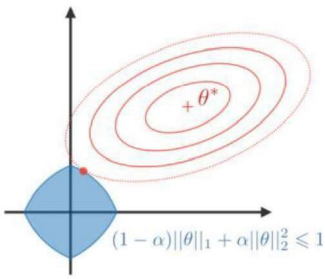
Literatürdeki çalışmalarda derin öğrenme uygulamaları genel olarak incelendiğinde konuşma tanıma (speech recognition), görüntü sınıflandırma (image classification), doğal dil işleme (natural language processing) gibi yeni çalışma ve araştırma alanlarında başarı gösterdiği tespit edilmiştir.

Otonom araç uygulamalarının ana unsurlarından biri sabit veya hareketli nesneleri algılamak, tanımlamak ve amaca uygun bir şekilde takip etmektir. Bu eylemler, otonom sistemlerin geliştirilmesine mani olabilecek düzeyde büyük bir sorun olmuştur. İstenilen bu eylemlerin gerçekleştirilebilmesi adına çeşitli özellik çıkarma algoritmaları geliştirilmiştir. Bu algoritmalar genel olarak; "Gradyan Histogramı (HOG - Histogram of Gradient) (Dalal & Triggs, 2005), Ölçekten Bağımsız Öznitelik Dönüşümü (SIFT - Scale Invariant Feature Transform) (Ke & Sukthankar, 2004), Hızlandırılmış Gürbüz Öznitelikler (SURF - Speeded up Robust Feature) (Bay, Tuytelaars, & Gool, 2006), İkili Gürbüz Temel Öznitelikler (BRIEF - Binary Robust Independent Elementary Feature) (Calonder, Lepetit, Strecha, & Fua, 2010), Yönlendirilmiş Hızlı ve Döndürülmiş BRİEF (ORB - Oriented Fast and Rotated BRIEF) (Rublee, Rabaud, Konolige, & Bradski, 2011), İkili Gürbüz Değişmez Ölçeklendirilebilir Anahtar Noktalar (BRISK - Binary Robust

Invariant Scalable Keypoints) (Leutenegger, Chli, & Siegwart, 2011) ve Hızlı Retina Noktası (FREAK - Fast Retina Keypoint) (Alahi, Ortiz, & Vandergheynst, 2012)" şeklinde sınıflandırılabilir (Bingöl, 2018).

Diğer taraftan nesnelerin algılanması için geliştirilen sınıflandırma metotlarına örnek olarak ise "Makine Öğrenmesi (Machine Learning), Yapay Sinir Ağları (Artificial Neural Network)) (Duda, Hart, & Stork, 2000); Küresel / Eliptik sınıflayıcı (Uçar, Demir, & Güzeliş, A penalty function method for designing efficient robust classifiers with input–space optimal separating surfaces, 2014), Destek Vektör Makineleri (Support Vector Machine) (Uçar, Demir, & Güzeliş, A penalty function method for designing efficient robust classifiers with input–space optimal separating surfaces, 2014), Aşırı Öğrenme Makineleri (Extreme learning machine) (Uçar, Demir, & Güzeliş, 2016), Adaboost, Naive Bayes ve Karar Ağaçları (Decision tree learning) (Webb, Boughton, & Wang, 2005)" verilebilir (Bingöl, 2018).

Makine öğrenmesinde modelin seçimi konusunda veri setleri en temel haliyle ikiye ayrılabilir. Eğitim veri seti ve test veri seti. Eğitim veri seti algoritmanın göreceği ve eğitim için kullanacağı şekilde olmalıdır. Algoritmanın daha öncesinde hiç karşılaşmadığı görüntüler ise test kümesi olarak adlandırılmalı ve eğitim esnasında bu veri kümesi kullanılmamalıdır. Aynı zamanda makine öğrenmesi algoritması, girilen veri kümelerini çok tekrar yaparak aşırı öğrenmeye maruz kalmamalıdır. Bunu engellemek amacıyla algoritma içerisinde düzenlileştirme (regularization) işlemine ihtiyaç duyulur. Şekil 2.2'de algoritmanın çalışma süresince doğru tahmin oranını arttırmak için tercih edilen farklı düzenlileştirme teknikleri verilmiştir.

Lasso	Ridge	Elastic Net
		
... $+\lambda.\ \|\|\theta\|\|_1$ $\lambda \in R$	... $+\lambda.\ \|\|\theta\|\|_2^2$ $\lambda \in R$	... $+\lambda.[(1-\alpha)\|\|\theta\|\|_1+\alpha\|\|\theta\|\|_2^2]$ $\lambda \in R$

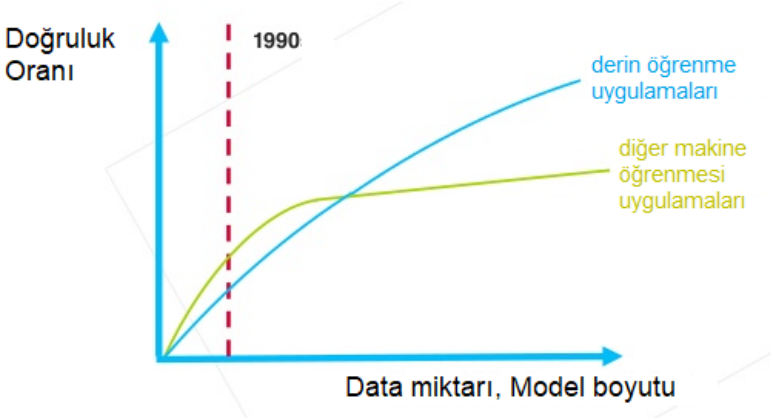
Şekil 2. 2. Tahmin doğruluğunu arttırmak için tercih edilen farklı düzenlileştirme teknikleri (Amidi & Amidi, 2018)

Lineer regresyonda ana amaç bağımsız değişkenlerin kullanılması ile bağımlı yani hedef değişkenin tahmin edilmesidir. Bu regresyon biçimi, makine öğrenmesinin en temel algoritmalarından biridir. Temel olmasına karşın veri setindeki öz nitelik sayısının artması ile beraber modelin karmaşıklığı da artmakta, bu da lineer regresyonda modelin ezberlenmesine yol açmaktadır. Ezberleme yapan bir model, dış kaynaktan gelen bir veriyi iyi tahmin edemez ve oluşturulan model genelleştirilemez bir hal alır. Bu soruna çözüm sunması amacıyla Şekil 2.2'de verilen Ridge ve Lasso gibi farklı regresyon modelleri kullanılmaktadır. Bu regresyon modelleri, tahmin ettiği ve olması gereken ağırlık değerleri arasında çıkarma işlemi gerçekleştirerek iterasyon esnasında oluşan kayıp değerini tespit ederler. Aralarındaki fark ise Ridge regresyonunda ağırlık değerlerinin karesi alındıktan sonra çıkarma işlemi uygulanırken, Lasso regresyonunda ağırlık değerleri mutlak değer içerisine alınarak çıkarma işlemi gerçekleştirilir. Her ikisi de pozitif bir kayıp değeri oluşturur. Ridge regresyonda karmaşıklık oranı daha az iken, daha karmaşık ve yüksek öznitelik istenen yerlerde Lasso regresyonu tercih edilir.

Özellikle son 10 yılda işlenmesi gereken veriler çok büyük boyutlara ulaşmış ve yeni veri kaynaklarından sürekli olarak büyük, karmaşık veri kümeleri oluşmuş ve oluşmaya devam etmektedir. Bu veriler büyük veri (big data) olarak adlandırılabilir. Büyük veri temel olarak analiz edilebilen, sınıflandırılan ve bir anlam ifade edecek hale

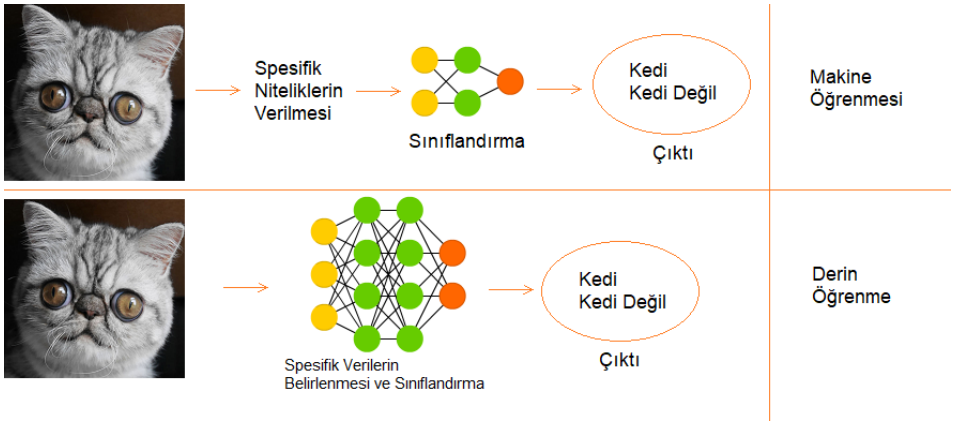
dönüştürülebilen verilerin tümüdür. Büyük veri her geçen yıl artarken bu kavramı net bir veri miktarı ile sınırlandırmak gitgide zorlaşmaktadır. Ancak sezgisel olarak bir milyon örnek, büyük veri denmesi için yeterlidir. Hemen hemen her alanda kendine yer bulan büyük verinin kullanım alanını sınırlandırmak da pek mümkün olmamaktadır. Tüketicilerin davranışlarını izlemek isteyen firmalar, yaratıcı eğilimler oluşturmak isteyen şirketler, oluşan bir durumu açıklamaya çalışan araştırmacılar veya girişimciler olmak üzere birçok kişi ve kurum büyük veriyi analiz ederek kendi amaçları doğrultusunda kullanmaktadırlar. Bu veriyi işleyebilen şirketler yatırımlarını en verimli şekilde gerçekleştirebilme imkanına sahip olmakta ve bunun olumlu katkılarını kısa sürede görmektedirler. Yapılan araştırmalara göre büyük veriyi kullanan şirketler; %50 daha fazla kazanç elde etmiş, pazar çalışmalarında %41 etkili olmuş, reklam harcamaları %37 azalmış ve sosyal medya kullanımında %37 gibi yüksek oranlara ulaşan bir başarı elde etmişlerdir (Big Data Turkey, 2019).

Ekonomik açıdan böyle bir getirisi olan ve sürekli olarak artan büyük veriyi işlemek için algoritmaların kabiliyetlerinin arttırılması gerekmiş ve diğer makine öğrenmesi uygulamaları performans açısından yeterli sonuçlar verememeye başlamıştır. Bu eksiklik doğrultusunda özellikle son yıllarda yapılan çalışmalarda işlenmesi gereken veri miktarının artması sonucunda derin öğrenme metodu diğer makine öğrenmesi uygulamalarına kıyasla ön plana çıkmış ve veri miktarının artması ile daha doğru sonuçlar verdiği tespit edilmiştir. Şekil 2.3'te verilen grafikte işlenmesi gereken verilerin artması sonucu diğer makine öğrenmesi uygulamalarına kıyasla derin öğrenme uygulamalarının avantajı ortaya konmuştur.



Şekil 2. 3. Derin öğrenme ile diğer makine öğrenmesi uygulamalarının veri miktarına göre kıyası (Dossman, 2018)

Derin öğrenme yöntemi diğer geleneksel yaklaşımlardan işleyiş olarak farklıdır. Diğer makine öğrenmesi uygulamalarında algoritmanın çıktı vermesi için işleyeceği veriler hakkında manuel olarak girilmesi gereken ön bilgiler istenmektedir. Örneğin kedi ve köpek görsellerinin bulunduğu bir veri setinde makine öğrenmesi yöntemi kullanılması durumunda bu hayvanların kendine has özelliklerinin önceden belirlenmesi ve algoritmaya verilmesi gerekmektedir. Aynı çalışma derin öğrenme üzerinden gerçekleştirileceğinde ise, algoritma kendisine verilen görüntüler sayesinde bu ayrımı kendisi yapmaktadır. Diğer bir deyişle derin öğrenme uygulamalarının tahmin yürütebilmesi için algoritma, sınıflandırma özelliklerini doğrudan verilerden alır. Makine öğrenmesi ve derin öğrenme arasındaki bu fark Şekil 2.4'te görsel olarak izah edilmiştir.



Şekil 2. 4. Makine öğrenmesi ve derin öğrenme yöntemleri arasındaki fark (Anonim, 2019)

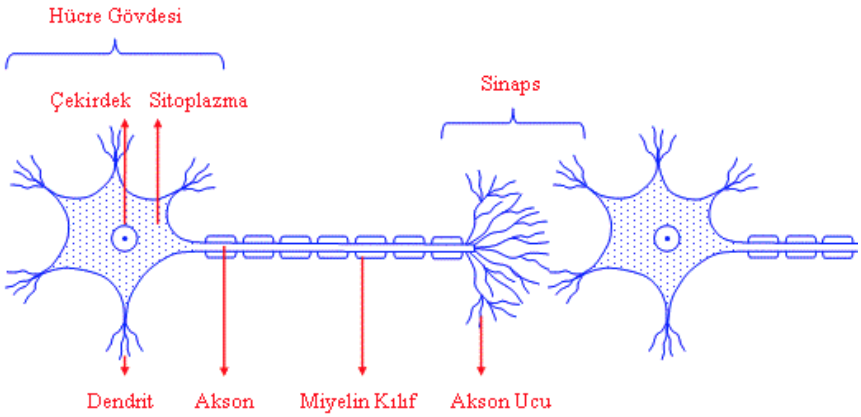
Bu yaklaşım modeli üzerinden otonom bir aracın sürüş uygulaması ele alındığında algoritma, sürüş esnasında alınan görüntülerin işlenmesi ile doğrudan öğrenir. Bu yöntemde açık ve gizli katmanlar bulunmakta ve algılama derinlikleri her katmanda değişmektedir. Örneğin çizgi, sınır ve köşe gibi basit özellikleri, düşük seviyeli katmanlar; yaya, araba veya trafik işaretleri gibi yüksek özellik barındıranları yüksek seviyeli katmanlar algılar. Diğer bir deyişle derin öğrenme yöntemleri, nesnelerin farklı düzlemlerde ve katmanlarda incelenmesine olanak sağlar. Özellikle ImageNet sınıflama yarışmalarında derin öğrenme yöntemlerinin ve bu paralellikte kullanılan algoritmaların gösterdiği başarılar kullanılan yöntemin önemini ortaya çıkarmıştır (Berg, Deng, & Fei-Fei, 2010). Bu yarışmalarda yapay sinir ağlarının bir türü olan Evrişimli Sinir Ağları (CNN) kullanılmıştır (Bingöl, 2018).

2.1. Yapay Sinir Ağları (ANN)

Canlıların davranışlarının incelenip, matematiksel olarak modellenip, benzer yapay modellerin üretilmesine sibernetik denir. Eğitilebilir, uyum sağlayabilen (adaptive), kendi kendine organize olup öğrenebilen ve değerlendirme yapabilen yapay sinir ağları ile insan beyninin öğrenme yapısı modellenmeye çalışılmaktadır. Tıpkı insanda olduğu gibi yapay sinir ağları vasıtasıyla makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır. Yapay sinir ağları, kuramsal olarak verilen herhangi bir dönüşüm görevini belli bir doğruluk oranıyla öğrenme yeteneğine sahiptir (Bingöl, 2018). İnsanda

olduğu gibi yapay sinir ağları vasıtasıyla makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır.

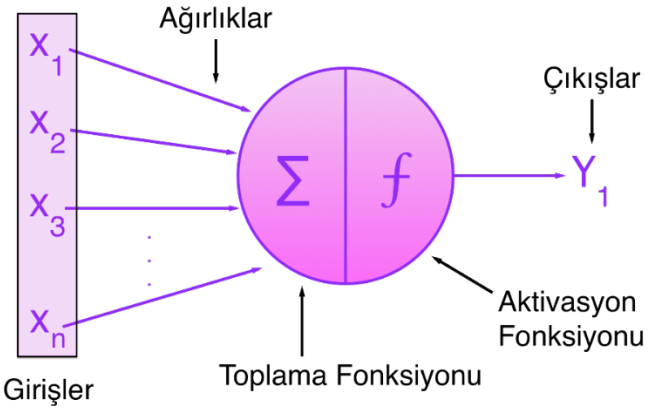
İnsandaki bir sinir hücresinin (nöron) yapısı Şekil 2.5'te gösterildiği gibidir.



Şekil 2. 5. Bir sinir hücresinin biyolojik gösterimi (Güzel, 2018)

Biyolojik bir sinir hücresinin yapısında bulunan sinapslarda sinyaller oluşmakta ve oluşan sinyaller dendritler vasıtasıyla hücre içerisine alınmaktadır. İçeriye alınan sinyaller, hücre gövdesinde işlenmektedir. İşlenen bu sinyaller aksonlar aracılığı ile denetlenmekte ve sinapslar vasıtası ile hedef hücrelere iletilmektedir. İnsan beyni 10 milyar sinir hücresinden ve 60 trilyon sinaps bağlantısından oluşur (Güzel, 2018).

Yapay sinir ağında kullanılan hücre modeli, insan sinir hücresine kıyasla daha basit ve kolay anlaşılabilir bir çalışma mekanizmasına sahiptir. Bu yapı Şekil 2.6'da gösterildiği gibi incelenebilir.



Şekil 2. 6. Yapay sinir ağı hücre modeli (Güzel, 2018)

Temel bir yapay sinir ağı hücresinde:

- Giriş değerleri
- Ağırlık parametreleri
- Toplam fonksiyonu
- Aktivasyon fonksiyonu
- Çıkışlar değerleri bulunur.

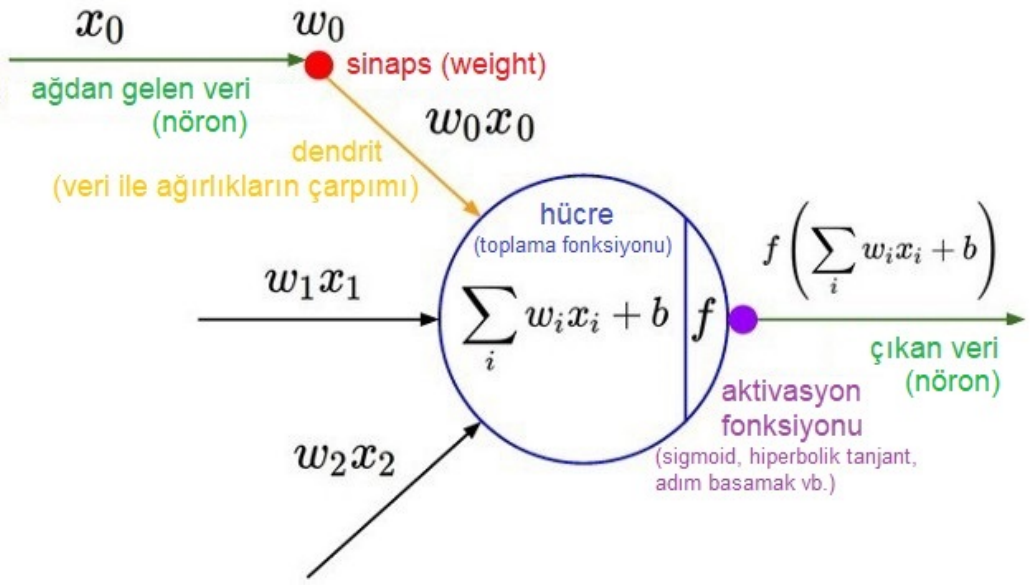
Dış kaynaktan veya çevredeki hücrelerden gelen girdi bilgileri, ağırlık değerleri ile hücre içine iletilir. Kullanılacak olan toplam fonksiyonu seçildikten sonra bu fonksiyon üzerinden net girdi değeri hesaplanır. Net girdi değerinin aktivasyon denklemine iletilmesi ve fonksiyon üzerinden sonuç elde edilmesiyle bir çıktı değeri oluşturulur. Her hücre içi irtibatın bir önemlilik değeri bulunmaktadır. Tüm bu uygulamanın ana hedefi, ağa verilen örnekler sonucunda tüm veriler için doğru çıktı üretecek optimum seviyedeki ağırlık değerlerini bulmaktır.

