

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME YAKLAŞIMI KULLANILARAK OTONOM
ARAÇLAR İÇİN PARK YAPMA

EBUBEKİR SİDDİK CEYLAN
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2021

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME YAKLAŞIMI
KULLANILARAK OTONOM ARAÇLAR
İÇİN PARK YAPMA

EBUBEKİR SİDDİK CEYLAN
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
DOÇ. DR. FUAD ALIEW

GEBZE
2021

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

PARKING FOR AUTONOMOUS VEHICLES
USING DEEP LEARNING APPROACH

EBUBEKİR SİDDİK CEYLAN
A THESIS SUBMITTED FOR THE DEGREE OF
MASTER OF SCIENCE
DEPARTMENT OF ELECTRONIC ENGINEERING

THESIS SUPERVISOR
ASSOC. PROF. DR. FUAD ALIEW

GEBZE
2021

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 22/01/2021 tarih ve 2021/04 sayılı kararıyla oluşturulan jüri tarafından 28/01/2021 tarihinde tez savunma sınavı yapılan Ebubekir Sıddık CEYLAN'ın tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) : Doç. Dr. Fuad ALIEW

ÜYE

: Dr. Öğr. Üyesi Atilla UYGUR

ÜYE

: Dr. Öğr. Üyesi Bülent İRFANOĞLI

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

...../...../..... tarih ve/..... sayılı kararı.

ÖZET

Günümüzde otonom araçlar hayatımızın birçok alanında kullanılmaya başlamıştır. Bunun sebeplerinden biri de otonom araçların insan hayatını oldukça kolaylaştırmasıdır. Ancak bu faydalar yanında zorluklar da getirmektedir. Bu zorlukların olduğu problemlerden biri de aracın kendi kendine park etme sorunudur. Bu alanda otonom işlemler artıkça hesap yükü, maliyet, karmaşıklık artmaktadır. Bu sorunu çözmek için yapılmış çalışmalar genel olarak klasik SLAM algoritmaları kullanılarak ya da makine öğrenmesi ile sınıflandırma yapılarak aracın kendi kendine park yapmasını sağlamak amacıyla bazı çalışmalar yapılmıştır. Bu tezde ise, bu probleme daha yenilikçi bir yaklaşımla bakılarak derin öğrenme yaklaşımı ile çözüm getirilmek istenmiştir. Birinci aşamada veri toplamak amacıyla telefon ile uzaktan kontrol edilebilen otonom araç tasarımı yapılmıştır. İkinci aşamada veri toplamak ve aracın testini yapmak amacıyla parkur tasarlanmıştır. Uzman kullanıcı tarafından kontrol edilen araç ile bu parkur üzerinde park yerinin müsait olduğu ve olmadığı durumdaki veriler toplanılmış, bu ham veriden çıkarımlar yapılmıştır. Bu verilere ek olarak CNRPark veri seti de kullanılarak modelin eğitilmesi için çeşitli bir veri tabanı oluşturulmuştur. Bu veri tabanından rastgele şekilde seçilerek %75'i eğitim amaçlı, %25'i ile de test amaçlı kullanılarak derin öğrenme modeli geliştirilmiştir. Eğitilen bu modeli de araca gömerek, aracın müsait park yeri tespit edebilmesindeki başarısı ve başarılı şekilde park manevrası yapıp yapmadığı incelenmiştir. Bu çalışmada, bir aracın otonom park problemine yönelik olarak kendi kendine park yeri sınırlarının içerisinde kalacak şekilde park yapabilmesi için bir çalışma sunulmuştur.

Anahtar Kelimeler: Otonom Araçlar, Derin Öğrenme, Yapay Zekâ, Evrişimli Sinir Ağları, Park Yapma.

SUMMARY

Today autonomous vehicles have been started to be used in many area of our lives. One of the reasons for is that autonomous vehicles make human life much easier. However, these benefits also bring difficulties. One of these difficulties is the parking problem of vehicle itself. As autonomous operations increase calculated load, cost and complexity increase in this aera. Studies to solve this problem are presented with solutions to self-parking by using classically SLAM algorithms or classification by machine learning. In this thesis, it is aimed to look at this problem with a more innovative way and to find a solution with deep learning approach. In the first stage, an autonomous vehicle that can be controlled remotely by phone was designed to collect data. In the second stage, the track was designed to collect data and test the vehicle. With the vehicle controlled by the expert user, the data on availability and absences of parking space on this track we collected, and inferences were made from this raw data. In addition to these data, a diverse database was created to train the model by using the CNR parking dataset. A deep learning model has been developed by using 75% of these data for training and %25 of them selected by randomly for testing. By embedding this trained model in the vehicle, the success of the vehicle in detecting a suitable parking space and whether it successfully maneuvered parking was analyzed.

In this study, a solution is proposed for autonomous parking problem to enable a vehicle to park safely itself.

Key Words: Autonomous Vehicles, Deep Learning, Artificial Intelligence, Convolutional Neural Network, Parking.

TEŞEKKÜR

Başta, yüksek lisans eğitimimde ve akademik hayatımda desteğini ve yardımlarını hiçbir zaman esirgemeyip bu çalışmanın oluşmasına olanak sağlayan danışmanım Doç. Dr. Fuad ALIEW'e, yüksek lisans eğitimimi almış olduğum Gebze Teknik Üniversitesindeki üzerimde emeği bulunan tüm hocalarıma teşekkürlerimi sunmayı bir borç bilirim.

Son olarak bütün çalışmam boyunca yanımda olan, stresli zamanlarımda benden desteklerini esirgemeyen anneme, babama ve neşe kaynağım yeğenim İkranur Aksoy'a en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

	<u>Sayfa</u>
ÖZET	v
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER	viii
SİMGELER ve KISALTMALAR DİZİNİ	x
ŞEKİLLER ve RESİMLER DİZİNİ	xi
TABLOLAR DİZİNİ	xiii
1. GİRİŞ	1
2. OTONOM ARAÇ TASARIMI	4
2.1. Kullanılan Mekanik-Elektronik Modüller	4
2.1.1. Aracın Gövdesi	4
2.1.2. Mikrobilgisayar	5
2.1.3. Motorlar	7
2.1.4. Motor Sürücüsü	7
2.1.5. Arduino Uno	8
2.1.6. Haberleşme Modülü	10
2.1.7. Kamera Modülü	11
2.1.8. Mesafe Sensörü	12
2.1.9. Güç Kaynağı	12
2.2. Bütün Komponentlerin Birleştirilmesi	13
2.3. Park Alanı Kurulması	18
3. EVRİŞİMLİ SİNİR AĞLARI	20
3.1. Genel bilgiler	20
3.2. Evrişimli Sinir Ağ Mimarisi	21
3.2.1. Giriş Katmanı	21
3.2.2. Evrişim Katmanı	21
3.2.3. Ortaklama Katmanı	22
3.2.4. Tam Bağlantı Katmanı	23
3.2.5. Çıkış Katmanı	23

3.2.6. Aktivasyon Fonksiyonları	23
3.2.6.1. Sigmoid Fonksiyonu	24
3.2.6.2. Hiperbolik Tanjant Fonksiyonu	24
3.2.6.3. ReLU Fonksiyonu	25
3.2.6.4. Sızıntı ReLU Fonksiyonu	26
3.2.6.5. Swiss Fonksiyonu	26
3.2.7. Seyreltme Katmanı	27
3.2.8. Paket Normalizasyonu	27
3.2.9 Maliyet ve Optimizasyon Fonksiyonları	27
3.3. Nesne Tanıma ve Sınıflandırma	28
4. MODEL EĞİTİMİ VE YAZILIM GELİŞTİRİLMESİ	29
4.1. RPI Kurulumu	29
4.2. Python ve Kütüphane Seçimi	31
4.2.1. Yazılım Dili Seçimi	32
4.2.2. Derin Öğrenme Kütüphaneleri	32
4.3. Veri Seti Oluşturulması	33
4.3.1 Veri Seti Toplanması	33
4.3.2 Veri Seti Etiketlenmesi	35
4.4. Evrişimli Sinir Ağı Model Eğitimi	36
4.4.1. Google Colab Üzerinde Model Eğitimi	36
4.4.2. Windows CPU Üzerinde Model Eğitimi	38
4.5. Önerilen Sinir Ağı Modelinin Mimarisi ve Performansı	40
4.5.1. Önerilen Modelin Mimarisi	41
4.5.2. Önerilen Modelin CNR Park Veri Seti Üzerindeki Performansı	44
4.5.3. Önerilen Modelin Bütün Veri Seti Üzerindeki Performansı	45
4.6. Acinet Modeli ile Diğer Modellerin Karşılaştırılması	46
4.7. Öğrenme Aktarımı	51
4.8. Park Yapma İşlemi	52
5. SONUÇLAR VE GELECEK ÇALIŞMALARI	55
KAYNAKLAR	56
ÖZGEÇMİŞ	59
EKLER	60

KISALTMALAR DİZİNİ

Kısaltmalar

ESA

Acinet

RPİ

RC

FC

ReLU

CPU

SGD

VGG-11

VGG-13

VGG-16

SLAM

CNRPark

Açıklamalar

Evrişimsel sinir ağları

Önerilen sinir ağı modeli

Raspberry Pi

Uzaktan kontrol

Tam bağlantı

Doğrultulmuş doğrusal birimler

Merkezi işlemci birimi

Stokastik gradyan inişi

11 katmanlı sinir ağı modeli

13 katmanlı sinir ağı modeli

16 katmanlı sinir ağı modeli

Eş zamanlı konum belirleme ve haritalama

Park yeri görsel boşluk tespiti için toplanmış hazır veri seti

ŞEKİLLER DİZİNİ

<u>Şekil No:</u>	<u>Sayfa</u>
2.1: Rock crawler elektrikli aracı.	4
2.2: Piyasaya sürülen bütün rpi modelleri.	5
2.3: RPİ bağlantı diyagramları.	6
2.4: RPİ L298N dc motor sürücü kartı.	8
2.5: Arduino uno bağlantı şeması.	9
2.6: HC-05 bluetooth modülü.	10
2.7: RPi kamera V1.3 modülü.	12
2.8: HC-SR04 ultrasonik sensör.	12
2.9: Dexim 5000 mAh powerbank.	13
2.10: Arduino uno HC-05 bağlantısı.	14
2.11: RC otonom araç önden görünüşü.	15
2.12: RC otonom araç soldan görünüşü.	16
2.13: RC otonom araç sağdan görünüşü.	16
2.14: RC araç sisteminin genel bağlantı şeması.	17
2.15: Dik park alanı yukarıdan görünüşü.	18
2.16: Çapraz park alanı yukarıdan görünüşü.	19
3.1: Evrişimli sinir ağları genel mimari gösterimi.	20
3.2: 5x5 girdi ve 3x3 filtre ile öznitelik çıkarımı.	22
3.3: 4x4 girdi ile maksimum ve ortalama havuzlama.	22
3.4: Sigmoid fonksiyonu ve türevi.	24
3.5: Hiperbolik tanjant fonksiyonu ve türevi.	24
3.6: ReLU fonksiyonu ve türevi.	25
3.7: Sızıntı ReLU fonksiyonu ve türevi.	245
3.8: Swiss fonksiyonu ve türevi.	256
4.1: Raspberry pi os ve çeşitleri.	29
4.2: Raspberry pi konfigürasyon ara yüzü.	30
4.3: RPİ ait IP adresi.	30
4.4: VNC ait arayüz ekranı.	31
4.5: Telefon ile veri toplamak için gerekli kullanıcı arayüzü.	34
4.6: Uzman kullanıcı tarafından toplanan görüntülerin bir kısmı.	35

4.7: Etiketlenmiş bir örnek görüntü.	36
4.8: Colab çalışma zamanı değiştirme işlemi.	37
4.9: Colabta gpu üzerinden eğitilmiş bir modelin görüntüsü.	38
4.10: Csv uzantılı veri seti oluşturulması.	39
4.11: Windowsta cpu üzerinden eğitilmiş bir modelin görüntüsü.	40
4.12: Önerilen ağ mimarisi.	41
4.13: Önerilen modelin tüm yapısı.	42
4.14: Acinet'in CNR veriseti üzerindeki başarısı.	42
4.15: Acinet'in bütün veriseti üzerindeki başarısı.	42
4.16: VGG-11 modelinin bütün veri seti üzerindeki performansı.	46
4.17: AlexNet modelinin bütün veri seti üzerindeki performansı.	47
4.18: VGG-13 modelinin bütün veri seti üzerindeki performansı.	48
4.19: VGG-16 modelinin bütün veri seti üzerindeki performansı.	49
4.20: Öğrenme aktarımı ile anlık park yeri tespiti.	52
4.21: Çapraz park manevrası.	53
4.22: Dik park manevrası.	54
B1.1: 1. Test nesne sınıflandırma senaryosu.	60
B1.2: 2. Test nesne sınıflandırma senaryosu.	60
B1.3: 3. Test nesne sınıflandırma senaryosu.	61
B1.4: 4. Test nesne sınıflandırma senaryosu.	61
B1.5: 5. Test nesne sınıflandırma senaryosu.	61
B1.6: 6. Test nesne sınıflandırma senaryosu.	62
B1.7: 7. Test nesne sınıflandırma senaryosu.	62
B1.8: 8. Test nesne sınıflandırma senaryosu.	62

TABLÖLAR DİZİNİ

<u>Tablo No:</u>	<u>Sayfa</u>
2.1: Arduino uno özellikleri.	9
2.2: HC-05 arduino bluetooth modül.	10
2.3: Raspberry pi kamera V1.3 teknik özellikler.	11
4.1: Yazılım ve donanım araç bilgisi.	33
4.2: Kullanılan ana kütüphaneler.	33
4.3: AciNet CNR üzerindeki toplam performans sonuçları.	44
4.4: AciNet bütün verilerinde üzerindeki toplam performans sonuçları.	45
4.5: VGG-11 bütün veriler üzerindeki performans sonuçları.	47
4.6: AlexNet bütün veriler üzerindeki performans sonuçları.	48
4.7: VGG-13 bütün veriler üzerindeki performans sonuçları.	49
4.8: VGG-16 bütün veriler üzerindeki performans sonuçları.	50
4.9: Bütün mimarilerin karşılaştırılması.	50

1. GİRİŞ

Geçmişten günümüze insan hayatını kolaylaştıran birçok robot ortaya çıkmıştır. Bunlardan bazıları insandan daha hızlı çalışan mutfak robotları, insandan daha verimli çalışan endüstriyel robotlar, kargo işlerini daha hızlı yapan drone'lar, daha derin noktalara inen sualtı araçları, terörist kovalayan İha'lar gibi birçok örnek verebiliriz. Bunlar gibi örnek verebileceğimiz bir teknolojik gelişim ise özellikle yapay zekâ teknolojilerin artması, verinin daha çok işlenebilir olması, bununla birlikte daha hızlı işlemcilerin piyasaya çıkmasıyla hız kazanmış olan otonom araçlardır. Son zamanlarda hayatımıza giren bu teknolojilerin güvenilir olması için, insanın gösterdiği performansa eşdeğer ya da daha iyi bir performans göstermesi beklenmektedir. Otonom araçların çözüm sunabileceği konulardan biri de insanların park ederken ya da park yaptığı yerden çıkarken yaptığı park kazalarıdır. Bu kazalar çoğunlukla sürücünün tecrübesiz olmasından ya da dikkatsiz olmasından kaynaklanmaktadır. Bu hem maddi olarak zarar vermekte hem de şoförün kendine olan güvenini zedelemektedir. Bu problem otonom araçların insandan daha iyi performans göstereceği problemlerden biridir. Araç hasarı maliyetleri düşürülebilir, yakıt tasarrufu yapılabilir, araç park etme süresi kısaltılarak zamandan kazanç sağlanabilir. Bunu yaparken iki aşama göze alınmalıdır. İlk aşamada otonom aracı tasarlamak ve ikinci aşama da ise seçilecek yöntemlerdir.