Sinir ağı hücresine gelen girdi verileri, ağın eğitileceği amaca uygun örnekler olmalıdır. Girdi verisi, çevre ortamından alınan veriler olabildiği gibi başka sinir ağı hücrelerinden de bilgi akışı gerçekleşebilir. Ağırlık değerleri ise, hücreye iletilen verinin önemlilik derecesini ve bulunduğu hücre üzerindeki etkisini gösterir. Bütün girdi verilerinin kendine ait bir ağırlık değerleri bulunmaktadır. Ağırlık verileri, yapılan çalışma ve seçilen fonksiyon kapsamında değer bakımından pozitif, negatif, sabit veya değişken olabilir.

Hücreye gelen net girdiyi hesaplamak için toplam fonksiyonu kullanılır. Toplam fonksiyonu olarak birçok farklı fonksiyon türetilmiştir. Bu fonksiyon, genellikle işlenecek veriye ve istenen sonuca göre deneme-yanılma yolu ile belirlenir. Oluşturulan sinir ağında, her hücrenin kendine ait ağırlık değerinin olması gibi yine her hücre kendine ait farklı bir toplama fonksiyonuna sahip olabilir.

Aktivasyon fonksiyonunun temel görevi, hücre içerisinde toplanan net girdi değerlerini kullanarak hücrenin bu girdi verisine karşı oluşturacağı çıktıyı belirlemesidir. Aktivasyon fonksiyonları doğrusal olmayan bir fonksiyon olması gerekir. Özellikle geri yayılım işlemi esnasında fonksiyonun türevinin alınabilmesi adına aktivasyon fonksiyonu türevlenebilir olmalıdır. Tercih edilen fonksiyonlar arasında en çok kullanılan aktivasyon fonksiyonu sigmoid fonksiyonudur (Güzel, 2018).

Şekil 2.6'da verilen hücrenin matematiksel modeli Şekil 2.7'deki gibidir. Bu model literatürde lojistik regresyon ağ grafiği olarak da adlandırılmaktadır.



Şekil 2. 7. Tek katmanlı yapay sinir ağı matematiksel ağ grafiği

2.2. Yapay Sinir Ağının Öğrenmesi

Yapay sinir ağlarının öğrenme mekanizması biyolojik sinir ağı yapısına çok benzemektedir. Gerçek hayatta, yapılacak olan eylemin daha iyi sonuç vermesi için eğitime ihtiyaç duyulmaktadır ve bireyler eğitim üzerinden kendilerini geliştirirler. Aynı şekilde sinir ağı modelleri de giriş sinyallerinden aldıkları verileri değerlendirip doğru tanımlamak için eğitime ihtiyaç duyarlar. Ağ modeli bir sonuç üretir. Elde edilmesi gereken mutlak doğru ile ağ modeli tarafından üretilen sonuç kıyaslanır ve aradaki fark kayıp değeri olarak hesaplanır. Elde edilen bu kayıp değeri model içerisinde tekrar işlenmek üzere geri gönderilir ve tüm katmanlar kayıp değerini analiz ederek bir sonraki giriş sinyali için weight (w_i, ağırlık) ve yanlılık ($b, bias$) matrislerini tekrar günceller. Bu sayede sinir ağı, her giriş verisi ile kendisini sürekli eğitmiş olur. Tüm bu işlemler belirli başlıklar altında sınıflandırılarak anlatılmıştır.

2.2.1. İleri yayılım (forward propagation)

Sinir ağı modelinde, girdi katmanından kayıp ve maliyet fonksiyonlarına kadar gerçekleştirilen bütün ileri yönlü hesaplama akışına ileri yayılım adı verilmektedir.
Yapay sinir ağının en küçük parçası olarak bilinen algılayıcı (perceptron), aşağıdaki gibi lineer bir fonksiyonla ifade edilmektedir. Algılayıcı, ilk defa 1957 yılında Frank Rosenblatt tarafından tanımlanmıştır (Kızrak, 2018).

$$Z = \sum_i^n w_i \ . x_i + b \tag{2.1}$$

Burada;
x_i: bağımsız giriş verisini,
w_i: ağırlık parametresini,
b: yanlılık (bias) değerini temsil etmektedir.

Tüm derin öğrenme modellerinde asıl hedef, modelin doğruya en yakın değeri vereceği w_i ve b parametrelerini hesaplamaktır. Bu işlem ile Z değeri elde edilir ve her piksel üzerinde elde edilen bu değer aktivasyon fonksiyonu ile ihtimallere dayalı (probabilistic) bir tahmin değerine dönüştürülür. İhtimallere dayalı bir tahmin değerine

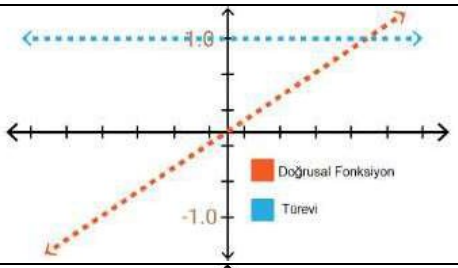
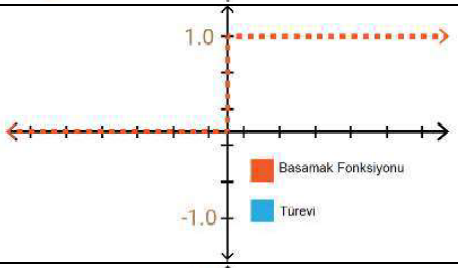
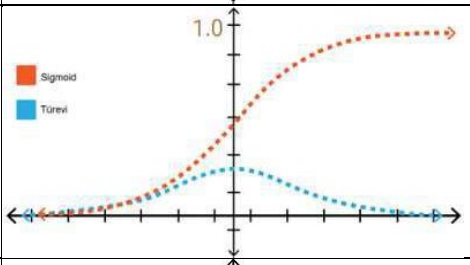
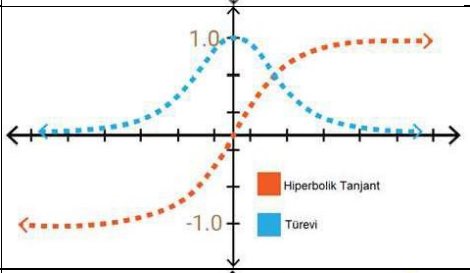
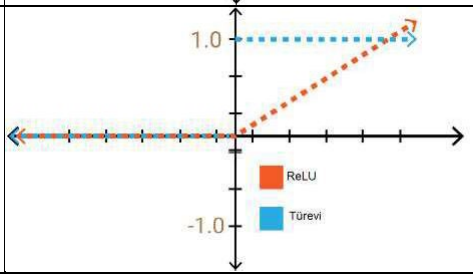
dönüştürülmesinin temel nedeni, sistemi fonksiyon ile daha karmaşık bir duruma getirip doğrusal olmayan bir hale sokmak ve yeni üretilen değerin daha nitelikli sonuç vermesini sağlamaktır. Aktivasyon fonksiyonu olarak tercih edilecek fonksiyonlar veri kümesine göre öğrenme ve başarı oranı açısından birbirlerinden değişik sonuçlar ve farklı doğruluk oranları verebilmektedirler.

Aktivasyon fonksiyonu

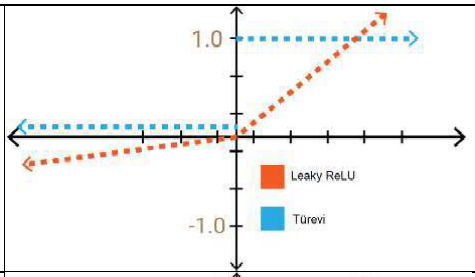
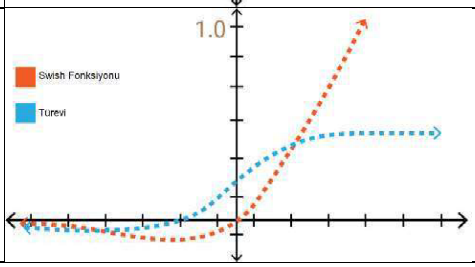
Aktivasyon fonksiyonu, algoritmanın öğrenme parametresi olarak tanımlanabilir. Bu fonksiyonlar temel olarak ileri yayılım esnasında elde edilen veriyi analiz ederek bu veriye karşılık üreteceği çıktı bilgisinin belirlemesine olanak tanır. Ayrıca bu fonksiyonlar çıktı verilerini beklenen değerler arasında sınıflandırmayı sağlar.

Aktivasyon fonksiyonları belirlenen aralıkta bir tahmin değeri oluşturmakla beraber bu tahmin değerinin doğruluğunun değerlendirilebilmesi amacıyla süreklilik arz eden türevi alınabilir fonksiyonlar olmalıdırlar. Sinir ağı modelinde tek veya çift girdili fonksiyonlar kullanılabilmektedir. Fonksiyon seçimi, algoritmanın kullanım hedefine göre değişiklik gösterebilir. Aktivasyon fonksiyonlarının doğrusal olan veya olmayan problemlere uygulanması, sinir ağı içerisinde karmaşık hale gelen problemlerin analiz edilmesine olanak sağlamaktadır (Rosenblatt, 1958). Yapılacak çalışmaya en uygun fonksiyonun seçilmesi modelin verimliliği açısından önemli bir kriterdir. Çizelge 2.1'de, yapılan çalışmalarda en çok tercih edilen aktivasyon fonksiyonları verilmiş, Çizelge 2.2'de ise bu fonksiyonların matematiksel ifadeleri belirtilmiştir.

Çizelge 2. 1. Yapay sinir ağı modellerinde kullanılan aktivasyon fonksiyonları (Aki, 2019)

Aktivasyon Fonksiyonu Adı	Fonksiyon Grafiği	Fonksiyon Açıklaması
Doğrusal Fonksiyon		Fonksiyonun grafiği doğrusal olduğundan, alınan türev değeri sabittir ve bu sebebple istenen öğrenme işlemi gerçekleşemez. Girdi katmanı ile çıktı katmanı arasında aynı lineerlik değeri söz konusudur.
Basamak (Adım) Fonksiyonu		Fonksiyon çıktısı iki farklı değer alabilir. Çalışma kapsamında iki farklı sınıflama yapılacağı zaman tercih edilebilir. Türev ile öğrenme değeri oluşamadığından dolayı gizli katmanlardan ziyade çıktı katmanlarında kullanılır.
Sigmoid Fonksiyonu		Aktivasyon fonksiyonu olarak en sık kullanılan fonksiyondur. Girdi verilerini (O, 1) aralığına dönüştürerek çıktı oluşturur. Türev işlemine sokularak algoritmanın öğrenme işlemi gerçekleştirilebilir.
Hiperbolik Tanjant Fonksiyonu		(-1, +1) aralığında çıktı bilgisi oluşturur. Türevlenebilirdir. Daha fazla değer alabilir. Yüksek öğrenme hızı ve geniş bir aralıkta sınıflama işlemini gerçekleştirdiğinden dolayı daha verimli bir çalışma sunabilmektedir.
ReLU		[O, cx:ı) aralığında değer üretir. C girdi bilgisini ifade eder. Doğrusal bir fonksiyon değildir ve tahmin başarısı yüksektir. Negatif bölgede sıfır değeri oluşturulmakta ve ağın çalışma hızıbu şekilde arttırılmaktadır. Özellikle gizli katmanları bulunan ağlarda tercih edilir.

Çizelge 2. 1. Yapay sinir ağı modellerinde kullanılan aktivasyon fonksiyonları (Aki, 2019)(devam)

Leaky ReLU		Fonksiyonun tanım bölgesi eksi sonsuza doğru uzanmaktadır. Fonksiyonun kapsadığı negatif bölgedede algoritmanın öğrenmesi devam etmektedir.
Swish Fonksiyonu		Negatif bölgede meydana gelen veriler doğrusallık oluşturmamaktadır. Diğer fonksiyonlara nazaran daha tekdüze bir yapısı vardır. Yumuşak bir interpolasyona sahiptir.

Çizelge 2. 2. Aktivasyon fonksiyonlarının matematiksel ifadeleri (Aki, 2019)

Aktivasyon Fonksiyonu	Denklem	Aralık
Doğrusal Fonksiyon	$f(x) = x$	$(-\infty, \infty)$
Basamak Fonksiyonu	$f(x) = \begin{cases} 0\ için & x < 0 \\ 1\ için & x \geq 0 \end{cases}$	$\{0, 1\}$
Sigmoid Fonksiyonu	$f(x) = \sigma(x) = \frac{1}{e^{-x}}$	$(0, 1)$
Hiperbolik Tanjant Fonksiyonu	$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} x < 0 için & 0 \\ x \geq 0\ için & x \end{cases}$	$[0, \infty)$
Leaky (Sızıntı) ReLU	$f(x) = \begin{cases} x < 0 için & 0.01 \\ x \geq 0\ için & x \end{cases}$	$(-\infty, \infty)$
Swish Fonksiyonu	$f(x) = 2x\sigma(\beta x) = \begin{cases} \beta = 0 & için \quad f(x) = x \\ \beta \to \infty\ iç\,in & \mathrm{f}(x) = 2\max(0, x) \end{cases}$	$(-\infty, \infty)$

Sınıflandırma işlemlerinde sıkça kullanılan, lojistik regresyon olarak da bilinen sigmoid fonksiyonu, aşağıda verildiği şekilde tanımlanabilir.

$$\forall z \in R,\ f(z) = \frac{1}{1+e^{-z}} \in]0,1[\tag{2.2}$$

$$z = b + (px_1.w_1) + (px_2.w_2) + (px_3.w_3) + \cdots + (px_n.w_n) \tag{2.3}$$

Denklemde verilen *b* değeri yanlılık (bias) değerini, *px* değerleri görselin piksellere ayrılıp her bir pikselin gri ölçeklendirme skalası uygulanarak elde edilen 0-1 aralığında matematiksel değerini ifade eder. *w* (weight) değeri her bir pikselin kendine has olan ağırlığını, n değeri ise işlenecek olan toplam piksel sayısını ifade etmektedir.

Kayıp (loss function) ve maliyet (cost function) fonksiyonları

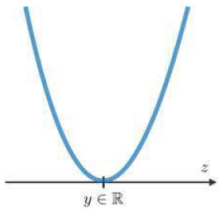
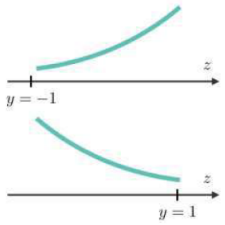
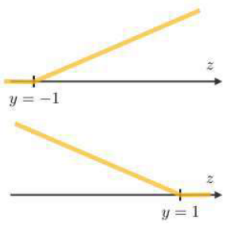
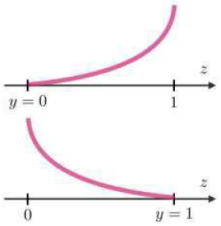
Kayıp ve maliyet fonksiyonları, sinir ağının eğitilmesinde ihtiyaç duyulan önemli fonksiyonlardandır. L: $(z,y) \in \mathrm{R} \times Y \rightarrow L(z,y) \in \mathrm{R}$ şekliyle ifade edilen kayıp fonksiyonu, aktivasyon fonksiyonu ile işlenen her bir görselden elde edilen tahmin değerinin gerçek değere olan farkından ortaya çıkan hata payını ifade eder. Bu hata payı, algoritmanın eğitilmesi için sisteme tekrar geri bildirilmek ve optimum *w*, *b* değerlerinin tespitini sağlamak amacıyla kullanılır. Başka bir deyişle kayıp ve maliyet fonksiyonları *w* ve *b* değerlerine bağlıdır. Her bir görsel için kullanılan bu fonksiyon aşağıdaki gibidir.

$$L = -(1-y) \cdot \log(1-\hat{y}) + 1+\log(\hat{y}) \tag{2.4}$$

Burada *y* değeri gerçek doğrunun matematiksel ifadesi, yani algoritmanın vermesi istenilen doğru çıktısıdır. $\hat{y}$ değeri ise algoritmanın verdiği çıktıdır. Sigmoid fonksiyonu üzerinden ele alındığında $\hat{y}$ değeri [0,1] aralığında olmaktadır. *y* değerinin 1 olması gerekirken $\hat{y}$ değerinin de 1 tahmin edilmesi durumunda kayıp fonksiyonunun sonucu sıfır çıkacaktır. Ancak algoritmanın yanlış tahmin çıktısı vermesi durumunda kayıp fonksiyonunun sonucu sıfırdan farklı bir değer olacak ve eğitimde kullanılan tüm

görseller için gerçekleştirilecek bu tahminlerin kayıp değerleri bu şekilde ortaya çıkacaktır. Yaygın olarak kullanılan kayıp fonksiyonları Çizelge 2.3'te verildiği gibidir.

Çizelge 2. 3. Yaygın biçimde kullanılan kayıp fonksiyonları (Amidi & Amidi, 2018)

En Küçük Kareler Hatası (Least squared error)	**Lojistik Kayıp (Logistic loss)**	**Menteşe Kaybı (Hinge loss)**	**Çapraz Entropi (Cross entropy)**
$\frac{1}{2}(y-z)^2$	$\log(1+\exp(-yz))$	$\max(0,1-yz)$	$-[y\log(z)+(1-y)\log(1-z)]$
			
Lineer Regresyon	Lojistik Regresyon	SVM	NN

Maliyet fonksiyonu ise her görselin verisinden elde edilen kayıp fonksiyonlarının toplamının görsel adedine bölünmesi ile hesaplanır. Bu da, sinir ağını eğitirken kullanılan tüm girdi verilerinin algoritma üzerinden bir kez işlenmesi demektir. Maliyet fonksiyonunun denklemi aşağıda gösterildiği gibidir.

$$J(\theta) = \sum_i^m L(h_0(x^{(i)}), y^i) \tag{2.5}$$

Denklemde verilen;

L: kayıp fonksiyonu,

h_0: modelin hipotezini,

$x^{(i)}$giriş kümesini,

$y^{(i)}$ çıkış kümesini belirtmektedir.

Elde edilen maliyet değeri ne kadar büyükse w ve b değerleri de o kadar hatalı seçilmiş anlamına gelir. Bu maliyet değeri ilk iterasyonda genellikle yüksek çıkmakta ardından iterasyon sayısına paralel bir biçimde düşüşe geçmektedir. Hedef, maliyet değerinin

olabildiğince küçültülmesi ve algoritmada belirlenen tolerans değerinin altına indirilmesidir. Bu amaçla ileri yayılım esnasında kullanılan fonksiyonların türevi alınacaktır.

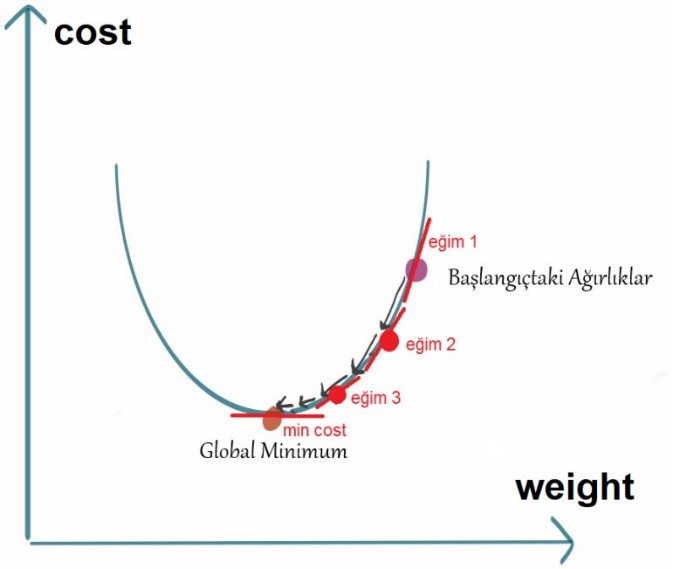
2.2.2. Geri yayılım (backward propagation)

Bu işlem, maliyet fonksiyonundan elde edilen hata değerinin tolerans sınırları içerisine çekilmesi için gerçekleştirilmektedir. Geri yayılım işlemi, ileri yayılım sürecinde kayıp fonksiyonlarının toplamı ile ortaya çıkan hata değerinin, ağ üzerinde çıktı katmanından terse doğru girdi katmanına ulaşıncaya dek çeşitli türev işlemlerine tabi tutularak geriye doğru yayılmasıdır. Bu süreç sonunda w_i ve b değerlerinin güncellenmesi sağlanır. Geri yayılım fonksiyonunun gerçekleştirilmesinde genellikle "gradyan iniş" metodu kullanılmaktadır.