Otonom araçlar konusunda başlangıç çalışmalarına 1939 senesinde Dünya fuarını düzenleyen General Motors firması öncülük etmiştir [1]. Ancak ilgi çekici çalışmalar 1977 senesinde Japonya'nın Tsukuba bölgesinde yer alan Makine Mühendisliği Laboratuvarında başladığı söylenebilir. Bu laboratuvarda 20km/s hıza ulaşabilen ve beyaz çizgiyi takip eden bir araç geliştirilmiştir [2]. 1995 yılında yaklaşık 1 milyar dolar harcanarak o döneme kadarki en uzun robot araba projesi olmuş olan "European Prometheus Projesi" inde boş sokaklarda araç 96km/s' e kadar çıkarak başarılı sürüş gerçekleştirmiştir [2]. Dickmns'ın meşhur S-sınıfı arabası otonom olarak Münih'ten Danimarka'ya insan dokunuşu olmadan saatte 158 km yol yaparak toplamda 1678 km yol yapmıştır [2]. 1989 tarihinde Pomerleau, Amerika da DARPA' tarafından finans edilen bir kara aracı olan ALVINN'e ilk yol sürüşünü yaptırdı. Bu aracın önceki çalışmalardan farkı sinir ağlarından faydalanılarak ve araca koyulmuş

olan kamera, lazer gibi sensörlerden gelen verileri işleyerek, aracı 30 km/s hıza ulaştırılmıştır [3].

Günümüze yaklaştıkça, yakın tarihte otonom araçların tasarımları, teknolojinin gelişmesi ile birlikte maliyeti düşmesi ve yeni seçeneklerin sunulması ile çeşitlenmiştir. Bunlara örnek olarak 2008 yılında Çayiroğlu ve Şimşir, PIC16F877A kullanarak tasarladıkları robotu ARX-34S alıcı ve ATX-34S verici kullanarak, wireless kameradan gelen kayıtları bilgisayara transfer etmişlerdir [4]. Fong ve Yusoff gerçek zamanda aracı kontrol etmek için Wi-Fi kullanarak aracı görüntülemek için çalıştılar [5]. XBee modülü ile bilgisayar üzerinden mobil araç kontrol edilmiştir ve görüntü transferi wireless kamera ile yapılır. Kuo, bu robotu Arduino ile yapmış bu robota wireless kontrollü temizleme aracı özelliği katmıştır [6]. Espes ve arkadaşları 2014 yılında internet üzerinden çalışan bir mobil robot tasarladı. Bu robotta internete bağlanmak için Rasperry Pi, robot ile direksiyon kontrolü ve robot kolunun hareketleri içinde Arduino mega kullanılarak evde görüntülemek için web kontrollü uygun masraflı bir mobil araç geliştirdiler [7]. Yine Ali de 2017 yılında bulanık mantık ile konum kontrolü yaptığında otonom araç tasarlamış, bu araçta Arduino Mega kullanarak bluetooth ile aracı uzaktan kontrol etmeye çalışmıştır [8].

Aracın tasarımı ve kullanılan teknolojiler kadar kullanılacak yöntemlerde önemlidir. Geçmişte daha klasik yöntemler kullanılsa da günümüzde geleneksel yöntemler yerine derin öğrenme algoritmalarına bırakmaya başlamıştır. 2019 yılında Klasik yöntemlerden birine örnek vermek gerekirse Alpkıray dönüş rotasını oluşturmak için PRM ve ABC algoritmasını kullanmıştır [10]. Yine aynı yılda Bingöl ve arkadaşları Derin öğrenme metoduna örnek olarak da daha kararlı bir sürüş için NVIDIA Jetson TX2 kartını kullanarak tasarladıkları araçta, önce aracı kendileri sürerek veri toplayıp bu veriden öğrenen derin öğrenme modeli geliştirerek aracın hedefine ulaşmasını sağlamıştır [9].

Otonom araçların uzman sürüşü olmadan kendi başlarına aşmaları gereken birçok problem vardır. Bunlardan biri de park yapma problemidir. Bu probleme çözüm bulmuş olan Notomista ve Botsch, probleme makine öğrenmesi yaklaşımı ile yaklaşmışlardır. Tasarladıkları araçta manevraları 3 ayrı parçaya bölmüş, bu parçaları iki farklı sınıflandırıcıdan geçirerek tasarladıkları genel radyal temel fonksiyon model ile birlikte aracın kendi kendine park etmesi konusunda başarılı olmuştur [11].

Zhang ve arkadaşları park problemine başka bir çözüm sunmuşlardır. Burada yaptıkları Pekiştirmeli öğrenme ile araca ceza ve ödül yoluyla her denemesinden öğrenmesi sağlanmıştır. Bu öğrenmeyle birlikte araca sadece giriş görüntüsü verilerek, uçtan uca öğrenme yöntemi yardımıyla kendiliğinden park etme işlemini gerçekleştirmiştir [12]. Aynı şekilde Jelena ve arkadaşları da yine uçtan uca öğrenme yöntemiyle ancak bu sefer Derin Sinir Ağları kullanarak gömülü otomotiv platformu için araç tasarlamış ve bu araca başarılı bir şekilde park işlemini yaptırmıştır [13]. Moon ve arkadaşları tasarladıkları yapay sinir ağı modelini klonlayarak ve bu klonu kendisiyle birleştirerek çift yapay sinir ağı modeli oluşturdular. Bu model ile başarılı park yapan otomatik park kontrolcüsü geliştirdiler. [14]

Bu tez kapsamında, öncelikle bir mobil araç tasarlanmıştır. Bu araç uzaktan uzman bir personel tarafından kontrol edilerek araca belli bir bölge de birçok kez park etme işlemi yaptırılmıştır. Bu park işlemleri sırasında kamera ile bu işlemler kayıt altına alınmıştır. Elde tutulan bu görüntü verilerinden özellik çıkartma işlemi yapılmıştır. Bu veriler CNR park veri seti ile birleştirerek, modelin eğitimi için gerekli veriler toplanmıştır. Bu özellikli veriler kullanılarak, evrişimli sinir ağı tabanlı derin öğrenme modeli eğitilmiştir. Bu geliştirilmiş model ile araç için daha önceden kurulmuş olan parkurda park yeri uygunluk durumu sınıflandırma işlemi yapılarak gerçek zamanda aracın kendi kendine otonom park yapması sağlanacaktır.

2. OTONOM ARAÇ TASARIMI

Bu bölümde, otonom araç tasarımı için ihtiyaç duyulan araç gövdesi, aracın üstüne koyulan kontrol kartları, kullanılan motorları, sürücü kartları, kamera modülü, devre şemasını ve haberleşme modülünü neden tercih edildiğini, özelliklerini ve ne için kullanıldıkları anlatılmıştır.

2.1. Kullanılan Mekanik-Elektronik Modüller

Burada araç gövdesi ve üzerine yerleştirilmiş elektronik-mekanik modüllerin ve cihazların özellikleri açıklanmıştır.

2.1.1. Aracın Gövdesi

Otonom aracın ana yapıtaşını olan araç gövdesi tercih edilmesindeki dikkat edilen noktalar, aracın hafif olması yani motorların üzerine düşecek kuvvetin az olması, böylece pil ömrünün daha uzun gitmesi sağlanması, maddi olarak ucuz olması, her türlü zemin için esnek olmasıdır. Bu sayede aracın park etmesi her türlü ortam için daha kolay olacaktır. Bu sebeple Rock Crawler 1/20 ölçekli 4 çeker elektrikli aracı tercih edilmiştir. Uzun ve arazi yolları için tasarlanmış amortisörü ile kararlı bir sürüş sağlar. 2.4 Ghz frekans genliğine sahip bu aracın kontrol mesafesi de 50 metredir. Kullanılan motor yükte iken 0.3A akım çeker iken, 3-6 V gerilim arasında çalışabilmektedir. Toplam ağırlığı 17 gram sahip bu motor 15km/s hıza çıkabilmektedir. Çok az güç tükettiği için 4.8V 700 mAh Ni-Cd pil kullanılmıştır. Şekil 2.1' de araç gösterilmiştir.