Gradyan iniş (gradient descent) metodu

Optimizasyon, değişkenlerin problemdeki en iyi sonucu verecek değerini bulmak amacıyla uygulanan metotlara denir. Gradyan iniş metodu da geri yayılım esnasında algoritmanın optimizasyonu için kullanılan bir metottur.

Gradyan iniş metodu, türevlenebilir bir fonksiyonun global veya yerel minimumunu bulmak için birinci dereceden yinelemeli bir optimizasyon algoritmasıdır. Bu metot, değişkenlerin ilk değerlerinin rastgele alınması ile başlayıp global minimum değerine ulaşmayı amaçlar. Temel olarak, ileri yayılım esnasında gerçekleştirilen adımların tersten başlanarak sırasıyla türevlerinin alınması sonucu w_i ve b değişkenlerinin her bir iterasyonda tekrar güncellenmesini sağlar. Bu sayede gerçekleşen her bir iterasyonda oluşacak olan maliyet ve hata payları minimize edilmiş olur. Yeterli iterasyon sonucunda maliyet ve hata payları belirlenen tolerans değerlerinin altına iner ve değişkenlerin optimum değerleri tespit edilmiş olur. Bu metot Şekil 2.8'de verildiği gibidir.



Şekil 2. 8. Gradyan iniş metodu grafiksel gösterimi (Akca, 2020)

Şekilde de görülebileceği üzere hedef, cost fonksiyonu üzerindeki global minimum değerine denk gelen w_i değerlerini tespit etmek ve bu sayede algoritmanın optimum değerlere ulaşmasını sağlamaktır. Bunu gerçekleştirmek için cost fonksiyonu üzerinde belirlenen w_i değerlerinin grafik üzerindeki eğimleri tespit edilir. Bu eğim değeri ne kadar yüksek ise w_i değeri o kadar hatalıdır. Amaç, fonksiyonun w_i değerindeki eğimine göre türevinin alınması ve bu işlem sonucunda global minimum noktasına ulaşmaktır.

$$w \leftarrow w_1 - step \tag{2.6}$$

$$\frac{\partial L(\mathrm{p,y})}{\partial \mathrm{w}} = \frac{\partial L(\mathrm{p,y})}{\partial a} \cdot \frac{\partial La}{\partial p} \cdot \frac{\partial p}{\partial \mathrm{w}} \tag{2.7}$$

$$w \leftarrow w_1 - a.\frac{\partial L(\mathrm{w,b})}{\partial(\mathrm{b})} \tag{2.8}$$

Verilen denklemlerde w_i değerinin fonksiyona göre güncellenmesi gösterilmektedir. Burada step ifadesi, w_i değerinin cost fonksiyonunu dik kestiği eğim değeridir. Bu eğim değeri ilk veya önceki iterasyondan kalan w_i değerinden çıkarılarak bu değerin güncellenmesi sağlanmaktadır. Amaç cost fonksiyonunun minimum olduğu kısımdaki

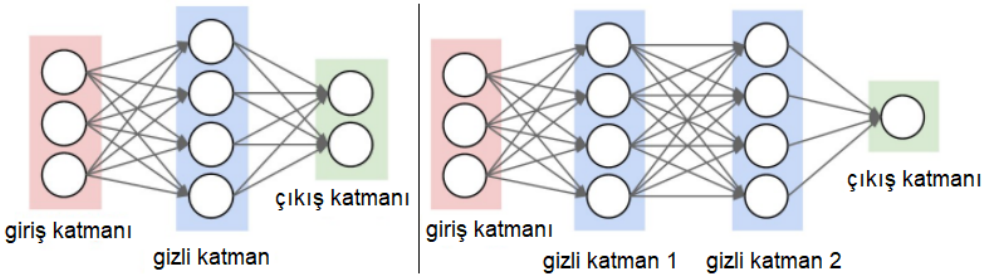
$w_i$değerini tespit etmektir. Yeni w_i değeri elde edildiğinde ileri yayılım işlemi tekrar gerçekleştirilmekte ve fonksiyon üzerinde toleransı belirlenmiş global minimum değerine ulaşılıncaya kadar işlem döngüsü devam etmektedir. Burada w_1 olarak ifade edilen değer, ilk *w* değeridir. Diğer denklemde ise cost fonksiyonunun w_i değerine göre türevinin ilk *w* değerinden çıkarılması gösterilmektedir. Denklemdeki *a* değeri öğrenme hızı (learning rate) olarak tanımlanabilir. Bu değer bir katsayıdır. Öğrenme hızı değeri olması gerekenden düşük olursa öğrenme işlemi esnasında optimum noktaya ulaşmak uzun sürebilir ve çokça iterasyon gerektirebilir. Hız değerinin gereğinden büyük olması durumunda ise (bu değer bir katsayı olduğu için ve eğim değeriyle çarpılacağı için için) bazen optimum nokta hiç bulunamayacak düzeye, yani belirlenen tolerans değerinin altına inemeyecek duruma gelebilir. Bu sebeple öğrenme hızı katsayısı için de optimum bir değer belirlenmelidir.

Son olarak, basit bir yapay sinir ağı modelinin görsel tahmin yürütme yöntemi esnasında gerçekleşen işlem basamakları 2.9'da verilmiştir.

Şekil 2. 9. Basit bir sinir ağı modelinin görsel tahmin yürütme işlem basamakları

2.3. Çok Katmanlı ve Evrişimli Sinir Ağları (CNN)

Yapay sinir ağlarında tek katman bulunabileceği gibi çok katmanlı veya gizli katmanlı yapılara da sahip olabilirler. Şekil 2.10'da tek katmanlı ve çok katmanlı sinir ağı yapısı örneği verilmiştir.



Şekil 2. 10. Tek katmanlı ve çok katmanlı sinir ağı yapısı (Doğan, 2020)

Şekilde de görülebileceği üzere aynı katman içindeki nöronların birbirleriyle ilişkileri yoktur. Farklı katmanlardaki nöronların arasındaki tüm yapısal bağlar bir ağırlık değeri ile ilişkilendirilir. Bu değer, girdi verisinin önemini ve sisteme etkisini belirtir. İlk ağırlık değerleri gelişigüzel bir biçimde ayarlanır. Tüm nöronlar bünyelerinde birer aktivasyon fonksiyonu bulundurmaktadırlar. Tüm nöronların kullandığı fonksiyonlar aynı olabileceği gibi farklı katmanlarda farklı fonksiyonlar da kullanılabilmektedir. Aktivasyon fonksiyonunun amacı her bir nörondan elde edilen bilgiyi standart bir hale dönüştürmektir. Bir veri kümesi, ağın bütün katmanlarından geçirilerek çıktı katmanına sonuç bilgisi olarak ulaşır. Uygulamada kullanılan katman sayısı işlem hacmini de arttırdığından katman sayısı uygulama için optimum değerde tutulmalıdır.

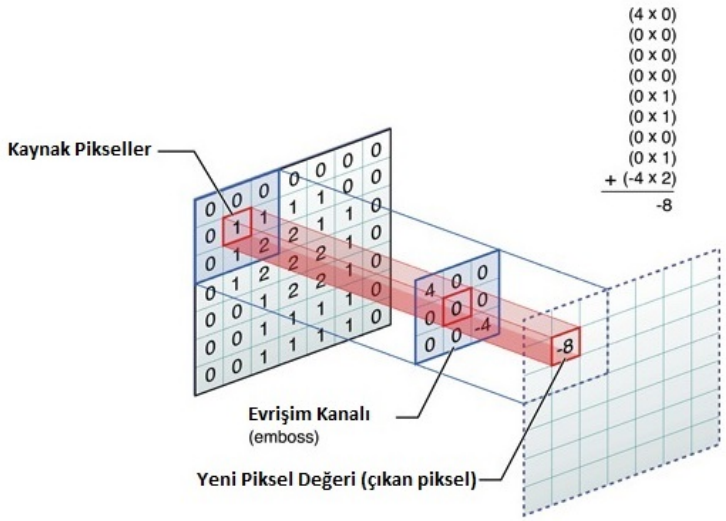
Şekil 2.10 incelendiğinde, sol taraftaki tek katmanlı yapay sinir ağı modelinde giriş katmanları hariç 4 + 2 = 6 nöron (x) bulunmaktadır. [3 x 4] + [4 x 2] = 20 weight, ve 4 + 2 = 6 yanlılık değeri olmak üzere toplamda 26 adet öğrenilmesi gereken parametre vardır.

Yine Şekil 2.10'da bulunan sağ taraftaki iki gizli katmanlı yapay sinir ağı modelinde ise 4 + 4 + 1 = 9 nöron, [3 x 4] + [4 x 4] + [4 x 1] = 32 weight ve 4 + 4 + 1 = 9 yanlılık değeri olmak üzere toplamda 41 adet öğrenilmesi gereken parametre vardır.

2.4. Evrişimli Sinir Ağında Kullanılan Katmanlar

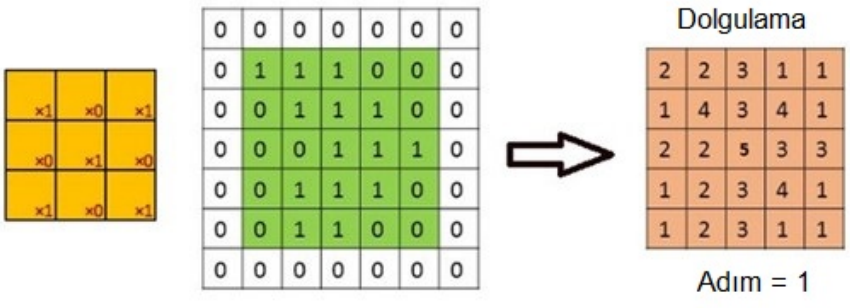
2.4.1. Evrişim katmanı (convolution layer)

Evrişimli sinir ağı modellerinde ilk katman olarak genellikle evrişim katmanı (convolution layer) bulunmaktadır ve görüntü ilk kez bu katmanda işleme tabi tutulmaktadır. Önceki başlıkta belirtildiği üzere görseller, algoritma içerisinde belirli değerler taşıyan piksellerden oluşan matrislere dönüştürülmektedirler. Evrişim katmanı ise görselin orjinal boyutlarından daha küçük bir filtre olarak görselin üzerinde gezer ve bu görsellerdeki objelerin belirli özelliklerini yakalamaya çalışır. Burada amaç algoritmaya verilen görüntü üzerinde kısım kısım filtrelemeler yaparak görseldeki nesnenin öne çıkan basit veya karmaşık şekillerden oluşan görsel niteliklerini tanımlamaktır. Bu işlem sonucunda elde edilen çıktı matrisi öznitelik haritası (feature map) olarak adlandırılmaktadır. Her bir özellik için farklı özellik algılayıcı (feature detector) katmana ve filtreye ihtiyaç duyulur. Örnek olarak kedi ve köpek verilerinin sınıflandırılması için, eğitim verisi üzerinden kedi ve köpeğin göz, kulak, kuyruk gibi özelliklerinin yavaş yavaş belirlenmesi bu katmanda meydana gelmektedir. Şekil 2.11'de evrişim kanalı üzerinde örnek bir filtre belirlenmiş ve belirlenen bu filtre üzerinden giriş katmanında alınan görüntünün piksel değerleri evrişim işlemine tabi tutulmuştur. Bu işlem sonucunda yeni bir piksel değeri türetilmiş ve aynı zamanda yeni oluşturulan matrisin daha küçük boyutlara sahip olması da sağlanmıştır. Daha küçük boyutlara sahip bir görüntünün ise algoritmanın çalışma hızına pozitif bir katkı sağladığı bilinmektedir.



Şekil 2. 11. Örnek bir filtre üzerinden evrişim işleminin gerçekleştirilmesi (Kapellmann-Zafra, 2013)

Görüntüdeki nesnenin özelliklerinin belirlenmesi amacıyla filtre uygulanması esnasında, matris boyutunun küçültülmesinin avantajlarının yanı sıra bazı dezavantajları da bulunmaktadır. Matris boyutu küçüldükçe görüntüde işlenmesi gereken bazı detaylar kaybolabilmekte ve çıkan matris her katmanda biraz daha küçülerek son katmanda neredeyse içinde özellik barındıramayacak hale gelebilmektedir. Bu durumu engellemek amacıyla evrişim işlemi esnasında dolgulama (padding) işlemi gerçekleştirilmektedir. Bu işlem, görüntünün matrisine adeta bir çerçeve olacakmış gibi dört taraftan sıfırlar eklenmesini sağlar. Uygulanan filtrenin boyutuna göre sıfır eklenen katmanlar artırılabilir. Bu sayede bir yandan matrisin karmaşık yapısı evrişim işlemi ile düzenlenirken diğer yandan filtre uygulanması sonucunda matrisin boyutunun sürekli olarak küçülmesine engel olunmuş olur. Bu işlem Şekil 2.12'de örnek olarak gösterilmiştir.



Şekil 2. 12. Evrişim katmanında gerçekleştirilen örnek dolgulama (padding) işlemi (Anonim, 2020)

Dolgulama (padding) işlemi sonucu meydana gelen çıktı boyutu, verilen denklemdeki gibidir.

$$y = (x - f + 2s)/k + 1 \tag{2.9}$$

Denklemde verilen;
y: çıktının boyutunu,
x: girdi verisinin boyutunu,
f: seçilen filtrenin boyutunu,
s: sıfır ekleme işlemini,
k: kaydırma işlemini tanımlamaktadır.

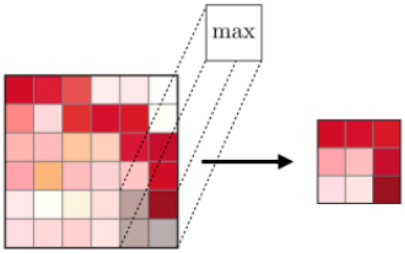
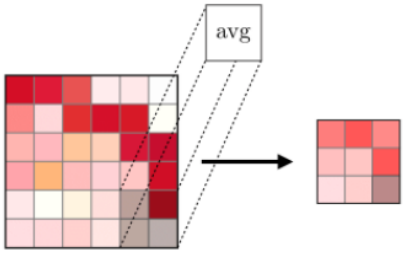
Evrişim işlemi esnasında kaydırma (adım, stide) değeri de önem arz etmektedir. Bu değer, filtrenin görüntü matrisi üzerinde nasıl hareket ettiğini denetler. Şekil 2.11 incelendiğinde bu değer bir piksel olarak alınmıştır ancak daha büyük de seçilebilir. Bu özellik, öznitelik haritasının yani çıkan matrisin boyutunu etkiler.

2.4.2. Ortaklama katmanı (pooling layer)

Evrişimli sinir ağı modellerinde, evrişim katmanının ardından ortaklama katmanına (pooling layer) geçiş yapılır. Ortaklama katmanının amacı, evrişim katmanında görüntüden elde edilen özelliklerin daha da derinine inilmesi ve spesifik hale

getirilmesidir. Aynı zamanda bu işlemle birlikte, nitelenen özelliğin matris boyutunun küçültülmesi de mümkün olmaktadır. Katmanın matris boyutunun küçülmesi işlem hızını arttırmakta ve gerçek zamanlı çalışmalarda daha hızlı sonuçlar elde edilmektedir. Ayrıca bu katman, algoritmada oluşabilecek aşırı öğrenme (overfitting) durumu da engelleyici bir unsurdur. Ortaklama katmanına örnek olarak, kedi ve köpek görüntülerinin sınıflandırması için evrişim katmanında matrisler üzerinden kulağın ayırt edici bir özellik olduğu tespit edilmekte, ortaklama katmanında ise kulağın bulunduğu matris kısmının parça parça matematiksel biçimde ortalama veya maksimum değeri alınarak tek bir değere dönüştürülmektedir. Bu sayede bir taraftan matris küçültme işlemi gerçekleştirilirken diğer taraftan o matristen elde edilen değer, dikkate değer bir kayba uğratılmadan kalabilmektedir. Bu işlem temel olarak maksimum ve ortalama ortaklama olmak üzere ikiye ayrılmaktadır. Bu işlemler, filtre uyguladığı kısımdaki matris değerlerinin sırasıyla maksimum veya ortalama değerin alındığı özel ortaklama türleridir. Bu yapılar Çizelge 2.4'te verildiği gibi değerlendirilebilir.

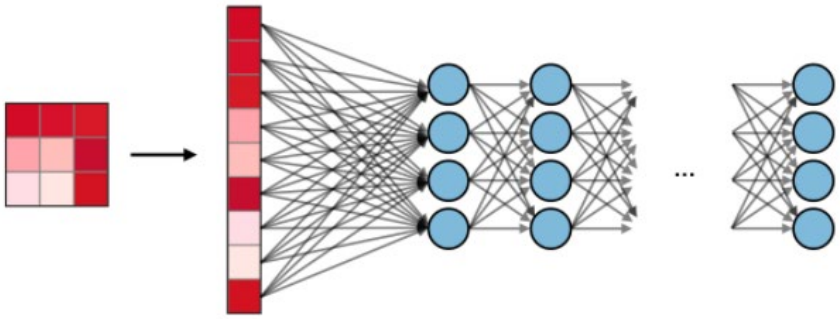
Çizelge 2. 4. Ortaklama katmanında gerçekleştirilen maksimum/ortalama ortaklama işlemi (Amidi S. & Amidi A., tarih yok).

Tip	Maksimum ortaklama	Ortalama ortaklama
Amaç	Her ortaklama işlemi, geçerli matrisin maksimum değerini seçer	Her ortaklama işlemi, geçerli matrisin değerlerinin ortalaması alır
Görsel Açıklama		
Açıklama	• Algılanan özellikleri korur • En çok kullanılan	• Boyut azaltarak örneklenmiştelik öznitelik haritası • LeNet'te kullanılmış

Evrişimli sinir ağlarının içerdiği matrisler, verinin durumuna da bağlı olarak ne kadar çok evrişim katmanı ve ortaklama katmanına tabi tutulursa o kadar iyi sınıflandırma yapılabilmektedir

2.4.3. Flattening katmanı

Bu katman temel olarak evrişimli sinir ağı modelinin sonuna doğru kullanılır ve amacı tam bağlantılı katmanın (FCL) girişine uygun şekilde verileri hazırlamaktır. Genel olarak sinir ağları, giriş verilerini tek boyutlu bir diziden alır. Buradaki katmanın amacı da evrişim ve ortaklama katmanından geçen matrislerin tek boyutlu diziye dönüştürülmesidir. Bu katmanda gerçekleştirilen işlem örnek olarak Şekil 2.13'te gösterilmiştir.

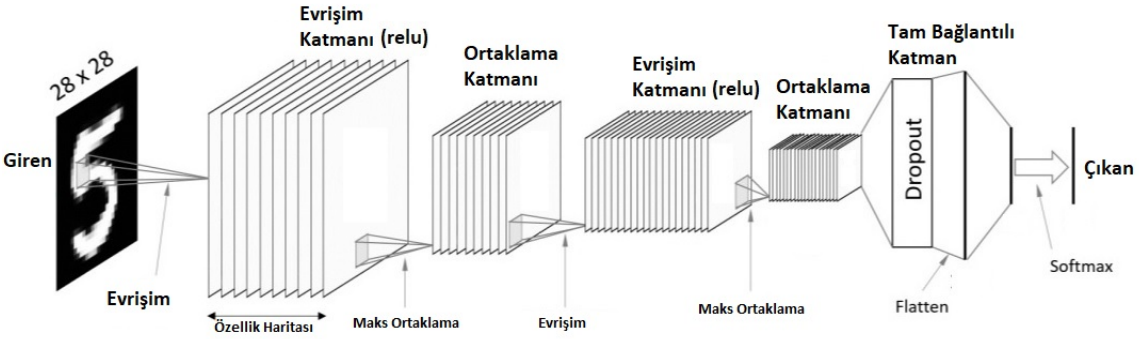


Şekil 2. 13. Flattening katmanında gerçekleştirilen, matrisin tek boyutlu diziye dönüştürülme işlemi (Amidi S. & Amidi A., tarih yok)

2.4.4. Tam bağlantılı katman (fully connected layer)

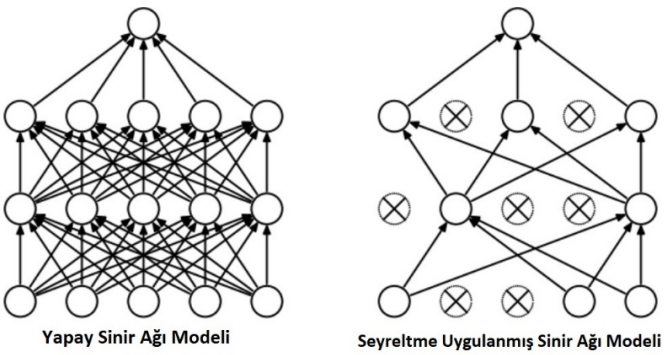
Veri setine bağlı olarak birkaç kez evrişim katmanından ve ortaklama katmanından geçen ve sonrasında flattening ile tek sütunlu bir matris haline gelen veri kümesi düz bir vektöre dönüşür. Oluşan bu veri kümesi tam bağlantılı katmana giriş verisi olarak kullanılacaktır. Bu katman evrişimli sinir ağı modellerinin son katmanıdır. Tam bağlantılı katman, adından da anlaşılacağı üzere her giriş değerinin tüm nöronlara bağlı olduğu bir yapı şeklinde çalışır. Tam bağlantılı katmanın görevi, klasik yapay sinir ağlarındaki çalışma mantığını gerçekleştirmektir.

Üstteki kısımlarda da belirtildiği yapıda bir işlem adımına sahip olan evrişimli sinir ağı modelinin üzerine farklı katmanlar eklenebilir olsa dahi başlıca katmanlarını da kapsayacak şekilde en temel yapısı Şekil 2.14'te verildiği gibidir.



Şekil 2. 14. Evrişimli sinir ağı modelinin temel katman yapısı (Anonim, 2020)

Evrişimli sinir ağı modellerinde ayrıca seyreltme (dropout) işlemi de uygulanmaktadır. Bu işlem, sinir ağı modelinde ileri yayılım işlemi gerçekleştirilirken sistem içerisinde rastgele bir şekilde nöronları devre dışı bırakır. Rastgele nöronların devre dışı kalması sonucunda her verinin tüm nöronlara ayrı ayrı ulaşması engellenerek algoritmada oluşabilecek aşırı öğrenme riski de ortadan kaldırılmış olur. Şekil 2.15'te sinir ağı modeli üzerinde örnek bir seyreltme işlemi gösterilmiştir.



Şekil 2. 15. Evrişimli sinir ağında gerçekleştirilen seyreltme (dropout) işlemi (Haverford, tarih yok)

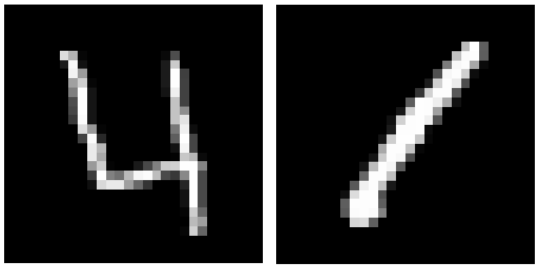
Temel olarak evrişimli sinir ağı modellerinde öğrenme işlemi filtrelerde gerçekleşmektedir. Kaldı ki, başlangıçta algoritma hangi görüntüye nasıl bir filtre uygulayacağını da bilmemektedir. Basit yapay sinir ağı modelinde $w_i$ve b değerleri öğrenme işlemine tabi tutulup sürekli güncellenirken, evrişimli sinir ağı modellerinde öncelikle kullanılması gereken filtrelerin nasıl bir matris yapısına sahip olması gerektiği öğrenilmektedir. Filtrelerin elde edilmesi ve uygulanması sonucunda flatten işlemi ile tam bağlantılı katmana geçiş yapılır ve ardından algoritma, yapay sinir ağı modeli üzerinden görüntüyü tahmin eder veya elde edilen görüntüye bağlı olarak önceden belirlenen komutları işletir.

Evrişimli sinir ağı modelleri üzerinde sürekli farklı çalışmalar yapılmakta ve bu modeller sürekli olarak geliştirilmektedir. Son yıllarda en çok bilinen bazı evrişimli sinir ağı mimarileri şu şekilde listelenebilir (Rubik's Code, 2018):

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet

2.5. Evrişimli Sinir Ağı Modeli Çalışma Örneği

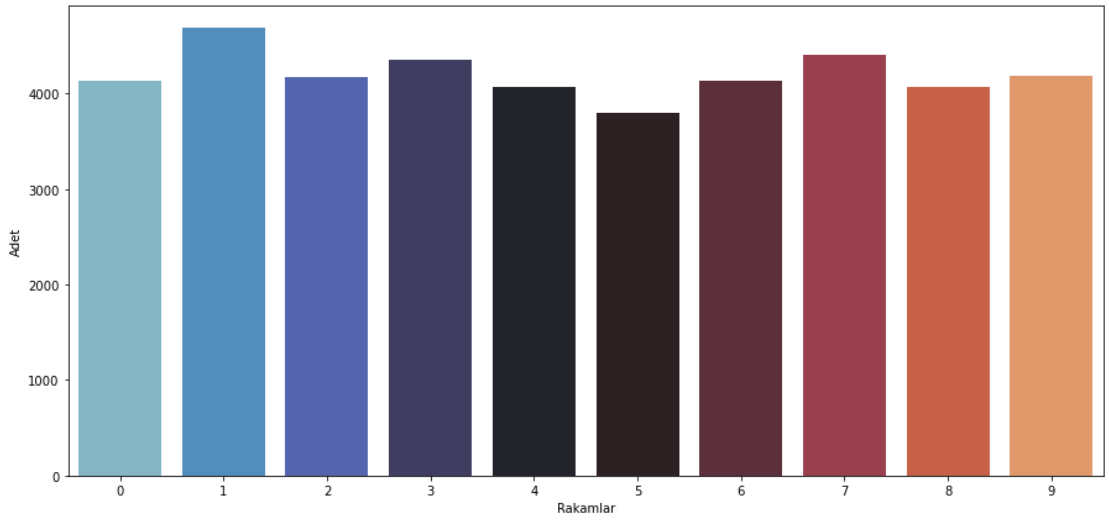
Bu başlık altında, evrişimli sinir ağı modeli oluşturulmuş olup, basit bir uygulamada üzerinden algoritmanın çalışma gösterimi gerçekleştirilecektir. Yapılan çalışma, kaggle platformunda bulunan bir veri setinin kullanılması ile gerçekleştirilmiştir. Kaggle platformu aslen makine öğrenmesi, yapay zeka, istatistik ve matematik gibi bilimleri bir arada toplayan bir oluşumdur. Ancak son dönemde paylaşımların artması ile eğitim veya veri seti dökümanlarının da güvenli bir şekilde edinilebileceği bir web platformu haline gelmiştir. Veri setinde elle çizilmiş 0-9 aralığındaki tüm rakamlar belirli bir adette bulunmaktadır. Bu veri setinden hareketle, alınan verinin öğrenilmesi ve oluşturulan evrişimli sinir ağının doğruluk testi test görselleri üzerinden gerçekleştirilmiştir. Şekil 2.16'da kullanılan görsel veri setinden örnekler verilmiştir.



Şekil 2. 16. Kullanılan görsel veri setinden 1 ve 4 rakamlarının görselleri

Veri setinde kullanılan rakamların toplam adedi 70,000'dir. 42,000 adedi algoritmanın eğitilmesi için kullanılırken 28,000 adedi ise algoritmanın test edilmesi amacıyla kullanılmak üzere ayrılmıştır. Çizelge 2.5'te, eğitim görselleri içerisinde 0-9 arasındaki rakamların ayrı ayrı adedi verilmiştir.

Çizelge 2. 5. Eğitim görselleri içerisinde 0-9 arasındaki rakamların adetleri



Rakamlar arasındaki sınıflandırma için oluşturulan evrişimli sinir ağı modeli ve algoritma kodları aşağıda verildiği gibidir. Her resim 28*28'lik piksel yani matris boyutunda kullanılacaktır. Uygulamada Keras sinir ağı kütüphanesi kullanılmıştır.

Kütüphanelerin eklenmesi ve gereksiz uyarıların temizlenmesi.

```
import numpy as np # lineer cebir
import pandas as pd # veri işleme, CSV dosyası vb. (pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Veri setlerinin yüklenmesi ve incelenmesi.

Algoritma eğitim verileri: Satır sayısı 42000, sütun sayısı (28*28)+1 = 785 çıkmaktadır. Sütun sayısındaki +1'in sebebi eğitim esnasında resmin hangi sayıya tekabul ettiğini belirten sınıfın da bu verilerin içinde bulunması.

```
train = pd.read_csv("train.csv")
print(train.shape)
train.head()
```

Test Verisi: Satır sayısı 28000, sütun sayısı 28*28 = 784 çıkmaktadır.

```
test= pd.read_csv("test.csv")
print(test.shape)
test.head()
```

Algoritma eğitimi veri setindeki sonucu veren etiket bilgilerinin (label) işlenecek olan veri setinden ayrılması.

Etiket bilgileri Y_train içerisine, kalan kısımlar X_train içerisine ayrılmaktadır. Etiket bilgisi, görselin gerçekte ifade ettiği rakamdır.

```
Y_train = train["label"]
X_train = train.drop(labels = ["label"],axis = 1)
```

Veriyi normalize etmek ve yazdırmak.

Normalizasyon işlemi ile resimdeki piksellerin renk değerleri 0-1 aralığındaki bir değere dönüştürülür. Bu işlem, algoritmanın resimleri çözümleme hızını arttırarak daha verimli çalışmasını sağlar. Bu amaçla gri ölçeklendirme (grayscale) işlemi uygulanır.

```
X_train = X_train / 255.0
```

```
test = test / 255.0
print("x_train shape: ",X_train.shape)
print("test shape: ",test.shape)
```

Veriyi, kütüphanenin istediği şekilde tekrar boyutlandırmak ve yazdırmak.

Keras kütüphanesi, kullanılacak resimlerin matrislerinin 3D olmasını istemektedir. Yani resimler 28*28*1 olarak tanımlanmalıdır.

```
X_train = X_train.values.reshape(-1,28,28,1)
test = test.values.reshape(-1,28,28,1)
print("x_train shape: ",X_train.shape)
print("test shape: ",test.shape)
```

“Label encoding” (etiket sınıflandırması) işleminin gerçekleştirilmesi.

“Label encoding” veriyi sayısallaştıran kısımdır. Temel olarak verilerin, yani kategorik değişkenlerin sayı formatına dönüştürülmesine yarar. “Label encoding” işlemi kısaca, eğitim verisindeki gerçek sonuç değerlerinin sayısal olarak ifade edilmesini sağlar. Bu işlem için Keras kütüphanesindeki “one hot encoding” işlemi kullanılmıştır. “One hot encoding”, gerçek sonuç çıktısı olarak tanımlanmış kategorik değişken adedi kadar sıfır ile vektör matrisi oluşturulması işlemidir. Yapılan örnek dikkate alındığında "3" rakamı için “label encoding” işlemi uygulanırsa [0,0,0,1,0,0,0,0,0,0] şeklinde bir sonuç elde edilir. 10 adet rakam kümesi için 10 uzunluğunda bir vektör matrisi oluşturulmuş ve 3. değer 1 olarak işaretlenmiştir.

```
from keras.utils.np_utils import to_categorical
Y_train = to_categorical(Y_train, num_classes = 10)
```

Eğitim verisinin kütüphanelere uygun biçimde tekrar matrislere ayrılması.

Bu basamakta eğitim verisi kütüphanenin işlemek için ihtiyaç duyduğu hale dönüştürülmüştür. En önemli kısım eğitim verisinin %10u ile, yine eğitim verileri vasıtasıyla oluşturulan algoritma, tekrar test edilmek için ayrılmaktadır. Rastgele seçim işlemi "2" olarak ayarlanmıştır.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1, random_state=2)
print("x_train shape",X_train.shape)
print("x_test shape",X_val.shape)
print("y_train shape",Y_train.shape)
print("y_test shape",Y_val.shape)
```

Evrişimli sinir ağı modelinin tanımlanması ve kullanılacak kütüphanelerin eklenmesi.

```
from sklearn.metrics import confusion_matrix
import itertools
from keras.utils.np_utils import to_categorical # one-hot
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop,Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
```

Genel model yapısının oluşturulması.

Bu modelde kullanılan katmanlar: conv => max pool => dropout => conv => max pool => dropout => fully connected (2 katman).

```
model = Sequential()
```

İlk evrişim katmanı.

Özellik sezinleyici (feature detector) filtrelerinin adedinin, filtre boyutunun, hangi dolgulama (padding) işleminin kullanılacağının ve aktivasyon fonksiyonunun belirlenmesi.

```
model.add(Conv2D(filters = 8, kernel_size = (5,5),padding = 'Same', activation ='relu', input_shape = (28,28,1)))
```

İlk ortaklama katmanı.

Ortaklama (pooling) katmanı boyutunun belirlenmesi ve seyreltme (dropout) için rastgelelik oranının girilmesi.

```
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

İkinci evrişim katmanı.

Özellik algılama (feature detector) filtre adedinin, filtre boyutunun, kullanılan dolgulama (padding) işleminin kullanılacağının ve aktivasyon fonksiyonunun belirlenmesi.

```
model.add(Conv2D(filters = 16, kernel_size = (3,3),padding = 'Same', activation ='relu'))
```

İkinci ortaklama katmanı.

Ortaklama katmanının ve boyutunun belirlenmesi, seyreltme (dropout) için rastgelelik oranının girilmesi.

```
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))
```

Tam bağlantılı katman (fully connected layer).

Matris vektör haline dönüştürülmüştür. Aktivasyon fonksiyonu Relu olan bir gizli katman eklenmiştir. Çoklu sınıflandırma yapılabilmesi için çıkış katmanında softmax fonksiyonu kullanılmıştır.

```
model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))
```

Öğrenme hızı (learning rate) optimize edici tanımlama.

Normalde sabit olarak tanımlanan öğrenme hızı değerinin algoritmanın ilerleyişine göre optimize edilmesi. Optimize işlemi için ADAM (adaptive momentum optimizer) kullanılmıştır.

```
optimizer = Adam(lr=0.001, beta_1=0.9, beta_2=0.999)
```

Modeldeki parçaların toplanması (compile işlemi).

Modelde kullanılan kayıp fonksiyonu "categorical cross entropy"dir. Keras kütüphasinin bir parçasıdır. İki veya daha fazla sınıf olduğunda çapraz entropi kaybı fonksiyonu kullanılması gerekmektedir. Çok sınıflı sınıflandırma problemlerinde genel olarak bu fonksiyon kullanılmaktadır. Burada optimize edici, kayıp fonksiyonu ve metrik birimi belirlenmiştir.

```
model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["accuracy"])
```

Hiper parametrelerin belirlenmesi.

Epochs and Batch size ın belirlenmesi. “Batch size”, her ileri-geri yayılım işleminde (forward-backward propagation), kaç görüntüyle bu işlemin tekrarlanacağını belirler. “Epochs size” ise, tüm görüntülerin ileri-geri yayılımının gerçekleşmesi sonucunda bir adet eğitim tur sayısı (epoch) gerçekleşmiş olur.

```
epochs = 50
batch_size = 250
```

Veri arttırma (data augmentation) işlemi.

Bu işlem, algoritmanın veriyi aşırı öğrenmesinin önüne geçilmesi ve daha doğru sonuçlar verebilmesini sağlar. Yapılan işlem, görüntüleri yakınlaştırma, uzaklaştırma, aynalama gibi farklı rotasyon işlemlerine tabi tutarak algoritmayı eğitmektir. Burada Keras küpüthanesinin bu işlem için üretilmiş bir metodu kullanılmıştır.

```
datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=5,  # rastgele görüntüleri 5 derece rotasyona uğratma
        zoom_range = 0.1, # rastgele görüntüleri %10 yakınlaştırma
        width_shift_range=0.1,  # rastgele 10% sağa-sola kaydırma
        height_shift_range=0.1,  # rastgele 10% yatay-dikey kaydırma
```

```
        horizontal_flip=False,
        vertical_flip=False)
datagen.fit(X_train)
```

Modelin eğitilmesi.

Keras kütüphanesini "fit generator" kodu kullanılmıştır. Eğitmin işlemi sonunda loss fonksiyonu değeri ve doğruluk oranını kullanıcıya vermektedir.

```
history = model.fit_generator(datagen.flow(X_train,Y_train, batch_size=batch_size),
epochs = epochs, validation_data = (X_val,Y_val), steps_per_epoch=X_train.shape[0] //
batch_size)
```

Modelin değerlendirilmesi.

Loss fonksiyonunun değişimi ve doğruluk oranını veren eğrinin oluşturulması ve eğrinin değerlendirilmesi.

```
plt.plot(history.history['val_loss'], color='b', label="loss")
plt.title("Test Verileri ile Elde Edilen Loss Değeri")
plt.xlabel("Epoch Sayısı")
plt.ylabel("Loss Değeri")
plt.legend()
plt.show()
```

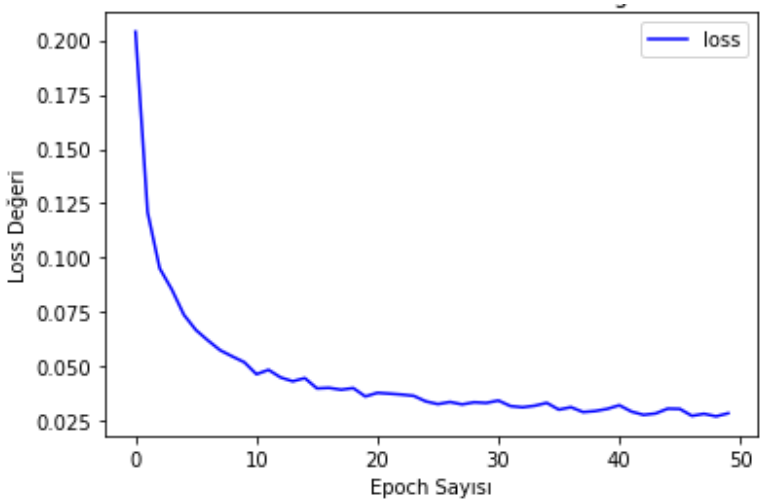
Hata matrisi (confusion matrix) oluşturma.

Hata matrisi temel olarak, sınıflandırma modellerinin performansını değerlendirmek için, hedef niteliğe ait tahminlerle gerçek değerlerin karşılaştırıldığı bir matristir. Bu matrisin oluşturulması için seaborn kütüphanesi kullanılmıştır. Kısaca, algoritmanın hangi sınıf görüntülerde daha çok yanlış sonuç verdiğini ve bu yanlış sonuçların doğru sınıflarının hangileri olduğunu göstermektedir.