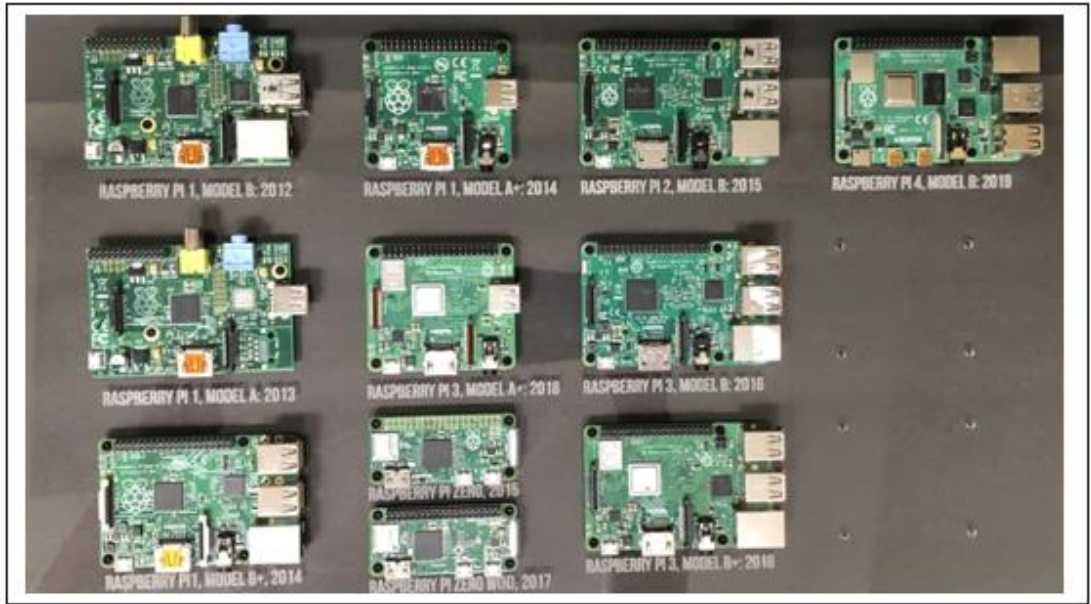


Şekil 2.1: Rock crawler elektrikli aracı.

2.1.2. Mikrobilgisayar

Raspberry Pi Birleşik Krallık'ta Raspberry Pi Vakfı tarafından okullarda bilgisayar bilimi öğretmek amacıyla geliştirilmiş kredi kartı büyüklüğünde tek kartlı bir bilgisayardır. İlk modeli RPi 1 2012 yılında, ikinci modeli RPi 2 2015 yılında, üçüncü modeli RPi 3 A ve RPi 3 B 2016 yılında, RPi 3B+ 2018 yılında ve son olarak RPi 4 de 2019 yılında piyasaya sunulmuş olup yine bu süre farklı kartlarda üretilmiştir. Bir geliştirme kartı dışında monitör ve klavye ile bir bilgisayara dönüşebilir.

RPİ işletim sistemi, Linux tabanlı işletim sistemi olan Raspian'dır. Bu işletim sistemi dışında, arch linux ARM, LibreELEC, OSMC ve Windows 10 IoT Core sistemlerini de çalıştırabilmektedir. Programlama dili olarak Python kullanılsa da C ve Perl programlama dili ile program geliştirilebilir. Piyasaya sürülen bütün RPI modelleri Şekil 2.2 de gösterilmiştir [32].



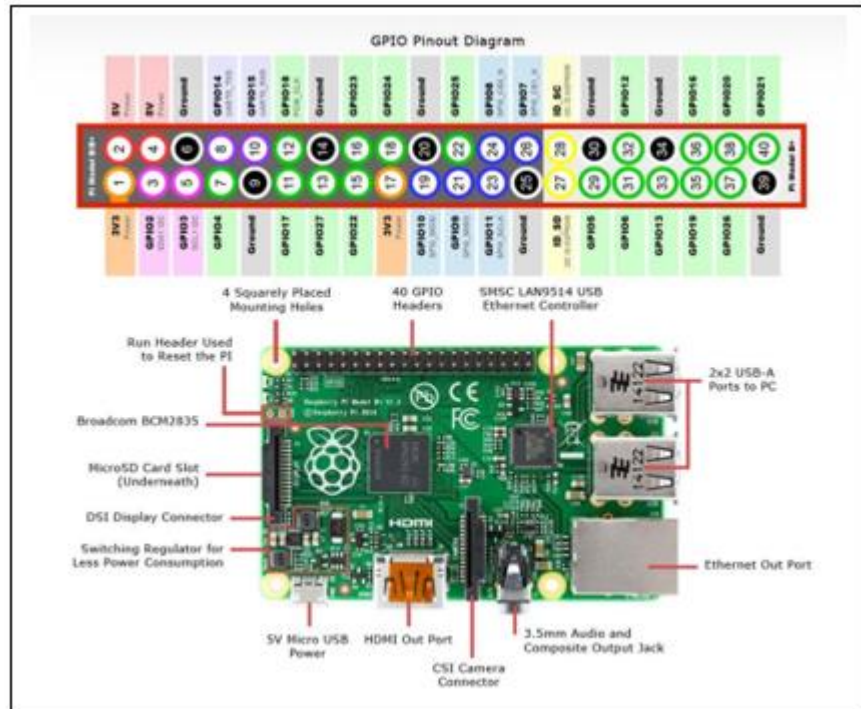
Şekil 2.2: Piyasaya sürülen bütün rpi modelleri.

Bu çalışmada tercih ettiğimiz model 2018 yılında piyasaya sürülmüş olan RPi 3 B+ olan modeldir. Bu model en yeni özelliklere sahip olması yanı sıra yüksek işlemci gücüne sahip ve Python programlama dilini desteklemektedir.

Sistem içerisinde, 1.4 Ghz, 4 çekirdekli 64 bit ARM Cortex-A53 işlemci ve Broadcom BCM2837B0 mikroişlemcisi vardır. 1 GB bellek kapasitesine sahip bu kart 2 çekirdekli Videocore IV GPU grafik birine sahiptir. Üzerinde 4 adet USB 2.0 port,

300 Mbit/s Ethernet portu ve HDMI çıkışı vardır. Ayrıca dahili olarak gelen Wi-Fi 2.4 GHz+5GHz WLAN ve Bluetooth 4.2, BLE özellikleri sayesinde uzaktan kontrol edilmesine olanak sağlar.

RPİ 3B+ model bu kart, 40 tane GPIO pinine sahiptir. Bu pinlerden 2 tanesi 5V, 8 tanesi GND, 2 tanesi 3.3V ve son 2 tanesi ise EEPROM için ayrılmıştır. Kalan 26 pin ise giriş ve çıkış pinleridir. Bu 26 pinin 5'i SPI haberleşmesi, 2'si I2C haberleşmesini ve 2 tanesi de UART haberleşmesini desteklemektedir. Aşağıdaki şekil RPİ 3B'e ait olsa bile gösterilen bölge için 3B+ modelinden farklı değildir [15].



Şekil 2.3: RPİ bağlantı diyagramları.

Raspberry Pi 3B+ Teknik Özellikleri:

- 1.4 GHz, 1 GB RAM
- Wi-Fi 2.4GHz ve 5GHz destekli IEEE 802.11, Bluetooth 4.2, BLE destekli
- 40 Adet GPIO
- 4 Adet USB 2
- 4 uçlu Stereo çıkışı ve Kompozit video çıkışı
- Full HDMI

- RPI Kamera bağlantısı için CSI kamera portu
- RPI 7” dokunmatik ekran için DSI ekran portu
- Micro SD soketi
- 5V/2.5A dc GÜÇ girişi
- Güç ve aktivasyon ledi.

2.1.3. Motorlar

Bu aracı tasarlarken kullanılan araç içerisinde iki tane DC motor vardır. Bunlardan ilki aracın ön kısmına yerleştirilmiş ve araca yön vermesini sağlayan motor ve ikinci olarak da aracın arka bölümünde bulunan ve araca hız veren dc motordur.

2.1.4. Motor Sürücüsü

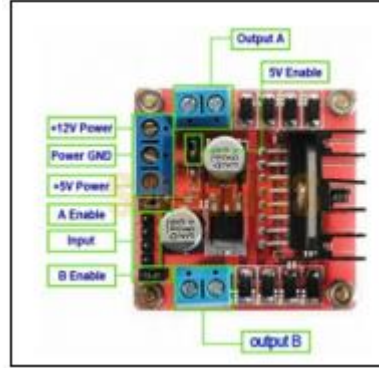
Aracın kontrolcü bölmesinde bulunan RPI ve Arduino Uno kartları vardır. RPI de PWM giriş pinleri bulunmadığından, Arduino Uno da ise bu pinler olmasına rağmen karttan çok fazla akım çekebileceğinden motor ile mikro kontrolcüler arasında motor sürücüsü kullanmak gereklilik olarak ortaya çıkmıştır. Burada bu konuda en çok tercih edilen motor sürücülerinden biri olan L298N motor sürücü devresi tercih edilmiştir. Aracımızdaki iki motoru

Resim 2.4 ‘deki A ve B uçlarına bağlayarak aracımızın gaz ve direksiyon kontrolünü yapmamız sağlanmıştır.

L298N DC Motor Sürücü Kartı Özellikleri:

- Birbirinden bağımsız olarak iki ayrı motoru kontrol edebilir.
- Kanal başına 2A akım verebilmektedir.
- Üzerinde dahili regülatörü vardır.
- Yüksek sıcaklık ve kısa devre koruması vardır.
- Motor dönüş yönüne göre yanan ledler vardır.
- Kart üzerinde dahili soğutucu vardır.
- Akım okuma (current sense) pinleri dışa verilmiş haldedir.

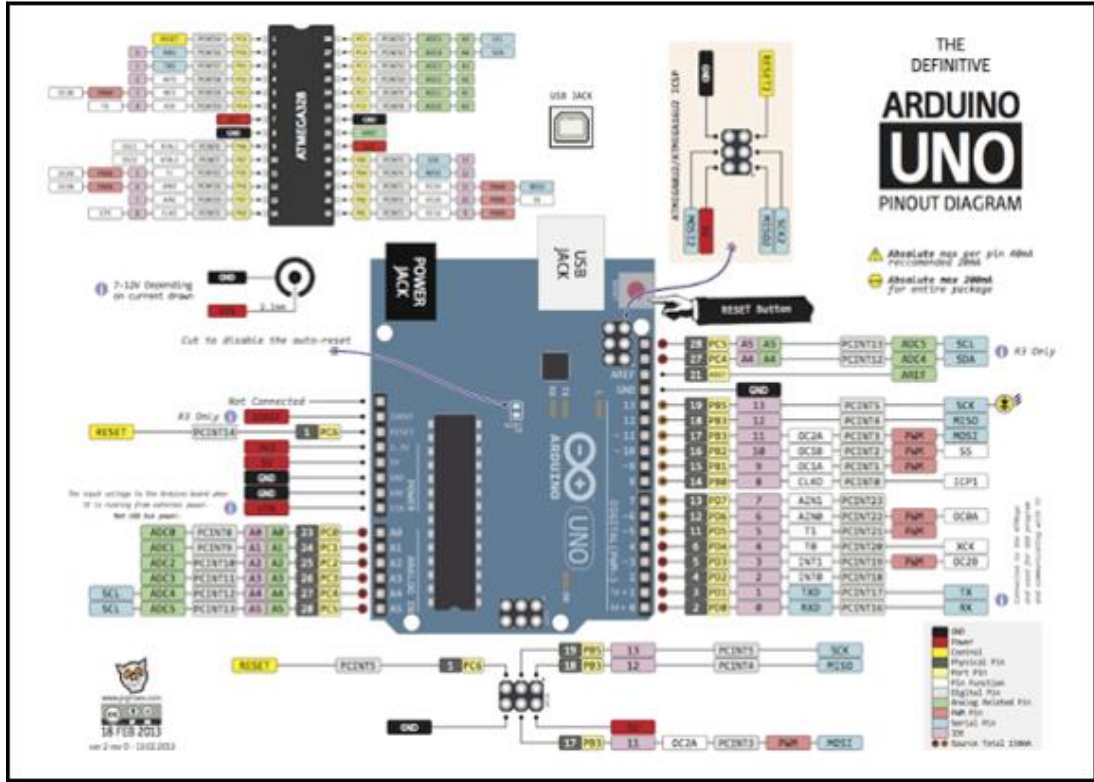
- Kartın 4 yanında istenilen yüzeye sabitleyebileceğiniz 4 adet vida deliği bulunmaktadır.



Şekil 2.4: RPi L298N dc motor sürücü kartı.

2.1.5. Arduino Uno

2005 yılında Massimo Banzi ve David Cuartielles tarafından geliştirilen Arduino; bir Giriş /Çıkış kartından ve açık kaynaklı bir donanım teknolojisinden oluşan bir fiziksel programlama platformudur. Arduino, Arduino-IDE geliştirme ortamında basitleştirilmiş C tabanlı programlama dili ile programlanabilir. Arduino tek başına çalışan interaktif nesneler geliştirmek için kullanılabileceği gibi bilgisayar üzerinde çalışan yazılımlara da bağlanabilir. Arduino kartları bir Atmel AVR mikro denetleyiciden ve programlama ve diğer devrelere bağlantı için gerekli yan elemanlardan oluşur. Her kartta en az bir 5 voltluk regüle devresi ve bir 16 MHz kristal osilator bulunur [16]. Arduino, kolay programlanabilmesinin yanında ucuz ve kolay ulaşılabilir olması nedeniyle de büyük kullanıcı kitlelerine hitap etmektedir.



Şekil 2.5: Arduino uno bağlantı şeması.

Tablo 2.1: Arduino uno özellikleri.

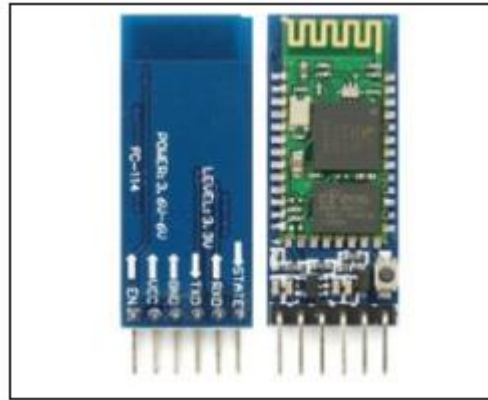
– Mikro denetleyici	– ATmega328
– Çalışma Gerilimi	– 5V
– Giriş Gerilimi (önerilen):	– 7-12V
– Giriş Gerilimi (limit):	– 6-20V
– Dijital G/Ç Pinleri:	– 14 (6 tanesi PWM çıkışı)
– Analog Giriş Pinleri:	– 6
– Her G/Ç için Akım:	– 40 mA
– 3.3V Çıkış için Akım	– 50 mA
– Flash Hafıza:	– 32 KB (ATmega328)
– SRAM, EEPROM:	– 2 KB, 1 KB
– Saat Hızı:	– 16 MHz
– Uzunluk, Genişlik:	– 68.6 mm, 53.4 mm
– Ağırlık:	– 25 g

2.1.6. Haberleşme Modülü

Bir aracı uzaktan haberleşmesini sağlamak en önemli aşamalardan bir tanesidir. Bu haberleşmeyi sağlamak için birçok modül vardır. Bu çalışma uzaktan bağlantı ayarları kolay olması, haberleşmede bir sorun yaşanmaması, ucuz olması sebebiyle HC-05 Bluetooth modülü tercih edilmiştir. Bu modül, bluetooth 2.0.'ı destekleyen 2.4 GHz frekansında kablosuz haberleşme uygulamaları için tasarlanmıştır. 10 metrelik haberleşme mesafesine sahiptir. HC05 modül düşük güç ile çalışabilmekte ve 3 V ile 5 V arasında kullanılabilir. Bu modülün tasarlanan robot üzerinde temel görevi, seri portun yerini alabilmektedir. Modül RX ve TX pinleri üzerinden PC veya ikinci bir mikro denetleyici ile kablosuz haberleşme protokolünü uyguladığında yöneten bölüm “Master” yönetilen bölüm ise “Slave” olarak iletişim sağlar.

Tablo 2.2: HC-05 arduino bluetooth modül.

– Özelleştirme:	– Evet
– Güç:	– Düşük Güç
– Band:	– 2.4GHz
– Güç girişi:	– + 3.6V ~ 6VDC
– Spp modülü:	– VAR
– Çıkış Gücü Sınıfı:	– Sınıf2
– Bluetooth Özellikleri:	– VER2.1 + EDR



Şekil 2.6: HC-05 bluetooth modülü.