```
import seaborn as sns
# Tahmin verilerinin indexlerini bulma
Y_pred = model.predict(X_val)
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Y_true verilerinin indexlerini bulma
```

```
Y_true = np.argmax(Y_val,axis = 1)
# Hata matrisini oluşturma
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# confusion matrix
f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx,annot=True,linewidths=0.01,cmap="Greens",linecolor="gr
ay", fmt= '.1f',ax=ax)
plt.xlabel("Tahmin Değerleri")
plt.ylabel("Doğru Değerler")
plt.title("Hata Matrisi")
plt.show()
```

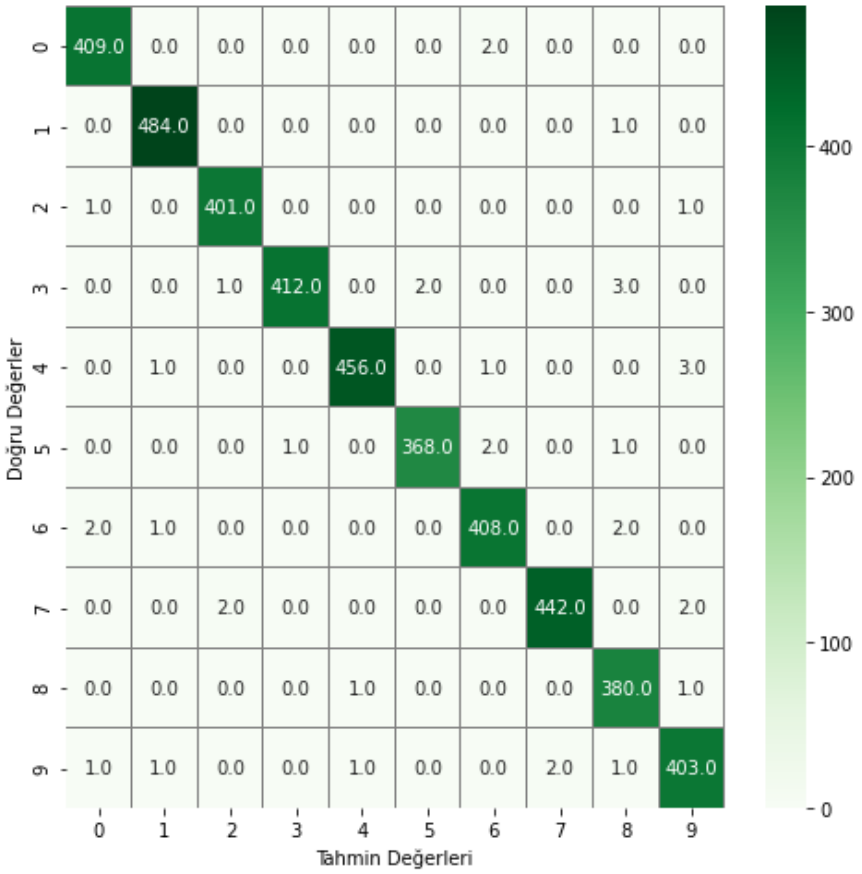
Sinir ağı uygulamasında eğitim tur sayısı (epoch) adedi 50 olarak girilmiştir. Çizelge 2.6'da her bir eğitim tur sayısı esnasında kayıp değerinin düştüğü ve algoritmanın doğruluk oranının arttığı gözlemlenmiştir.



Şekil 2. 17. Her bir epoch esnasında kayıp değerinin değişimi

Çizelge 2.6'de ise test görselleri üzerinden hangi rakamın kaç kez doğru tespit edildiği gösterilmiştir.

Çizelge 2. 6. Test görselleri üzerindeki hata matrisi



Yapılan çalışma sonucunda elde edilen kayıp (loss) 0,0807 değerine kadar düşmüş ve algoritmanın doğruluk oranı %97,51 olarak tespit edilmiştir. Yapılan eğitim tur sayısı (epoch) adedinin arttırılması sonucunda doğruluk oranı artmakla beraber istenen çözüm süresinin de arttığı gözlemlenmiştir.

3. MATERYAL VE YÖNTEM

Bu bölümde sürücüsüz aracın bulundurduğu donanımlar incelenecek, yazılımsal açıdan hangi platformların kullanıldığı ve hangi kütüphanelerden destek alındığı belirtilecektir. Aynı zamanda aracın parkur boyunca yol takibini gerçekleştirebilmesi için direksiyon açılarının ve hız kontrolünün belirlenme işlemi gerçekleştirilecektir. Parkur üzerinde gerçekleştirilen çalışmanın yan ısıra derin öğrenme yöntemlerine de değinilecektir.

Sürücüsüz araçların, bireysel ve ticari kullanıma onay almadan ve diğer araçların yerine bir alternatif haline gelmeden önce kendi içerisinde çözülmesi gereken problemleri vardır. Kamuya açık alanlarda sürücüsüz araçların güvenli bir şekilde hareket edemediği durumlarda güvenlik açısından endişe verici büyük zafiyetler ortaya çıkabilir. İnsanın öğrenme mantığında olduğu gibi, yapay sinir ağlarında da ne kadar farklı görüntü işlenirse ve eğitim gerçekleştirilirse gerçekleştirilsin hiçbir zaman %100 algılama başarısı elde edilemez. Ancak çalışmalardaki amaç, başarı oranını mümkün olduğunca yüksek seviyelere taşımaktır. Sürücüsüz araçların başlıca problemleri aşağıda sıralanmıştır.

- Şerit takibi esnasında şerit çizgilerini yakalayamaması ya da şeritlerin olmadığı/silik olduğu yollarda nasıl davranacağına karar verememesi,
- Aracın, kendisini istikrarlı bir şekilde şeride sabitleyememesi,
- Hızın artması durumunda direksiyon hakimiyetinin kaybedilmesi,
- Otonom sürüş ve algılayıcıların farklı yol koşullarından (otoban, arazi, şehir, farklı yerleşim yerleri) ve çevresel faktörlerden (farklı ışık seviyeleri, kar yağmur vb. ortamlar) etkilenmesi,

Belirtilen bu sorunlar her ne kadar karmaşık görünse de, tüm bu problemlerin çözümü için derin öğrenme tabanlı sistemler kullanılabilmektedir. Yapılan çalışma kapsamında bir taraftan araç ile önce manuel bir şekilde yol boyunca kamera vasıtasıyla görüntüler alınmış, diğer taraftan derin öğrenme algoritması geliştirilmiştir. Oluşturulan algoritmanın öğrenme başarısının arttırılması için bazı görüntü işleme yöntemleri ve farklı makine öğrenmesi kütüphanelerinden yararlanılmıştır. Alınan görüntüler üzerinden sinir ağı eğitimi gerçekleştirilmiş ve çalışma sonunda aracın belirlenen parkur üzerinde test sürüşü gerçekleştirilmiştir.

3.1. Araç Parkuru

Araç parkuru, araca bağlı bulunan kamera modülü vasıtası ile veri toplayacağı ve bu verilerin işlenmesi ile yol takibini sağlayacağı zemini oluşturmaktadır. Yapay zeka ve derin öğrenme çalışmalarının uygulanması amacıyla Waveshare tarafından üretilmiş olan bir parkur kullanılmaktadır. Parkurun eni iki metre, boyu ise üç metredir. Parkurda, trafik koşullarını simgelemesi amacıyla yol üzerinde engel bulunmaktadır. Araç parkuru Şekil 3.1'de verildiği gibidir.

Şekil 3. 1. Araç parkuru

3.2. Otonom Araç Mekanik Tasarımı

Radyo kontrollü araçlar, özel bir verici veya uzaktan kumanda kullanılarak kullanıcı vasıtasıyla kontrol edilebilen minyatür araç modelleridir. Bu araç modellerinden bazıları, gerçek araçlarda kullanılan ackerman prensibi gibi bazı özellikleri kendi bünyelerinde bulundurmakta ve bu sayede daha gerçekçi çalışmalar yapılmasını sağlamaktadırlar.

Mekanik tasarım başlığında, kullanılan araç şasesi, fiziksel donanımlar ve elektronik elemanların araç üzerine montajının yapılması, tercih edilen elemanların özellikleri ve tercih edilme sebepleri açıklanmıştır.

3.2.1. Kullanılan ekipmanlar

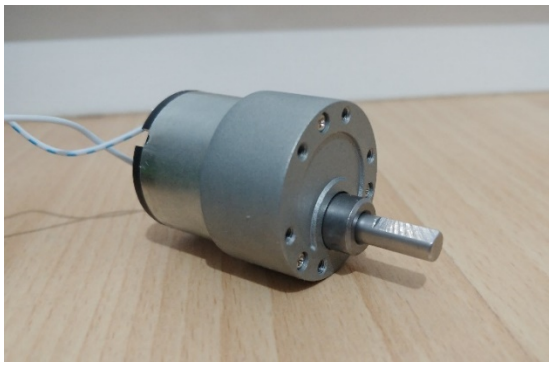
Kullanılan ekipmanlar temel olarak şunlardır:

- Araç şase ve sürüş bileşenleri
- Direksiyon mekaniğinin kontrolü için MG996R servo motor
- Araç hareketi için arka tekerlerde toplamda iki adet redüktörlü DC motor
- Nvidia Jetson Nano geliştirme kartı
- Raspberry Pi IMX219 kamera modülü
- Intel Wireless AC 8265NGW kablosuz haberleşme modülü
- Arduino Uno
- HC-SR04 Ultrasonik sensör
- Batarya amacıyla Panasonic NCR18650B 3400 mAh 3.7 V Li-Ion şarj edilebilir pil

Kullanılan tüm bu ekipmanların içerikleri, kullanım şekilleri ve birbirleriyle bağlantıları alt başlıklarda açıklanmıştır.

Araç şase ve sürüş bileşenleri

Çalışma kapsamında kullanılması için tercih edilen araç, Waveshare'in yapay zeka çalışmalarında kullanılması için üretmiş olduğu bir araç kitidir. Araç, kullanılan parkurda rahat hareket kabiliyetine sahip olmakla beraber üzerine yerleştirilen DC motor ile otonom sürüş esnasında yeterli çekiş gücüne ulaşmaktadır. Çalışma kapsamında iki adet DC motor kullanılmış olup, metal redüktörlü ve 150 rpm devre sahiptirler. Kullanılan motor Şekil 3.2'de gösterildiği gibidir.



Şekil 3. 2. Tez kapsamında kullanılan redüktörlü DC motor

Motorlar doğrudan arka iki tekere bağlanmıştır ve elde edilen güç herhangi bir aktarma organı bulunmadan doğrudan tekerlere iletilmiştir. Araçtaki direksiyon sistemi ackerman prensibi ile çalışmaktadır. Direksiyon kontrolünün otonom şekilde gerçekleştirilmesi için bir adet MG996R servo motor kullanılmıştır. Kullanılan servo motor Şekil 3.3'te verildiği gibidir.

Şekil 3. 3. MG996R servo motor

Geliştirme Kartı

Geliştirme kartı, bir ürünün kodlanmasını sağlayarak amaca uygun hareket edebilmesini sağlayan portatif bir elektronik kart olarak adlandırılabilir. Temel olarak işlemci, bellek ve depolama donanımlarından en az birine sahip devre kartlarına geliştirme kartı denir. Geliştirme kartları, robotik sistemlerinin işlevsellik kazanması, sensörlü projeler, kolay kontrol edilebilir cihazlar ve daha birçok projenin gerçekleştirilmesine imkan sağlarlar. Tez kapsamında Nvidia şirketinin ürettiği bir geliştirme kartı tercih edilmiştir. Nvidia, sunduğu geliştirme kartları arasında farklı GPU (grafik işlemci birimi), CPU (merkezi işlem birimi), bellek ve depolama değerlerine sahip birçok farklı ürünü geliştiricilerle buluşturmuştur. Şirketin geliştirmiş olduğu bu kartlar ve donanım özellikleri Çizelge 3.1'de verildiği gibidir.

Çizelge 3. 1. Nvidia Jetson geliştirme kartlarının donanım özellikleri (Nvidia, tarih yok)

	Jetson Nano	Jetson TX2 Serisi			Jetson Xavier NX	Jetson AGX Xavier Serisi	
		TX2 4GB	TX2	TX2i		AGX Xavier 8GB	AGX Xavier
Yapay Zeka Performansı	472 GFLOPS	1,33 TFLOPS		1,26 TFLOPS	21 TOPS	20 TOPS	32 TOPS
GPU	128 çekirdekli NVIDIA Maxwell™ GPU	256 çekirdekli NVIDIA Pascal™ GPU			48 Tensor Çekirdeğiyle Donatılmış 384 Çekirdekli NVIDIA Volta™ GPU	48 Tensor Çekirdeğiyle Donatılmış 384 Çekirdekli NVIDIA Volta™ GPU	64 Tensor Çekirdeğiyle Donatılmış 512 Çekirdekli NVIDIA Volta™ GPU
CPU	Dört çekirdekli ARM® Cortex®-A57 MPCore işlemci	İki çekirdekli Denver 1.5 64-bit CPU ve dört çekirdekli Arm® Cortex®-A57 MPCore işlemci			6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6MB L2 + 4MB L3	6-core NVIDIA Carmel Arm®v8.2 64-bit CPU 6MB L2 + 4MB L3	8-core NVIDIA Carmel Arm®v8.2 64-bit CPU 8MB L2 + 4MB L3
Bellek	4 GB 64-bit LPDDR4 25.6GB/s	4 GB 128-bit LPDDR4 51.2GB/s	8 GB 128-bit LPDDR4 59.7GB/s	8 GB 128-bit LPDDR4 (ECC Support) 51.2GB/s	8 GB 128-bit LPDDR4x 51.2GB/s	8 GB 256-bit LPDDR4x 85.3GB/s	32 GB 256-bit LPDDR4x 136.5GB/s
Depolama	16 GB eMMC 5.1 *	16 GB eMMC 5.1	32 GB eMMC 5.1	32 GB eMMC 5.1	16 GB eMMC 5.1	32GB eMMC 5.1	
Güç	5W | 10W	7.5W | 15W		10W | 20W	10W | 15W	10W | 20W	10W | 15W | 30W
PCIe	1 x4 (PCIe Gen2)	1 x1 + 1 x4 OR 1 x1 + 1 x1 + 1 x2 (PCIe Gen2)			1 x1 (PCIe Gen3) + 1 x4 (PCIe Gen4)*	1 x8 + 1 x4 + 1 x2 + 2 x1 (PCIe Gen3)	1 x8 + 1 x4 + 1 x2 + 2 x1 (PCIe Gen4, Kök Bağlantı Noktası ve Uç Nokta)
CSI Kamera	Up to 4 cameras 12 lanes MIPI CSI-2 D-PHY 1.1 (up to 18 Gbps)	Up to 6 cameras (12 via virtual channels) 12 lanes MIPI CSI-2 D-PHY 1.2 (up to 30 Gbps) C-PHY 1.1 (up to 41Gbps)			6 kameraya kadar (sanal kanallar aracılığıyla 24) 14 şerit (3x4 veya 6x2) MIPI CSI-2 D-PHY 1.2 (30 Gb/sn'ye kadar)	Up to 6 cameras (36 via virtual channels) 16 lanes MIPI CSI-2 | 8 lanes SLVS-EC D-PHY 1.2 (up to 40 Gbps) C-PHY 1.1 (up to 91 Gbps)	
Video Kodlama	1x 4K, 30 (HEVC) 2x 1080p, 60 (HEVC)	1x 4K, 60 (HEVC) 3x 4K, 30 (HEVC) 4x 1080p, 60 (HEVC)			2x 4K, 30 (HEVC) 6x 1080p, 60 (HEVC)	2x 4K, 30 (HEVC) 6x 1080p, 60 (HEVC) 14x 1080p, 30 (HEVC)	4x 4K, 60 (HEVC) 16x 1080p, 60 (HEVC) 32x 1080p, 30 (HEVC)
Video Kod Çözümü	1x 4K, 60 (HEVC) 4x 1080p, 60 (HEVC)	2x 4K, 60 (HEVC) 7x 1080p, 60 (HEVC) 20x 1080p, 30 (HEVC)			2x 4K, 60 (HEVC) 12x 1080p, 60 (HEVC) 32x 1080p, 30 (HEVC)	2x 4K, 60 (HEVC) 12x 1080p, 60 (HEVC) 32x 1080p, 30 (HEVC)	2x 8K, 30 (HEVC) 6x 4K, 60 (HEVC) 26x 1080p, 60 (HEVC) 72x 1080p, 30 (HEVC)
Ekran	2 çoklu mod DP 1.2/eDP 1.4/HDMI 2.0 1 x2 DSI (1,5 Gb/sn / şerit)	2 çoklu mod DP 1.2/eDP 1.4/HDMI 2.0 2 x4 DSI (1,5 Gb/sn /şerit)			2 çoklu mod DP 1.4/eDP 1.4/HDMI 2.0 DSI desteği yok	3 çoklu mod DP 1.4/eDP 1.4/HDMI 2.0 DSI desteği yok	
DL Hızlandırıcı	-	-			2x NVDLA Motoru	2x NVDLA Motoru	
Görüş Hızlandırıcı	-	-			7 Yönlü VLIW Görme İşleyici	7 Yönlü VLIW Görüntü İşlemcisi	
Ağ	10/100/1000 BASE-T Ethernet	10/100/1000 BASE-T Ethernet	10/100/1000 BASE-T Ethernet, WLAN	10/100/1000 BASE-T Ethernet	10/100/1000 BASE-T Ethernet	10/100/1000 BASE-T Ethernet	
Mekanik	69.6 mm x 45 mm 260 pimli kenar konnektörü	87 mm x 50 mm 400-pimli konnektör			69.6 mm x 45 mm 260 pimli kenar konnektörü	100 mm x87 mm 699-pimli konnektör	

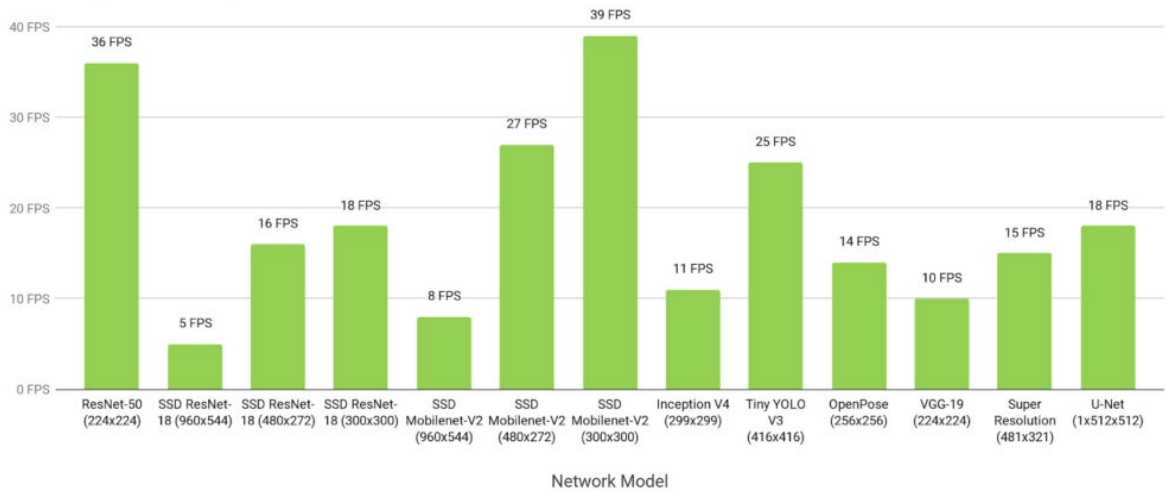
Geliştirme kartlarının bünyelerinde bulundurduğu donanımlar ve terimlerin açıklanması, kartların içeriklerinin daha net algılanmasını sağlayacaktır. CPU birimi, kartın üzerindeki tüm donanımları kontrol eden ve sistemin tümünü işleyen bir birimdir. Sistemin çalışma hızına etki eder ancak sistemin bellek boyutu ve depolama hızı gibi etmenler de bu hıza etki eden önemli kriterlerdendir. Yani kullanılan CPU'dan maksimum verimlilik alınması için sistemdeki diğer donanımların özelliklerinin de yeterli düzeyde olması gerekmektedir. GPU ise yalnızca grafik işlemlerini yapan işlemcidir. Temel olarak GPU'lar en hızlı şekilde yüksek çözünürlükteki görüntü ve videoları işlemek amacıyla tasarlanmışlardır. Aynı anda büyük bir veri kümesindeki paralel işlemleri, içerdikleri yüksek çekirdek adedi sayesinde CPU'lara kıyasla çok daha kısa bir süre içerisinde hesaplayabilmektedirler. Bu açıdan gerçek zamanlı görüntü işleme çözümlerinde CPU'lara nazaran çok daha hızlı geri dönüşler alınmaktadır. Ancak GPU'lar bu yüksek hızlarına rağmen yalnızca tek bir konuda, grafik verilerini işlemek konusunda, başarıya sahiptirler. Bu nedenle CPU gibi birçok farklı görevi aynı anda yapma özellikleri yoktur.

Geliştirme kartı adı altında üretilen Nvidia ürünleri, kabul görmüş diğer Raspberry Pi gibi donanımlardan daha güçlü grafik işlemcilerini bünyelerinde bulundurmakta ve gerçek zamanlı yapılan görüntü işleme projelerinde daha hızlı sonuçlar vermektedirler. Bu uygulamalarda ayrıca yüksek çözünürlük işleyebilme kabiliyeti sayesinde de daha doğru sonuçlara ulaşılmaktadır. Linux işletim sistemine sahip olan bu geliştirme kartları temel olarak otonom sistemlerin oluşturulması amacıyla üretilmişlerdir.

Yapılan tez çalışması kapsamında geliştirme kartı olarak Nvidia Jetson Nano modeli kullanılmıştır. Nvidia Jetson Nano, minimal boyutu sayesinde özellikle yapay zeka ve görüntü işleme çalışmalarında yüksek güç gerektiren işlemleri düşük maliyetle gerçekleştirebilmektedir. Yapay zeka alanında uygulama gerçekleştiren araştırmacılar, geliştirilen kartlar sayesinde özellikle görüntü işleme ve sınıflandırma, obje algılama ve ses işleme gibi çalışmaları yüksek maliyet ve büyük donanımlara ihtiyaç duymadan çok az bir elektrik gücüyle gerçekleştirebilmektedirler. Netson Nano geliştirme kartı, donanım maliyeti yüksek olan uygulamaları diğer kartlara nazaran çok daha verimli ve maliyetsiz çalıştırabilmektedir. Örnek olarak sürekli popülerleşen derin öğrenme ve yapay zeka uygulamalarını gerçekleştirmek için donanım maliyeti yüksek cihazlara

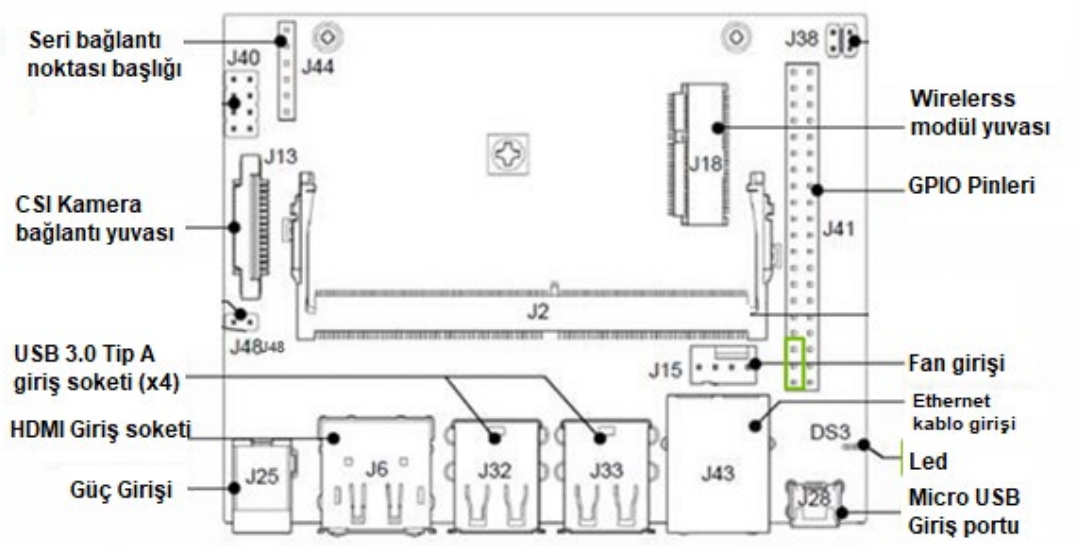
gereksinim oluşmaktadır. Nvidia Jetson Nano kartı ise yapay zeka çalışmaları doğrultusunda gerçekleştirilen yazılımsal optimizasyonlar ve donanımsal özellikleri sayesinde, diğer geliştirme kartlarının bir adım önüne geçmektedir. Ayrıca geliştirme kartlarında kullanılan derin öğrenme algoritmaları her geçen gün ilerlemekte ve çalışma performansları sürekli olarak arttırılmaktadır. Derin öğrenme algoritmalarının Jetson Nano bünyesindeki çalışma performansları Çizelge 3.2'de verildiği gibidir.

Çizelge 3. 2. Derin öğrenme algoritmalarının Jetson Nano üzerindeki performansı (Nvidia, 2019)



Nvidia Jetson Nano geliştirme kartı ve donanımsal bölümleri ise Şekil 3.4 ve Şekil 3.5'te verilmiştir.

Şekil 3. 4. Nvidia Jetson Nano geliştirme kartı (Nvidia, tarih yok)



Şekil 3. 5. Nvidia Jetson Nano donanım ve giriş/çıkış portları (Anonim, 2021)

Şekil 3.5 incelendiğinde, kartın ön yüzünde micro USB ve güç kablosu için giriş yuvalarının olduğu görülmektedir. Geliştirme kartına bu giriş portları üzerinden enerji sağlanmaktadır. Proje kapsamında Jetson Nano'ya enerji girişi pinler üzerinden gerçekleştirilmiştir. Ayrıca micro USB girişi ile cihaz, monitör gibi dış birimlere bağımlı kalmadan kullanılabilmektedir. Üzerinde bulunan HDMI girişi sayesinde kartı ufak bir bilgisayar şeklinde kullanabilme imkanını kullanıcıya sunmaktadır. Cihazın üzerinde ethernet girişi bulunmakla birlikte istenirse harici olarak eklenebilecek kablosuz (wireless) modül yuvası da bulunmaktadır. Ayrıca dört adet USB giriş portu bulunmakta ve bu portlar sayesinde farklı ve daha kapsamlı uygulamaların gerçekleştirilebilmesine imkan sağlamaktadır.

Jetson Nano, yüksek güç gerektiren işlemlerin gerçekleştirilmesine imkan sağlamakla beraber, oluşabilecek yükselen sıcaklık seviyesinin getirdiği donma, yavaşlama gibi sorunları ortadan kaldırmak amacıyla işlemci özerine yerleştirilebilecek soğutucu modül yuvası da oluşturmuştur. Çalışma esnasında soğutucu modül kullanılmıştır. Ayrıca üzerinde bulunan GPIO (genel amaçlı giriş/çıkış) pinleri sayesinde uygulanacak proje doğrultusunda step veya servo motor kontrolü, sensör uygulamaları ve diğer birçok işlemi gerçekleştirebilme kabiliyetine sahiptir. Jetson Nano GPIO pin şeması Şekil 3.6'da verildiği gibidir.

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC *Power*	1	2	5.0 VDC *Power*	
	I2C_2_SDA *I2C Bus 1*	3	4	5.0 VDC *Power*	
	I2C_2_SCL *I2C Bus 1*	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX */dev/ttyTHS1*	
	GND	9	10	UART_2_RX */dev/ttyTHS1*	
gpio50	UART_2_RTS	11	12	I2S_4_SCLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3 VDC *Power*	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_1_CS0	gpio19
	GND	25	26	SPI_1_CS1	gpio20
	I2C_1_SDA *I2C Bus 0*	27	28	I2C_1_SCL *I2C Bus 0*	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_PZ0	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio77
	GND	39	40	I2S_4_SDOUT	gpio78

Şekil 3. 6. Jetson Nano GPIO pin şeması (Kaloha, 2019)

Aynı zamanda ihtiyaç halinde haberleşme protokolleri de GPIO pinleri üzerinden gerçekleştirilebilmektedir.

Kamera modülü

Otonom aracın algoritma eğitimi aşamasında dış kaynaktan verilerin alınması, görüntülerin işlenmesi, şerit çizgileri üzerinden görüntü akışını sağlayabilmesi için harici kamera modüllerine ihtiyaç duyulur. Bu amaçla tez kapsamında kamera modülü olarak Raspberry Pi V2 IMX219 kamera modülü kullanılmıştır. 1080p çözünürlük imkanı sunan kamera, üzerinde Sony tarafından üretilen 8 megapiksel yüksek hassasiyetli ve hızlı video desteği sunan IMX219 görüntü algılayıcı sensör ile beraber çalışmaktadır. 160 derece diyagonal görüş açısını sunan kamera modülü, çevresini çok daha geniş biçimde algılayabilmektedir. Özellikle nesne tanımlama ve zaman atlamalı uygulamalarda tercih edilen bu model, 3280 x 2464 piksel görüntü alımı ve 30 kare 1080p, 60 kare 720p ve 90 kare 640x480p video kayıt desteği sağlamaktadır. Modül, dahili olarak bulunan görüntü kablosu vasıtasıyla Jetson Nano cihazının CSI konektörüne bağlanılarak kullanılabilmektedir. Bağlantı sonrası Linux API'leri üzerinden kameraya erişim ve kullanım mümkün hale gelmektedir. Şekil 3.7'de kullanılan kamera modülü gösterilmiştir.

Şekil 3. 7. Raspberrry Pi V2 ile uyumlu IMX219 kamera modülü (SeedStudio, tarih yok)

Raspberry Pi IMX219 Kamera Modülü Teknik Özellikleri:

- Sabit odaklı lens
- 8 megapiksel çözünürlüklü kamera
- 160 derece diogonal görüş açısı
- 3280 x 2464 piksel fotoğraf desteği
- 30 kare 1080p, 60 kare 720p ve 90 kare 640x480p video desteği

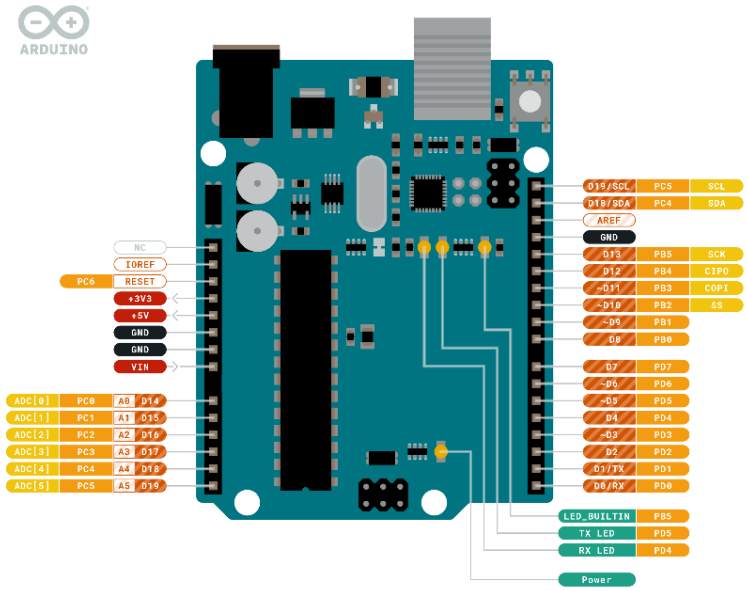
Kablosuz haberleşme modülü

Tez kapsamında, Jetson Nano'nun ana bilgisayar ile bağlantısının kurulması, programlanması ve dosya aktarımı gibi işlemlerin gerçekleştirilebilmesi adına kablosuz haberleşme modülüne ihtiyaç duyulmuştur. Bu amaçla tez kapsamında Intel Çift Bantlı Kablosuz AC 8265NGW modeli tercih edilmiş ve karta montajı gerçekleştirilmiştir. Kullanılan modül Şekil 3.8'de gösterilmiştir.

Şekil 3. 8. Intel Çift Bantlı AC 8265NGW Kablosuz Haberleşme Modülü (Intel, tarih yok)

Arduino Uno

Arduino Uno, Atmega328P mikro denetleyici çipine sahip açık kaynak kodlu bir mikro denetleyici kartıdır. Bu mikro denetleyici kartı, diğer devrelere bir arayüzle bağlanabilen dijital ve analog giriş/çıkış (I/O) pinleri ile donatılmıştır (Wikipedia, Arduino Uno, 2021). Bu pinlerin kart üzerindeki dağılımı Şekil 3.9'da verildiği gibidir.



Şekil 3. 9. Arduino Uno pin dağılımı (Arduino, tarih yok)

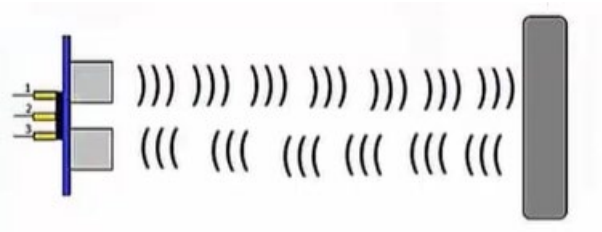
Arduino Uno teknik özellikleri aşağıda verildiği gibidir (Wikipedia, Arduino Uno, 2021).

- Mikrodenetleyici: Mikroçip ATmega328P
- Çalışma Gerilimi: 5 Volt
- Giriş gerilimi: 7 ila 20 Volt
- Dijital I / O pinleri: 14 (hangi 6 PWM çıkışı sağlayabilir)
- UART: 1
- I2C: 1
- SPI: 1
- Analog Giriş Pinleri: 6
- Her bir pin için DC Akım/O Pin: 20 mA
- 3.3 V pin için DC akım: 50 mA
- SRAM: 2 KB
- EEPROM: 1 KB
- Saattaki Hızı: 16 MHz
- Uzunluk: 68.6 mm
- Genişlik: 53.4 mm
- Ağırlık: 25g

Arduino Uno, ultrasonik mesafe sensörünün Jetson Nano üzerinden Python dili ile kodlanmasını sağlamak amacıyla kullanılacaktır. Bu kullanım için Arduino içerisine firmata kütüphanesi yüklenmiştir.

Ultrasonik mesafe sensörü

Ultrasonik mesafe sensörü, mesafe algılaması işlevini sonar iletişim yöntemi ile sağlamaktadır. Ultrasonik sensör ve sonar sistemi ses dalgalarını kullanarak cismin uzaklığının hesaplanmasına yardımcı olur. Temel olarak ultrasonik sensör, baktığı açıya ses sinyalleri gönderir. Sinyallerin bir yüzeye çarpıp yansıması ile ses dalgaları cihaza geri döner. Geri dönüş süresi üzerinden cihazın karşısındaki nesnenin uzaklığı tespit edilmiş olur.



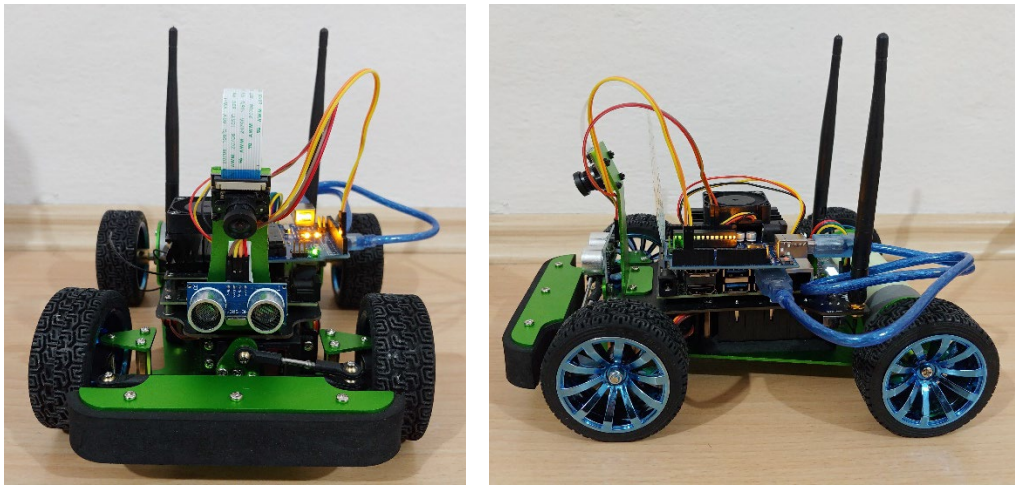
Şekil 3. 10. Ultrasonik mesafe sensörü çalışma prensibi

Sensör üzerinde vcc, gnd, trig, echo olmak üzere 4 adet pin mevcuttur. Çalışma kapsamında kullanılan ultrasonik sensör modeli HC-SR04'tür. Bu model Şekil 3.11'de verildiği gibidir.

Şekil 3. 11. Ultrasonik mesafe sensörü (RoboBlok, tarih yok)

3.2.2. Ekipmanların montajı

Diğer başlıklarda anlatılan tüm alt donanımların araca montajı gerçekleştirilmiştir. Aracın hareketinin sağlanması için arka tekerlerde iki adet redüktörlü DC motor, direksiyon kontrolü için ise bir adet MG996R servo motor araç üzerine yerleştirilmiştir. Şasenin hemen üzerinde üç adet Panasonic NCR18650B pil bulunmakta ve araç ile üzerindeki tüm donanımlar güç ihtiyacını buradan karşılamaktadırlar. Pil yuvasının üzerinde Jetson Nano geliştirme kartı, karta bağlı biçimde İntel haberleşme modülü ve kamera modülü aracın üzerinde konumlanmaktadır. Jetson Nano'nun üzerine Arduino Uno uygun bir şekilde sabitlenmiştir. Kamera ve ultrasonik mesafe sensörü ise aracın bakış hizasına göre konumlandırılmıştır. Jetson Nano geliştirme kartının ısınmasını engellemek adına soğutucu bloklarının üzerine de bir adet fan eklenmiştir. Aracın montaj sonrasındaki görüntüsü Şekil 3.12'de verildiği gibidir.



Şekil 3. 12. Tez kapsamında kullanılan otonom araç

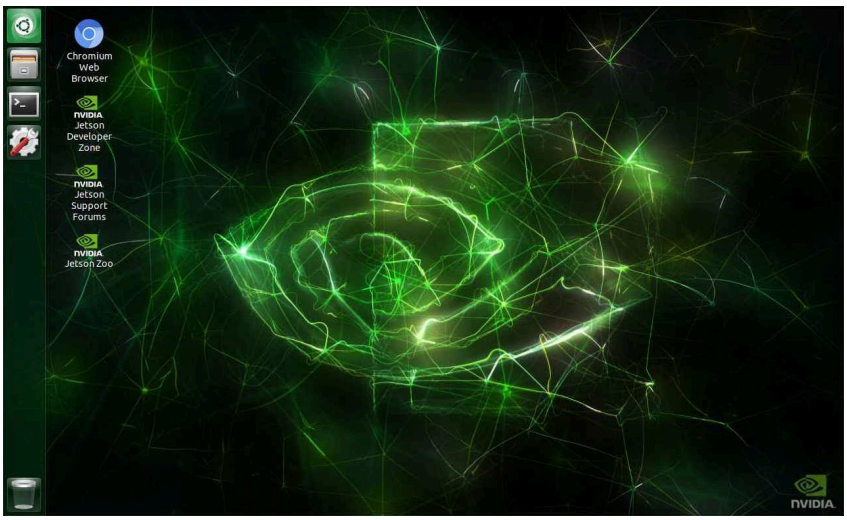
3.3. Otonom Araç Yazılım Tasarımı

Bu bölümde, otonom aracın oluşturulan parkur üzerinde alınan görüntüler sayesinde direksiyon açıları üzerinden hareketinin sağlanması için kullanılan programlardan, yazılım dillerinden ve makine öğrenmesi kütüphanelerinden bahsedilecektir.

Programlama öncesinde Jetson Nano geliştirme kartının, Intel AC 8265NGW kablosuz haberleşme modülü vasıtasıyla wifi ağına bağlantısı gerçekleştirilmiştir. Ardından ana bilgisayar üzerinden ağ bağlantısıyla cihazın IP adresi tespit edilmiş ve cihaza ait IP adresi sayesinde araç ana bilgisayar üzerinden kontrol edilebilir ve kodlanabilir hale getirilmiştir. Gerçekleştirilen bağlantı ve cihazın kablosuz internet bağlantısındaki IP adresi Şekil 3.13'te verildiği gibidir. Şekil 3.13'te Jetson Nanonun IP adresi bilgisi gösterilmiştir. Ana bilgisayar tarafından web tarayıcısı ile, bu IP adresi üzerinden cihaz kontrolü sağlanabilmektedir.

Şekil 3. 13. Ana bilgisayara Jetson Nano'nun IP adresinin girilmesi

Aracın üzerinde bağlı olan Jetson Nano geliştirme kartına işletim sistemi olarak Ubuntu yüklenmiştir. Ubuntu işletim sistemi, Linux tabanlı özgür ve ücretsiz bir sistemdir. İlk kararlı sürümü Linux tarafından 2004 yılında yayınlanmıştır. Ubuntu'nun kullanımının ücretsiz, tüm detaylarının özelleştirmeye açık ve yazılımının güvenilir oluşu kullanım için tercih sebebidir. Ayrıca işletim sisteminden dolayı oluşan RAM tüketimi minimum seviyede olması sebebiyle, araştırma-geliştirme uygulamaların altyapısı olarak sıkça kullanılmaktadır. Jetson Nano üzerinden alınan işletim sistemi ana sayfası Şekil 3.14'te verildiği gibidir.



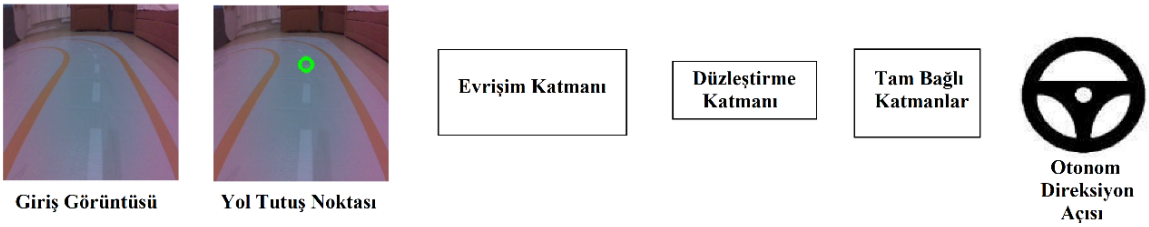
Şekil 3. 14. Jetson Nano Ubuntu işletim sistemi ekran görüntüsü

Jetson Nano'ya iletilen tüm yazılım, algoritma ve komut işlemleri Jupyter Notebook kullanılarak gerçekleştirilmiştir. Jupyter Notebook, birçok farklı programlama dilini bünyesinde barındıran ve kodlamalar için etkileşimli bir ortam sağlayan açık kaynak kodlu bir programdır. İnteraktif bir çalışma ortamı sunmasından ve kullanım pratikliğinden dolayı çalışmada tercih edilmiştir.

Özellikle son yıllarda otonom araçların ve kullanılan algoritmaların gelişmesinde çok büyük ilerlemeler kaydedilmiştir. Yeterli donanımın karşılanmasının yanı sıra otonom sürüşün başarı oranı için önem arz eden bir diğer kriter derin öğrenme algoritmasıdır. İhtiyaç duyulan donanımların kapasitelerinin artması ile beraber otonom sürüşün getirdiği karmaşık problemler gerçek zamanlı çözülebilir hale gelmiş ve bu amaçla birçok yeni kütüphane oluşturulmuştur. Bu bağlamda çalışma kapsamında tercih edilen programlama dili Python olmuştur. Python programlama dili, nesne yönelimli, birimsel, etkileşimli ve yüksek seviyeli bir programlama dilidir. Özellikle son yıllarda modern yazılım geliştirme, veri analiz etme ve makine öğrenmesi konularında öne çıkan bir dil olmuştur. Aynı zamanda Python, birçok yapay zeka kütüphanesini desteklemesi sayesinde deneysel uygulamalarda önemli bir tercih sebebi olmuştur. Arayüzündeki kütüphanelerin birçoğu derin öğrenme ve veri bilimi üzerine elverişlidir. Bu kütüphanelerdeki yüksek kaliteli komutlar, makine öğrenimi kütüphanelerinin ve diğer nümerik algoritma kütüphanelerinin geniş bir çatı altında toplanması ve sürekli gelişmesine büyük katkı sağlamıştır (Bilginç, tarih yok).