2.1.7. Kamera Modülü

Otonom araç tasarımı bir diğer önemli nokta aracın önünü görmesidir. Bunun için kullanılan kamera modülü, model eğitiminde kullanılmak üzere ilk olarak görüntülerin işlenmesinde, anlık video görüntüsünü ana bilgisayara aktarılmasında ve bu görüntülerin veri olarak kaydedilmesinde kullanılır. Kamera modülü için RPi ile uyumlu yüksek çözünürlüğe sahip Pi kamera V1.3 seçilmiştir. 5 mega piksel çözünürlüklü kamera ile 1080p video ve sabit fotoğraf çekme kapasitesine sahiptir ve RPi’ a doğrudan bağlanabilecek şekilde tasarlanmıştır. 2592x1944 piksel statik resim çözünürlüğü sunan kamera video çekimlerinde ise 1080p30, 720p60 çözünürlüğünü desteklemektedir [17].

Tablo 2.3: Raspberry pi kamera V1.3 teknik özellikler

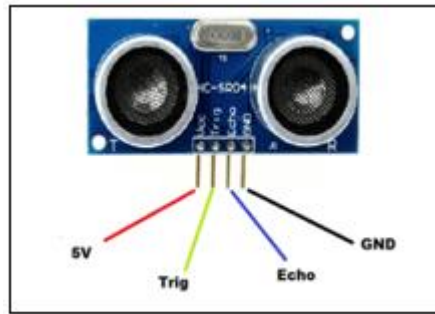
– Sensör Tipi:	– OmniVision OV5647 (5 megapiksel)
– Sensör Boyutu:	– Sensör Boyutu: 3.67 x 2.74 mm
– Piksel Sayısı:	– 2592 x 1944
– Piksel Boyutu:	– 1.4 x 1.4 um
– Lens:	– f = 3,6 mm, f / 2,9
– Görüş Açısı:	– 54 x 41 derece
– Görüş Alanı:	– 2 m'de 2.0 x 1.33 m
– Tam kare SLR lens eşdeğeri:	– 35 mm
– Sabit Odak:	– Sonsuza 1 m
– Video:	– Codec H.264 (AVC) özellikli 30 fps'de 1080p
– Pano boyutu:	– 25 x 24 x 9mm (esnek kablo dahil değil)



Şekil 2.7: Rpi kamera V1.3 modülü.

2.1.8. Mesafe Sensörü

Bu çalışmada seçilen mesafe sensörü 40 Khz frekansında ultrasonik ses üreten HC-SR04 sensörü seçilmiştir. Ses dalgaları sınıflandırılmasında 20 Khz- 1 Ghz aralığındaki ses sinyalleri ultrasonik sinyaller tanımlanmıştır. Burada önemli olan sesin yüksekliğinde belirleyici olan etken frekanstır. Sensörden çıkan sesin gidiş geliş süresine göre mesafe hesaplanır.



Şekil 2.8: HC-SR04 ultrasonik sensör.

2.1.9. Güç Kaynağı

Bu araçta gerekli olan bir diğer önemli konu da güç desteğidir. Bu aracın hem motorlarını hem de Raspberry Pi mikro bilgisayarını beslemesi için 5000 mAh gücü, 5V'luk çıkış gerilimi ile Dexim powerbank tercih edilmiştir. Bu güç kaynağı RPi 5V ile beslendiği ve dc motoru süren motor sürücü de 5V gerilim aralığında çalıştığı için bu güç kaynağı bu araç için yeterli görülmüştür.



Şekil 2.9: Dexim 5000 mAh powerbank.

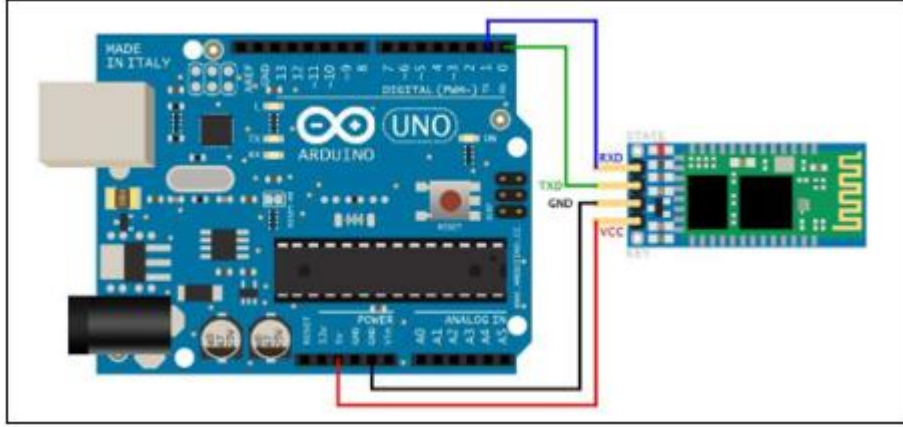
2.2. Bütün Komponentlerin Birleştirilmesi

Bu bölümde, tüm ekipmanlar ve komponentlerin birlikte uyum içerisinde nasıl çalıştığı incelenecektir. İlk olarak aracın içerisinde bulunan kontrolör çıkartılarak sadece gövde, motorlar ve tekerlekler kullanım için ayrılmıştır. Daha sonra bu gövdenin üzerine metal plaka yerleştirilmiştir. Bu plakaya mikrobilgisayar, arduino uno, 1298n motor sürücüsü, 3 adet ultrasonik sensörü, bluetooth modülü, pi kamera, powerbank ve yedek olarak aracın içerisinde var olan pil yerleştirilmiştir. Bu yerleştirmeler yapılırken aracın ağırlık merkezi plakanın tam ortasında olacak şekilde tasarlanmıştır. Bu sayede dengeli yük dağılımı sağlanarak aracın istenmeyen yöne gitmesi engellenmiştir. Plakaya kablo karışıklığını önlemek amacıyla fazladan pinler lehimlenmiştir. Pi Kamera açısını önünü rahatça görmesi için plakaya dikey şekilde sabitlenmiş plastik bir çubuğa koyulmuştur. Ultrasonik sensörler ise önüne çıkan engelleri tespit etmesi daha kolay olması açısından yere yakın olacak şekilde sabitlenmiştir.

Bluetooth bağlantısı yapmak için öncelikle bazı konfigürasyon ayarları yapılması gerekmektedir. İlk olarak yapılması gereken Resim 2.8'deki bağlantılar yapılmalıdır. Daha sonra enerjiyi sıfırlayın key pinini GND' e bağlayıp ve tekrar enerji verilecektir. Aynı şekilde key pinini +5V' a bağlayıp, enerjiyi sıfırladıktan sonra da bluetooth üzerindeki butona basılı tutarken enerji vererek modül üzerindeki ışık yavaşça yanıp sönmeye başlayacak ve konfigürasyon ayarlarını yapmak için hazır hale gelecektir [18].

Haberleşme frekansı değiştirilmediği takdirde 9600 olacak şekilde Arduino IDE'nin seri port kısmına AT komutu yazıldığında OK komutu geldiğinde bluetooth

modülü artık haberleşmeye hazır hale gelecektir. Enerjiyi sıfırlayıp key pinini GND' e tekrar bağladıktan sonra enerji vererek telefonumuza yüklemiş olduğumuz programı aktif ederek bluetooth eşleşmesi yapılmaktadır. Şifre “1234” yazıldıktan sonra artık telefon ile Arduino arasındaki haberleşmesi başarılı bir şekilde sağlanmış olacaktır.