Oluşturulan modelde aracın derin öğrenme yöntemi PyTorch kütüphanesi yardımıyla gerçekleşmiştir. PyTorch, Torch kütüphanesine dayanan açık kaynak kodlu bir makine öğrenme kütüphanesidir. Bilgisayarla görü ve doğal dil işleme gibi uygulamaların içerisinde sıkça kullanılmaktadır. PyTorch yazılımının önemli özelliklerinden biri de grafik işlem birimini (GPU) kullanması ve sinir ağı modellerinin pratik bir şekilde oluşturulmasına imkan sağlamasıdır. Aynı zamanda pythonic olan bu kütüphane, Python ortamı tarafından sunulan tüm hizmetlerden ve işlevlerden uyum problemi olmadan yararlanabilmektedir. Kütüphaneden faydalanılarak geliştirilen derin öğrenme algoritması sayesinde araç kamerası üzerinden alınan görüntüler işlenmekte ve araç üzerindeki servo motora tahmin komutlarının iletilmesi ile direksiyon açısı tayin

edilmektedir. Aynı şekilde şeritten dışarıya çıkma veya yoldan uzaklaşma durumunda DC motorun tekerlere verdiği güç hafifletilmekte ve bu sayede aracın kendisini tekrar şeride sokabilmesi kolaylaştırılmaktadır. Çalışma kapsamında uygulanan derin öğrenme eğitim modeli Şekil 3.15'te verildiği gibidir.

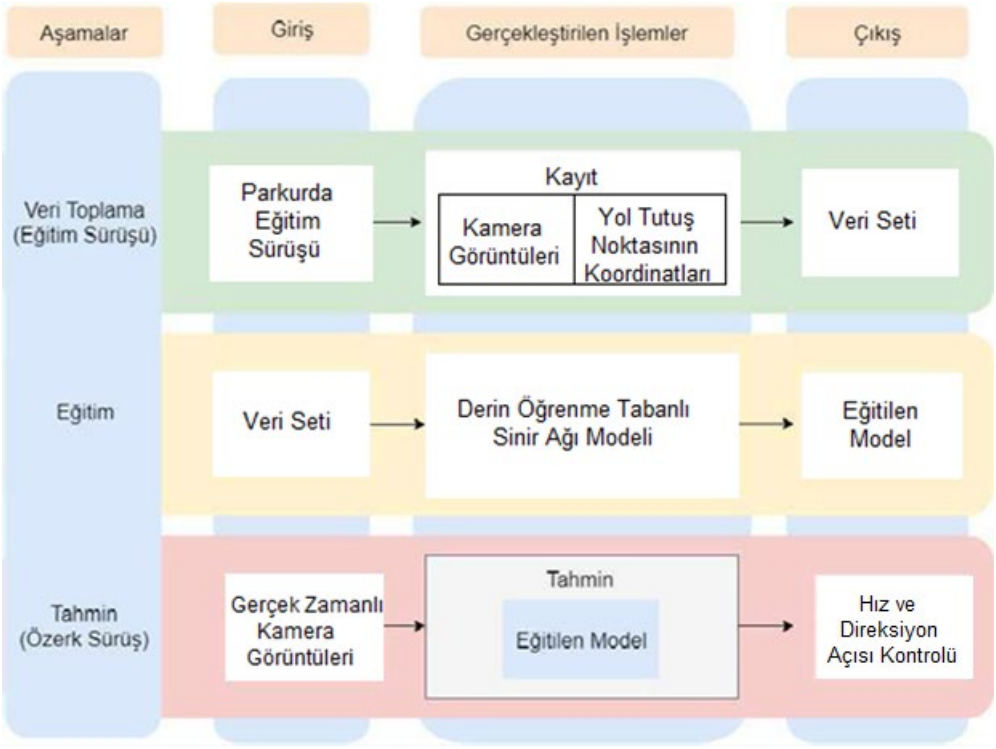


Şekil 3. 15. Derin öğrenme eğitim modeli

Şekil 3.15'te belirtilen yol tutuş noktası, eğitim verisi toplamak için kamera üzerinden alınan manuel bir görüntünün üzerine aracın ilerleyeceği yolun bildirilmesini ifade etmektedir. Oluşturulan otonom aracın özerk hareketi için uygulanan süreç üç ana başlık altında incelenebilir:

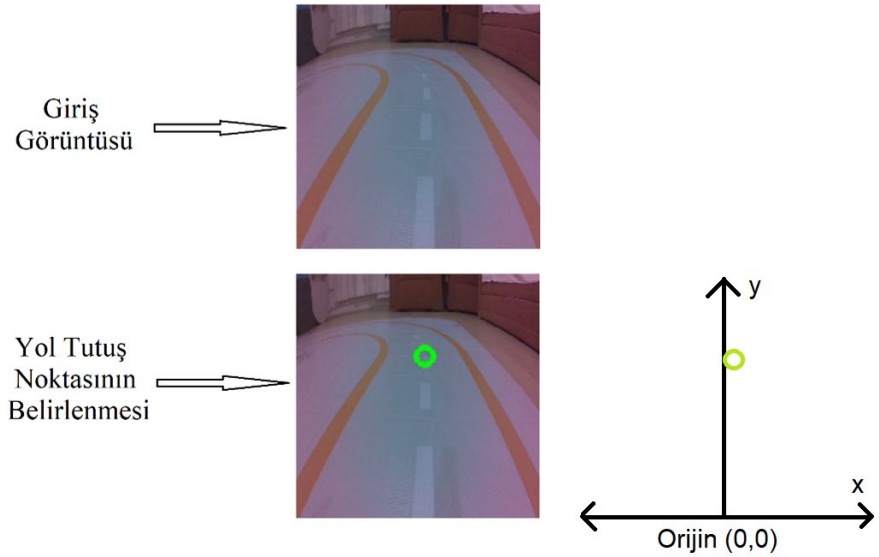
- Kamera üzerinden veri seri oluşturma
- Derin öğrenme modelinin eğitimi
- Otonom aracın test edilmesi

Uygulanacak olan bu süreç, alt tanımlamalarıyla beraber Şekil 3.16'da ifade edilmiştir.



Şekil 3. 16. Derin öğrenme tabanlı otonom sürüş uygulama çerçevesi (Kocic, Jovicic, & Drndarevic, 2019)

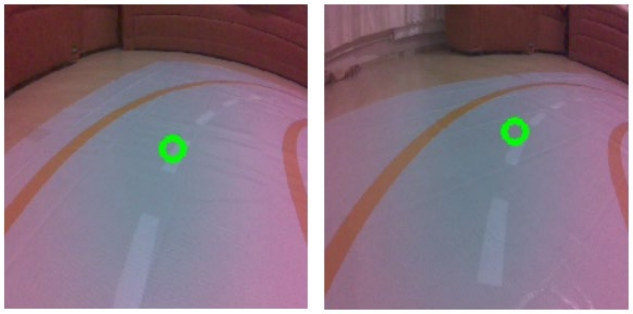
Modelin eğitimi için toplanacak görüntüler, modelin doğru özellikleri öğrenmesini sağlayacak biçimde olmalıdır. Otonom sürüş sırasında aracın nasıl hareket etmesi isteniyorsa veri kümesi de o şekilde oluşturulmalı ve görüntü alım işlemi ona göre yapılmalıdır. Bunun için ilk olarak Jetson Nano üzerine bağlı olan kamera vasıtasıyla görüntü alınacaktır. Bu amaçla aracın elle kullanılarak parkur yolu üzerinde farklı farklı yerlerde konumlandırılması gerekmektedir. Her bir görüntünün alınması esnasında alınan görüntüde aracın takip etmesi gereken nokta işaretlenmekte ve bu sayede eğitim esnasında kullanılacak rota belirlenebilmektedir. Bu işlem Şekil 3.17'de gösterildiği şekilde gerçekleştirilmektedir.



Şekil 3. 17. Yol takibi için çalışma ortamından alınan örnek veri kümesi

Kamera görüntüsü üzerinden yol tutuş noktasının belirlenmesi, otonom sürüşte aracın ilerlerken kullanacağı rota olarak hafızasında kalacaktır. Ana bilgisayar üzerinden gerçekleştirilen bu işlemin temel amacı, aracın karşılaşacağı görüntü karesinde direksiyon ve hız tepkisinin ne olacağı konusunda fikir edinmesini sağlamaktır. Görüntü üzerinde tıklanarak işaretlenen kısmın x eksenindeki konumu direksiyon açısı, y eksenindeki konumu ise ulaşması gereken hız konusunda veri oluşturmaktadır. Aracın çıkabileceği maksimum hız kod üzerinden sınırlandırılmıştır. Bu sayede veri kümesi işaretlenirken istenmeyen yüksek hızların önüne geçilmiştir. Her bir görüntü karesinde bu işlem gerçekleştirilmektedir. Kayıt işlemi için, Jupyter Notebook üzerinden yine Jupyter Notebook'un beraberinde sunduğu ipywidgets eklentisi vasıtasıyla algoritmaya eklenen butonlar üzerinden gerçekleştirilmiştir. Görüntü üzerine müdahale esnasında ise OpenCV kütüphanesinden destek alınmıştır.

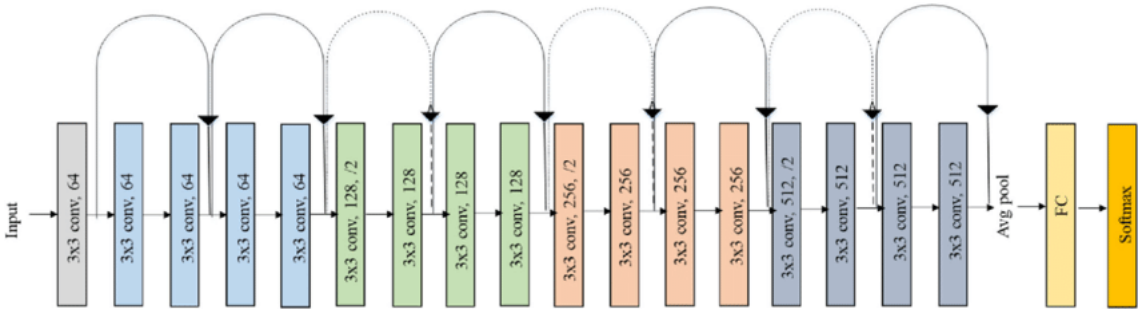
Modelin eğitiminde zor olan kısım virajların düzgün algılanması olmuştur. Örneğin modelin dönüşlerde ve yolun farklı sınırlarında aracı nasıl yönlendireceğini öğrenmesi gerekmektedir. Otonom sürüş için parkurun zorlayıcı kısmı da viraja giriş ve çıkışının gerçekleştiği kısımlardır. Bu kısımlar Şekil 3.18'de aracın kamerası üzerinden gösterilmiştir.



Şekil 3. 18. Viraj esnasındaki kamera görüntüsü ve yol tutuş noktaları

Çalışma kapsamında parkur üzerinden toplamda 1400 görüntü alınmıştır. Kaydedilen her bir görüntü üç kanal derinliğinde (red green blue, RGB) görüntüden oluşmaktadır. Alınan her bir görüntü 224x224 piksel haline dönüştürülerek boyutu küçültülmekte ve eğitim süresinin gereksiz şekilde uzaması engellenmektedir. Kaydedilen her bir görüntünün ortalama bellek boyutu yaklaşık 15 KB'dir. Gereksiz veya yanlış şekilde alınan görüntüler tespit edilmiş ve eğitim setinden çıkartılmıştır. Geriye kalan tüm görüntüler eğitim verisi olarak kullanılmak üzere Jetson Nano kartının hafızasına yüklenmiştir.

Veri setinin oluşturulması sonrası derin öğrenme modelinin eğitilmi gerçekleştirilmitir. Bu eğitimde temel olarak sinir ağının, giriş görüntüleri vasıtasıyla aracın hangi konumda hangi direksiyon açısını kullanması gerektiğini belirlemesi gerekir. Bunun yanında şeritten uzaklaşma durumunda hızı düşürmesi algoritmadan beklenmektedir. Veri seti üzerinden eğitim işlemi başladığında, sinir ağı sürekli çıktı üretir ve sinir ağının kararındaki hatayı hesaplamak için eğitim setindeki kullanıcı tarafından girilen veri değerleriyle araç çıktısı kıyaslanır. Derin öğrenme yönteminde ResNet-18 yapay sinir ağı yapısı kullanılmıştır. Resnet-18, oluşturduğu artık bloklar sayesinde veri kümesindeki giriş verisi doğrudan çıkış verisinde eklenebilmektedir. Bu bloklar, ara katmanların arttırılması ile yoğun derin öğrenme sonucu oluşan optimizasyon sorunlarına ve bozulmaya engel olmakta ve eğitim hatalarını minimize etmektedir. Oluşturulan kısayol blokları üzerinden atlanan bağlantılar sayesinde algoritmanın performansı bir zarara uğramamaktadır. ResNet bünyesinde kendi aralarında tekrar bölünerek ayrışan 18 katman bulundurmaktadır. Derin öğrenme yöntemi olarak araç bünyesinde kullanılan ResNet-18 sinir ağı mimarisi, katman detayları ve artık blok bağlantıları Şekil 3.19'da gösterildiği gibidir.

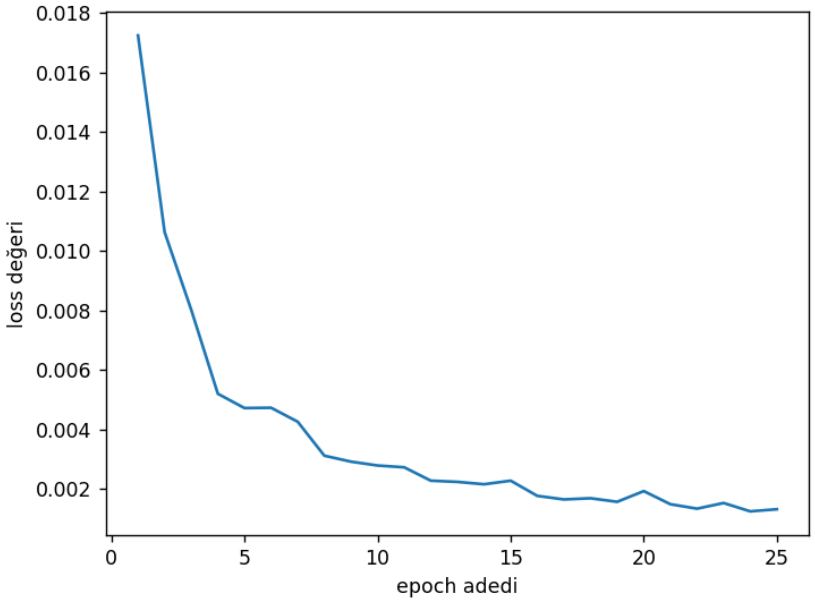


Şekil 3. 19. ResNet-18 yapay sinir ağı mimari diyagramı (Ramzan, Khan, & Iqbal, 2019)

Araçta ayrıca engel algılama ile hız faktörünü kontrol etmek maksadıyla ultrasonik sensör kullanılmıştır. Ultrasonik sensörün kodlanması ile üzerinden alınan mesafe bilgisi input olarak sisteme girilmekte ve aracın sensörünün bulunduğu konuma bir cismin yanaşması durumunda aracın kendisini durdurması istenmektedir. Çalışma sonucunda ultrasonik sensör vasıtasıyla otonom aracın engeller karşısında durması sağlanmıştır. Engel kaldırıldığında ise araç otonom bir şekilde parkur yoluna devam etmiştir.

4. BULGULAR VE TARTIŞMA

Çalışma kapsamında derin öğrenme algoritmasının eğitilmesi için 1400 görüntü kullanılmış ve epoch döngü adedi 25 seçilmiştir. Eğitim süresi yaklaşık iki saat sürmüştür. Eğitim esnasında oluşan kayıp değerlerinin her bir epoch adedinde azaldığı gözlemlenmiştir. Döngü sonunda elfe edilen kayıp değeri 0,00132 olmuştur. Epoch sayısına bağlı olarak kayıp değerlerinin değişim grafiği Şekil 4.1'de verildiği gibidir.



Şekil 4. 1. ResNet-18 mimarisi ile elde edilen Epoch/Loss değişim grafiği

Aynı görüntü kümesi ile epoch 40 ve 50 adede çıkarılarak aynı çalışma gerçekleştirilmiştir. 40 ve 50 epoch üzerinden yapılan eğitim çalışmasının süreleri yaklaşık 4-5 saat aralığında sürmüştür. Her ikisinde de kullanılan görüntü veri kümesi aynı tutulmuştur. Eğitim sonucunda 25 epoch değerli çözüme göre diğer iki sinir ağında 0,0002 gibi az bir farkla kayıp değeri daha düşük çıkmış ancak pratikte aracın otonom hareketinde önemli bir fark oluşmamıştır. Gerçekleştirilen uygulamalarda epoch oranlarına bağlı sürüş farkı oluşumunun fark edilemeyecek kadar benzer olmasının sebebi, yapılan projenin çok büyük bir kapsamı olmayışından da kaynaklanabilir. Kamuyu da kapsayan daha geniş çaplı gerçekleştirilecek olan projelerde epoch sayısı test

edilerek optimum değer belirlenmeli ve kayıp değerinin grafiği dikkate alınarak yaklaşım yapılmalıdır.

Şerit takibine ilaveten engel algılama işlevini de gerçekleştiren araç, 15 cm. den daha yakın bir cismin algılaması halinde kendini durduracak şekilde kodlanmıştır. Önüne konan nesne kaldırıldığında ise tekrar aynı şekilde parkurda otonom sürüşünü sürdürmüştür. Ancak bu süreçte iki farklı donanım olan Jetson Nano ve Arduino Uno kartlarının birbiriyle gecikmesiz haberleşmesi ve senkronize çalışması büyük önem taşımaktadır. Birbirleriyle veri alışverişi esnasında senkron hale getirilseler dahi işlemcilerinin zorlandığı ve komutlarda hafif bir gecikmenin gerçekleştiği gözlemlenmiştir. Bu problem özellikle aracın hızının arttırıldığı durumlarda daha net ortaya çıkmakta ve sinir ağı ile mesafe sensörünün aynı döngü içerisinde çalışması işlemcinin zorlanmasına yol açmaktadır. Bu zorlanma sonucu gecikme meydana gelmekte ve aracın kamera görüşüne ve direksiyon açısına tepkisinde gecikmeler yaşanmaktadır. Böyle bir durumda araç her ne kadar sonrasında kendisini toparlasa dahi yüksek hızlarda zaman zaman parkur dışına çıkabilmektedir.

5. SONUÇ

Yapılan tez çalışmasında yapay zeka, makine öğrenmesi ve derin öğrenme kavramları incelenmiş, yapay sinir ağlarının temel çalışma prensipleri, fonksiyonları ve barındırdıkları değişkenler tanıtılmıştır. Ardından, 0'dan 9'a kadar olan sayılardan meydana gelen MNIST veri seti kullanılarak bir evrişimli sinir ağı modeli oluşturulmuştur. Model oluşturulmasında tüm ağ katmanları detaylıca incelenmiş ve modelin test sonuçları grafikler ile değerlendirilmiştir. Sonuçlar incelendiğinde yeterli iterasyon sonucunda kayıp değerinin 0,0807'ye kadar düştüğü ve sinir ağının %97,51'lik bir doğruluk oranı verdiği tespit edilmiştir. Tez kapsamında gerçekleştirilen ana çalışma ise, bir kamera ve mesafe sensörü üzerinden küçük bir yer aracının otonom hale getirilip getirilemeyeceği ve sürüş konusunda başarılı olup olamayacağı sorusuna cevap bulmaktı. Çalışma esnasında derin öğrenme yöntemiyle otonom hale getirilmiş olan aracın, parkur üzerinde yapılan test sürüşleri sonucunda gerçek zamanlı durma-harekete geçme ve direksiyon dinamiklerinde başarılı olduğu gözlemlenmiş ve bu durum fiziki bir parkur üzerinde somut olarak ortaya konmuştur. Derin öğrenme yöntemi kullanılarak sürücüsüz sürüş yeteneğine sahip olmuş olan araç kendisinden tamamlanması istenen parkuru tamamlamış ve önüne yerleştirilen engelleri fark ederek durmuştur. Çalışmanın sonucunda, derin öğrenme tabanlı bir sinir ağı ile yaklaşık 1400 görüntü üzerinden veri kümesi oluşturulup, eğitim verisi ile hız ve direksiyon açısını fiziksel bir ortamda güvenilir bir şekilde tahmin eden ve parkurda hareket edebilen bir modelin oluşturulabileceği gösterilmiştir. Oluşturulan aracın küçük bir yer aracı olduğu unutulmamalıdır. Küçük olmasından dolayı bünyesinde bulundurabileceği donanımlar sınırlı kalmakta ve kimi zaman bu durum dezavantaj hale gelmektedir. Ancak günümüz taşıtlarında böyle bir sorun söz konusu olmayacaktır. Bu bağlamda düşünüldüğünde otonom araçlar büyük bir potansiyel barındırmakla birlikte tüketicilerin de talebi doğrultusunda gelişim hızı katlanarak artacaktır. Kamu alanlarında kullanılması konusunda önünde kat etmesi gereken yol olmasına rağmen otonom araçlar yakın bir gelecekte çokça karşılaşılacak araçlar haline gelmesi kaçınılmaz hale gelmektedir.

Otonom araç araştırmaları kendi bünyesinde çok geniş araştırma ve çalışma alanına sahiptir. Özellikle üç boyutlu tarama ve algılama sistemlerine ihtiyacın doğması sonucu LIDAR ve üç boyutlu tarayıcı teknolojisi bir adım öne çıkmıştır. Otonom araç çalışmalarında da kullanılan bu teknoloji, görüntünün dışında cihazın etrafının tamamının modellenmesini sağlayarak yapılacak işlemlerde daha doğru kararlar ortaya konulmasını sağlamaktadır. Aynı zamanda aracın sadece kendi sürüşünden sorumlu olmasının dışında, gelişen teknoloji ile görüş açısı kabiliyetinin artması sayesinde diğer sürücülerden kaynaklı gerçekleşen kazaları algılaması ve reaksiyon verebilir bir hale gelmesi mümkün olmaktadır. Diğer taraftan farklı derin öğrenme algoritmaları, pekiştirmeli öğrenme yöntemleri, VVG, AlexNet, SqueezeNet, DenseNet gibi farklı sinir ağı mimarileri de sürekli olarak geliştirilmektedir. Bu açıdan bakıldığında otonom sistemler ve otonom araçlar birçok farklı disiplini kendi bünyesinde bulundurmakla beraber, üzerinde çalışılabilecek sürekli genişleyen bir potansiyel araştırma alanı sağlamaktadırlar.

KAYNAKLAR

Aisha, A., Aliyu, A., & Amina, I. 2021. Development of a Prototype Autonomous Electric Vehicle. *Journal of Robotics and Controls (JRC)*.

Akca, M. F. 2020. Gradient Descent Nedir?. Medium, https://medium.com/deep-learning-turkiye/gradient-descent-nedir-3ec6afcb9900 (Erişim Tarihi: 03.06.2021).