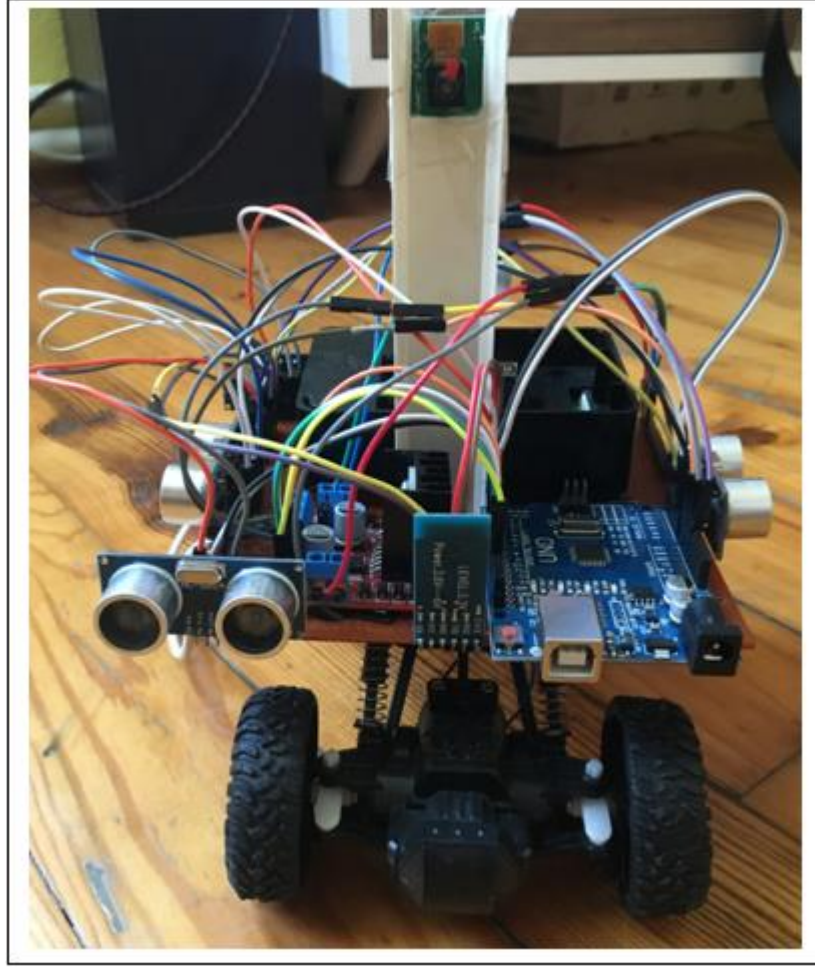


Şekil 2.10: Arduino uno HC-05 bağlantısı.

Kodu yüklemekten önce artık key pini ile bir ihtiyaç kalmadığı için çıkarılacaktır. Aynı şekilde eğer ki Rx ve Tx pinleri 0. ve 1. Pine bağlılar ise onların da bağlantıları kodu yüklemekten önce kaldırılıp, kodu yükledikten sonra da tekrar yerlerine sokulacaktır [18]. Bu işlem her defasında Arduino içerisine atılan program ile tekrarlanacaktır. Aksi halde Arduino kullanılamaz hale gelmektedir. Bu sayede aracımız RC araca dönüştürülerek uzaktan kontrol edilebilecek hale gelmiştir.

Araç sadece kameradan gelen verilerle değil ayrıca ultrasonik sensörlerden gelen veriler sayesinde kontrol edilmektedir. Aracın park işlemini iyi şekilde yapılabilmesi için aracın üç tarafına da mesafe sensörü eklenmiştir.

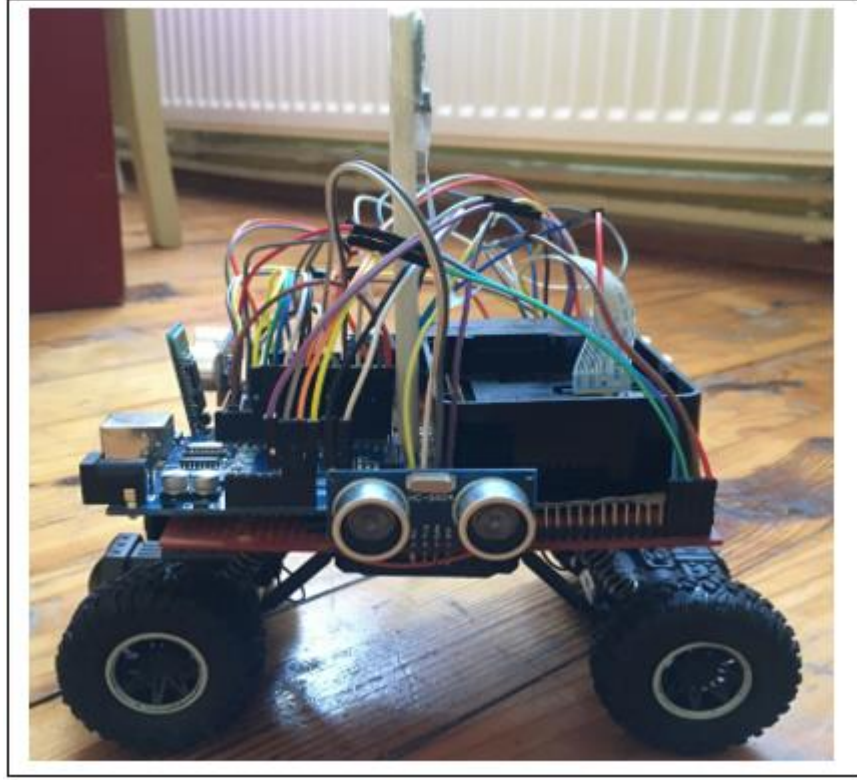
RC otonom aracın önden soldan ve sağdan görünüşleri gösterilmektedir.



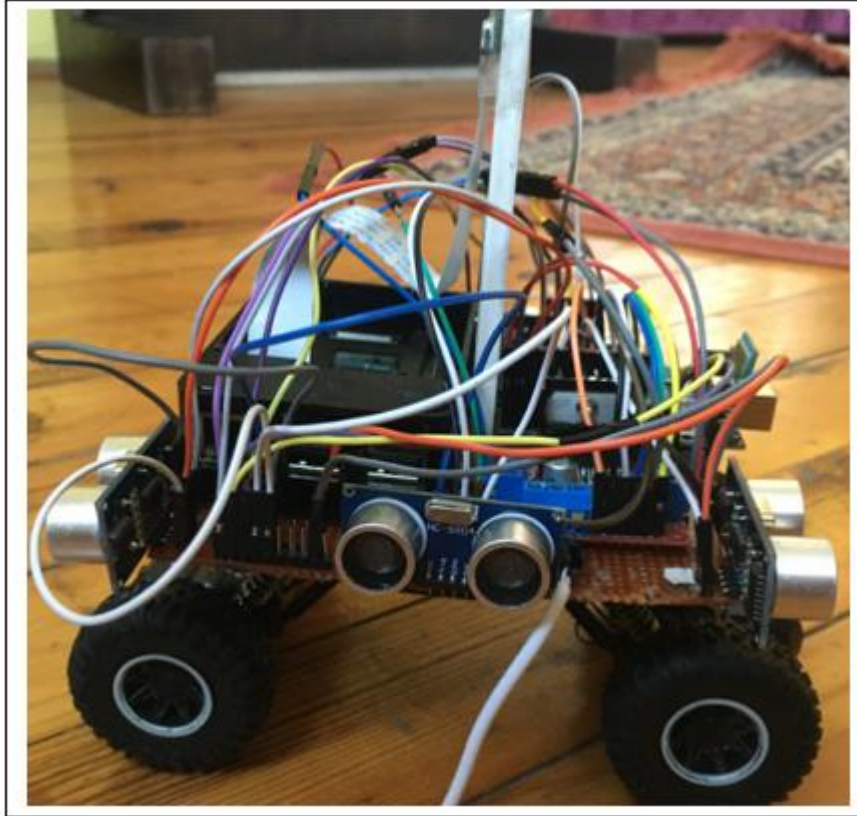
Şekil 2.11: RC otonom araç önden görünüşü.

RC aracın üstünde konumlanan komponentler silikon, vidalar, çift taraflı bant yardımıyla metal tabakaya sabitlenmiştir. Kamera dik açıyla koyulduğunda aracın önünü görmediğinden belli bir açı ile plakanın üzerine silikon ile yapıştırılmıştır. Kameranın yerleştirildiği beyaz çubuğun aracın tam merkezine koyularak maksimum açıda görüntü alması sağlanmıştır. Ayrıca park işlemi yapılacağı içinde kamera 45 derece sağ tarafa çevrilmiş olup, eğitim için daha uygun görüntüler toplanması amaçlanmıştır.