Aki, K. 2019. Derin Öğrenme Tabanlı Sürücüsüz Araç Sistemleri. *Yüksek Lisans Tezi*, Uludağ Üniversitesi Fen Bilimleri Enstitüsü, Bursa.

Aki, K., & Dirik, A. E. 2020. Derin Öğrenme Tabanlı Ve PID Kontrol Tabanlı Sürücüsüz Araç Sistemleri. *Journal of Engineering Sciences and Design*, 306-316.

Alahi, A., Ortiz, R., & Vandergheynst, P. 2012. Freak: Fast retina keypoint. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 510-517. Rhode Island, ABD.

Amidi, S., & Amidi, A. 2018. Handbook of Convolutional Neural Network. Stanford, https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-convolutional-neural-networks#overview (Erişim Tarihi: 16.01.2021).

Anonim, 2016. Sürücüsüz Otomobil Kaliforniya'da. TimeTürk, https://www.timeturk.com/surucusuz-otomobil-kaliforniya-da/haber-1652986 (Erişim Tarihi: 04.01.2021).

Anonim, 2019. Deep Learning Tutorial for Beginners. Kaggle, https://www.kaggle.com/kanncaa1/deep-learning-tutorial-for-beginners (Erişim Tarihi: 02.01.2021).

Anonim, 2020. Convolutional Neural Network (CNN) Tutorial. Kaggle, https://www.kaggle.com/kanncaa1/convolutional-neural-network-cnn-tutorial (Erişim Tarihi: 03.01.2021).

Anonim, 2021. Seeed Studio NVIDIA® Jetson Nano™ Developer Kits. EU Mouser, https://eu.mouser.com/new/seeed-studio/seeed-nvidia-jetson-nano-dev-kit (Erişim Tarihi: 16.08.2020).

Anonim, 2021. Yapay Zeka Nedir? Uygulama Alanları Nelerdir?. MySoft, https://www.mysoft.com.tr/yapay-zeka-nedir (Erişim Tarihi: 28.07.2020).

Arduino, (t.y.). Arduino Uno Rev3. Arduino.CC, https://store-usa.arduino.cc/products/arduino-uno-rev3/ (Erişim Tarihi: 16.08.2020).

Bay, H., Tuytelaars, T., & Gool, V. 2006. Surf: Speeded up robust features. European Conference on Computer Vision (ECCV), 404-417. Graz, Avusturya.

Bayraktar, E., 2013. Design and Control of an Autonomous Blimp. *Yüksek Lisans Tezi* İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, E.E. Mühendisliği, İstanbul.

Berg, A., Deng, J., & Fei-Fei, L. 2010. Large scale visual recognition. *International Journal of Computer Vision*, 211-252.

Bertozzi, M., & Broggi, A. 1998. GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection. IEEE Transactions on Image Processing, 62-81.

Big Data Turkey, 2019. Big Data (Büyük Veri) Nedir?. Medium, https://medium.com/dusunenbeyinler/big-data-b%C3%BCy%C3%BCk-veri-analizi-d53d8f8ab52b (Erişim Tarihi: 14.03.2020)

Bilginç, (t.y.). PYTHON HAKKINDA HER ŞEY. Bilginç, https://bilginc.com/tr/blog/158/python-nedir-python-hakkinda-hersey (Erişim Tarihi: 22.10.2020)

Bingöl, M. S., 2018. Grafik İşleme Ünitesi (GPU) Tabanlı Öğrenme Kullanarak Otonom Araçlar İçin Algılama Sisteminin Geliştirilmesi. *Yüksek Lisans Tezi.* Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ

Bojarski, M., Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., . . . Zieba, K. 2016. End to End Learning for Self-Driving Cars. *ArXiv*.

Bojarski, M., Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., . . . Zieba, K. 2016. Visualbackprop: efficient visualization of cnns. *arXiv*.

Bouchard, Y., 2019. Why Tesla's Fleet Miles Matter for Autonomous Driving. TowardsDataScience, https://towardsdatascience.com/why-teslas-fleet-miles-matter-for-autonomous-driving-8e48503a462f (Erişim Tarihi: 23.11.2020)

Calonder, M., Lepetit, V., Strecha, C., & Fua, P. 2010. Brief: Binary robust independent elementary features. *European Conference on Computer Vision (ECCV)* 778-792., Yunanistan.

Cardinal, D. 2011. Changing the world: DARPA's top inventions. ExtremeTech, https://www.extremetech.com/extreme/105117-inventing-our-world-darpas-top-inventions/4 (Erişim Tarihi: 22.11.2020)

Chen, Y., Wang, J., Li, J., & Lu, C. 2018. LiDAR-Video Driving Dataset: Learning Driving Policies Effectively. *Computer Vision and Pattern Recognition (CVPR) IEEE Conference*, pp. 5870-5878.

Cho, M.-g. 2019. A Study on the Obstacle Recognition for Autonomous Driving RC Car Using LiDAR and Thermal Infrared Camera. *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN).* Zagreb, Hırvatistan.

Dalal, N., & Triggs, B. 2005. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition (CVPR) IEEE Conference*, pp. 886–893. San Diego, Kalifornia

Datta, A. 2021. Waymo drops 'self-driving' from its name, opts for 'autonomous driving' instead. Laptrinhx, https://laptrinhx.com/waymo-drops-self-driving-from-its-name-opts-for-autonomous-driving-instead-4161846715/ (Erişim Tarihi: 24.11.2020)

Doğan, Ö. 2020. Derin Öğrenme Nedir? Yapay Sinir Ağları Ne İşe Yarar?. Teknoloji, https://teknoloji.org/derin-ogrenme-nedir-yapay-sinir-aglari-ne-ise-yarar/ (Erişim Tarihi: 24.11.2020)

Doruk, A. 2016. Şerit Takibi ve Hız Ayarı Yapabilen Otonom Robot Araba. *Uluslararası Blgisayar Bilimleri ve Mühendisliği Konferansı.* Namık Kemal Üniversitesi, Çorlu Mühendislik Fakültesi. Tekirdağ, Türkiye

Dossman, C. 2018. Deep Learning Performance Cheat Sheet. TowardsDataScience, https://towardsdatascience.com/deep-learning-performance-cheat-sheet-21374b9c4f45 (Erişim Tarihi: 26.11.2020)

Duda, R. O., Hart, P. E., & Stork, D. G. 2000. Pattern Classification (Second Edition). John Wiley & Sons Inc. New Jersey, ABD

Güner, Ş. 2012. Otonom Bir Otomobil İçin Hız Kontrolörü Tasarımı Ve Uygulaması. *Yüksek Lisans Tezi.* İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü. İstanbul

Güzel, Y. 2018. Yapay Zeka Ders Notları 03 | Biyolojik Sinir Sistemi ve Yapay Sinir Ağı Hücresi. Medium, https://medium.com/@yasinguzel/yapay-zeka-ders-notlar%C4%B1-03-biyolojik-sinir-sistemi-ve-yapay-sinir-a%C4%9F%C4%B1-h%C3%BCcresi-6555add68d80 (Erişim Tarihi: 16.11.2020)

Haverford. (t.y.). Haverford. Haverford, http://cs.haverford.edu (Erişim Tarihi: 12.12.2020)

Intel. (t.y.). Intel® Dual Band Wireless-AC 8265. Intel, https://www.intel.com.tr/content/www/tr/tr/products/sku/94150/intel-dual-band-wirelessac-8265/specifications.html (Erişim Tarihi: 12.11.2020)

Kaloha. 2019. Jetson Nano Series Tutorial 5: Life and death look down, do UART without accepting it. Waveshare, https://www.waveshare.net/study/article-888-1.html (Erişim Tarihi: 13.09.2020)

Kapellmann-Zafra, G. 2013. Design and Development of a FROG Instrument for the Characterization of Femto-second Laser Pulses. *Yüksek Lisans Tezi.* National Autonomous University of Mexico (UNAM), Mexico.

Kayaduman, A., Arıkan, A., Polat, S., Şimşek, Y., Dikmen, İ., Bakır, H., . . . Abbasov, T. 2018. Control method simulation and application for autonomous vehicles. *2018 International Conference on Artificial Intelligence and Data Processing (IDAP).* Malatya, Türkiye

Ke, Y., & Sukthankar, R. 2004. Pca-sift: A more distinctive representation for local image descriptors. *Computer Vision and Pattern Recognition (CVPR) IEEE Conference*, pp. 398–400. Washington, ABD.

Kızrak, A. 2018. ŞU KARA KUTUYU AÇALIM: Yapay Sinir Ağları. Medium, https://ayyucekizrak.medium.com/%C5%9Fu-kara-kutuyu-a%C3%A7alim-yapay-sinir-a%C4%9Flar%C4%B1-7b65c6a5264a (Erişim Tarihi: 09.08.2020).

Kocic, J., Jovicic, N., & Drndarevic, V. 2019. An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms. *Sensors Journal*.

Kurakin, A., Goodfellow, I., & and Bengio, S. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv*.

Leutenegger, S., Chli, M., & Siegwart, R. 2011. Binary robust invariant scalable keypoints. *IEEE International Conference on Computer Vision (ICCV)* pp. 2548–2555. Barselona, Spain.

Manyika, J., Chui, M., Bughin, J., Dobbs, R., Bisson, P., & Marrs, A. 2013. Disruptive technologies: Advances that will transform life, business, and the global economy. McKinsey Global Institute. Washington D. C.

Maurer, M., Behringer, R., Dickmanns, D., T. Hildebrandt, Thomanek, F., Schiehlen, J., & Dickmanns, E. 1995. VaMoRs-P: an advanced platform for visual autonomous road vehicle guidance. *Mobile Robots IX.* Boston.

Nvidia. 2019. Jetson Nano Brings AI Computing to Everyone. Developer-Nvidia, https://developer.nvidia.com/blog/jetson-nano-ai-computing (Erişim Tarihi: 10.08.2020).

Nvidia. (t.y.). NVIDIA Jetson Modülünün Teknik Özelliklerini Karşılaştırın. Nvidia, https://www.nvidia.com/tr-tr/autonomous-machines/embedded-systems/ (Erişim Tarihi: 12.08.2020).

Oh, S.-I., & Kang, H.-B. 2016. Fast Occupancy Grid Filtering Using Grid Cell Clusters From LIDAR and Stereo Vision Sensor Data. *IEEE Sensors Journal* , 16/19.

OICA. 2019. Organisation Internationale des Constructeurs d'Automobiles. OICA, https://www.oica.net/ (Erişim Tarihi: 15.09.2020).

Özgüner, Ü., Acarman, T., & Redmill, K. 2011. Autonomous Ground Vehicles. Artech House. ABD, 24-26.

Pomerleau, D. 1989. ALVINN: An Autonomous Land Vehicle In a Neural Network. *Advances in Neural Information Processing Systems*.

Ramzan, F., Khan, M. U., & Iqbal, S. 2019. A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks. *Journal of Medical Systems*.

RoboBlok. (t.y.). HC-SR04 Ultrasonik Mesafe Sensörü. RoboBloq, https://www.robobloq.com.tr/hc-sr04-ultrasonik-mesafe-sensoru-24 (Erişim Tarihi: 22.11.2020).

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage andorganization in the brain. *APA PsycArticles.* 386-408.

Rubik's Code. 2018. Introduction to Convolutional Neural Networks. Rubik'sCode, https://rubikscode.net/2018/02/26/introduction-to-convolutional-neural-networks/ (Erişim Tarihi: 18.04.2020).

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. 2011. Orb: an efficient alternative to sift or surf. *International Conference on Computer Vision (ICCV).* pp. 2564–2571. Barcelona, Spain.

SeedStudio. (t.y.). IMX219-160 8MP Camera with 160° FOV - Compatible with NVIDIA Jetson Nano/ Xavier NX. Seeed Studio, https://www.seeedstudio.com/IMX219-160-Camera-160-FOV-Applicable-for-Jetson-Nano-p-4603.html (Erişim Tarihi: 16.08.2020).

Şanlı, E. 2018. Yapay Sinir Ağı Kontrollü Otonom RC Araç Uygulaması. *Yüksek Lisans Tezi.* İstanbul Gelişim Üniversitesi Fen Bilimleri Enstitüsü., İstanbul.

TheVerge. 2020. Waymo Pulls Back The Curtain On 6.1 Mıllion Miles Of Self-Driving Car Data In Phoenix. Verge, https://www.theverge.com/2020/10/30/21538999/waymo-self-driving-car-data-miles-crashes-phoenix-google (Erişim Tarihi: 03.12.2020).

TUİK. 2020. Karayolu Trafik Kaza İstatistikleri, 2020. TUİK, https://data.tuik.gov.tr/Bulten/Index?p=Road-Traffic-Accident-Statistics-2020-37436 (Erişim Tarihi: 11.09.2020).

TÜİK. 2019. Türkiye İstatistik Kurumu Veri Havuzu. DATA-TUİK, https://data.tuik.gov.tr/ (Erişim Tarihi: 08.07.2020).

Uçar, A., Bingöl, M. S., & Kaymak, Ç. 2019. Derin Öğrenme Kullanarak Otonom Araçların İnsan Sürüşünden Öğrenmesi. *Fırat Üniversitesi Mühendislik Bilimleri Dergisi*.

Uçar, A., Demir, Y., & Güzeliş, C. 2014. A penalty function method for designing efficient robust classifiers with input–space optimal separating surfaces. *Turkish Journal of Electrical Engineering and Computer Science*, 1664–1685.

Uçar, A., Demir, Y., & Güzeliş, C. 2016. A new facial expression recognition based on curvelet transform and online sequential extreme learning machine initialized with spherical clustering. Neural Computing and Applications, 131–142.

Webb, G. I., Boughton, J., & Wang, Z. 2005. Not So Naive Bayes: Aggregating One-Dependence Estimators. Machine Learning (Springer), 5-24.

Wikipedia. 2021. Arduino Uno. Wikipedia, https://tr.wikipedia.org/wiki/Arduino_uno (Erişim Tarihi: 04.04.2020).

Wikipedia. 2021. DARPA Grand Challenge (2007). Wikipedia, https://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2007) (Erişim Tarihi: 16.08.2020).

William, B., & Edvin, V. 2016. Artificial Neural Network Autonomous Vehicle. Kth Royal Institute of Technology. Stockholm.

Yandex. 2017. Yandex Autonous Car. SDC.Yandex, https://sdc.yandex.com/ (Erişim Tarihi: 25.05.2020).

Ye, M., Xu, S., & Cao, T. 2020. HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection. *Computer Vision and Pattern Recognition (CVPR) ArXiv.* pp. 1631-1640.

Yumruktay, M. 2015. GNSS Destekli Prototip Otonom Araç Tasarımı. *Yüksek Lisans Tezi*. Marmara Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

ÖZGEÇMİŞ

Adı Soyadı : Mahmut Esat SEÇKİN
Doğum Yeri ve Tarihi : Bursa / 30.08.1995
Yabancı Dil : İngilizce

Eğitim Durumu
- Lise : Bursa İpekçilik Anadolu İmam Hatip Lisesi
- Lisans : Bilecik Şeyh Edebali Üniversitesi / Makine Mühendisliği
- Yüksek Lisans : Bursa Uludağ Üniversitesi / Otomotiv Mühendisliği

İletişim (e-posta) : esadseckin@gmail.com