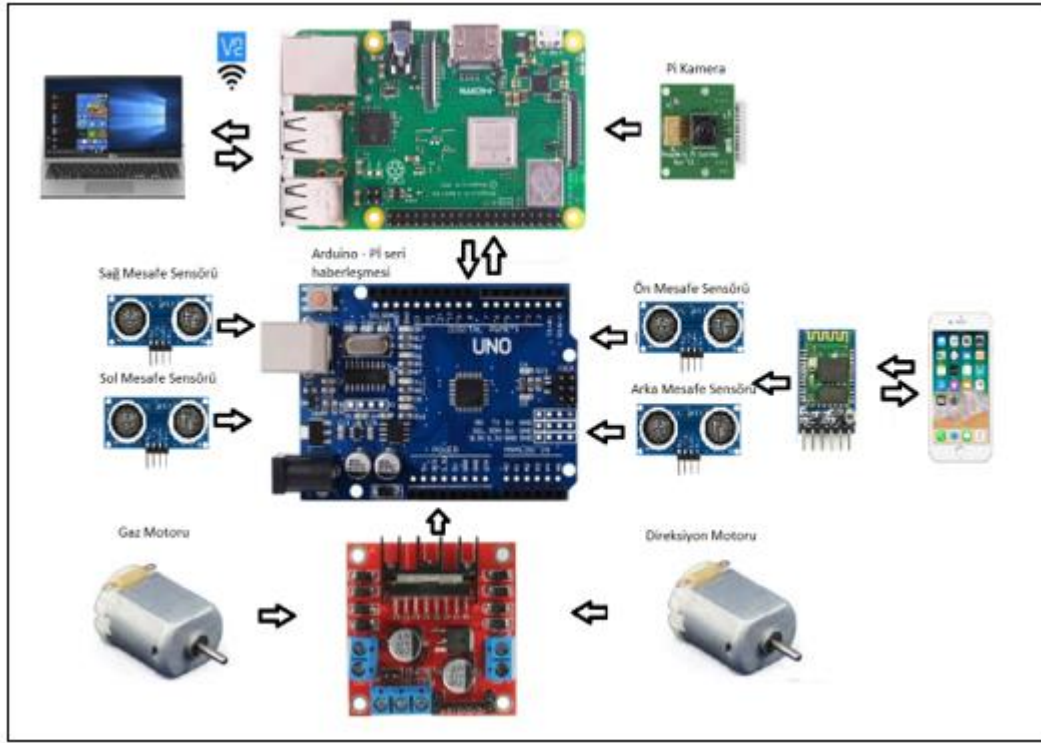
Şekil 2.11, Şekil 2.12, Şekil 2.13’ de aracın farklı açılardan görüntüleri gösterilmektedir. Resim 2.14 de ise sistemin genel bağlantı şeması görülmektedir.



Şekil 2.12: RC otonom araç soldan görünüşü.



Şekil 2.13: RC otonom araç sağdan görünüşü.



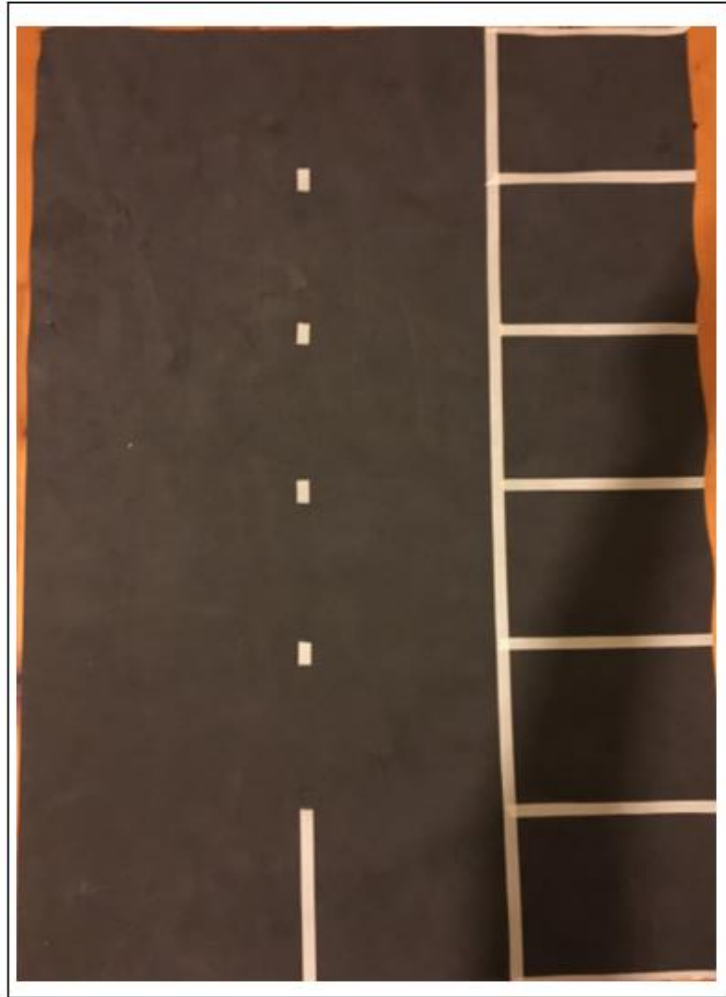
Şekil 2.14: RC araç sisteminin genel bağlantı şeması.

Bu çalışmada, RPi üzerinde yazılım geliştirmek için gerekli olan ekran paneli yerine VNC Viewer programı kullanılmıştır. Bu program sayesinde bazı konfigürasyon ayarları yapılarak aynı ağdan bağlanmak şartıyla aracımıza kendi bilgisayarımızdan rahatça ulaşılabilir. Bu sayede bilgisayarımızda eğitilmiş derin öğrenme modelini kolayca aracımıza monte edebiliriz. Linux işletim sisteminin seçilmesinin sebebi gerçek zamanlı işletim sistemi olması ve bu sayede aracın daha iyi performans vermesine olanak sağlar. Pi kamerayı da RPi içerisindeki konfigürasyon ayarları kısmından aktif ederek aracın pi kamera üzerinden anlık olarak görüntü alması sağlanmıştır. Pi kamera, aracın park alanında verimli bir şekilde görüntü alabilmesi için deneysel olarak belirlenen açıda araca monte edilmiştir. Uzman sürücü tarafından eğitim verisi toplamasına yardımcı olması için araçta bluetooth modülü kullanılmıştır. Bunun yanında aracın direksiyon ve gaz kontrolü işlemi de arduino vasıtasıyla yapılmıştır. Raspberry ve Arduino arasında bir seri haberleşme yapılarak hızlı bir veri alışverişi sağlamak, bu sayede aracın daha hızlı kontrol edilmesi amaçlanmıştır. Mesafe sensörü park manevrası esnasında çevresinde var olan engeller tespit ederek aracın park yapmasını engelleyecek durumların önüne geçmek adına bir çözüm olarak düşünülmüştür.

2.3. Park Alanı Kurulması

Araç tasarımı tamamlandıktan sonra, aracın içine eklenecek olan derin öğrenme modelini eğitmek amacıyla veri setine ihtiyaç duyulmaktadır. Hazır veri seti ile eğitilmiş modellerin gerçek zamanda sonuçları hayal kırıklığı yaratabilmektedir. Bunun olması adına yapılacak ilk şey verinin sıfırdan toplanmasıdır. Bu sayede daha başarılı modelle eğitilebilmektedir.

Bu çalışmada daha başarılı bir model eğitmek için bir parkur tasarlanmıştır. Bu parkur aracın enine boyuna ve manevra kabiliyetine bakılarak deneysel verilerden elde edilerek yapılmıştır. Aracın yaklaşık boyu 20 cm, eni ise 15 cm'dir. Aracın manevra kabiliyeti göz önüne alındığında park alanındaki her boş park yeri boyu 30 cm eni ise 25 cm olarak tasarlanmıştır. Parkur toplamda 1,5 m boy ve 1 m eni sahiptir. Hem çapraz park hem de dik park olmak üzere iki farklı tipte parkur hazırlanmıştır.



Şekil 2.15: Dik park alanı yukardan görünüşü.

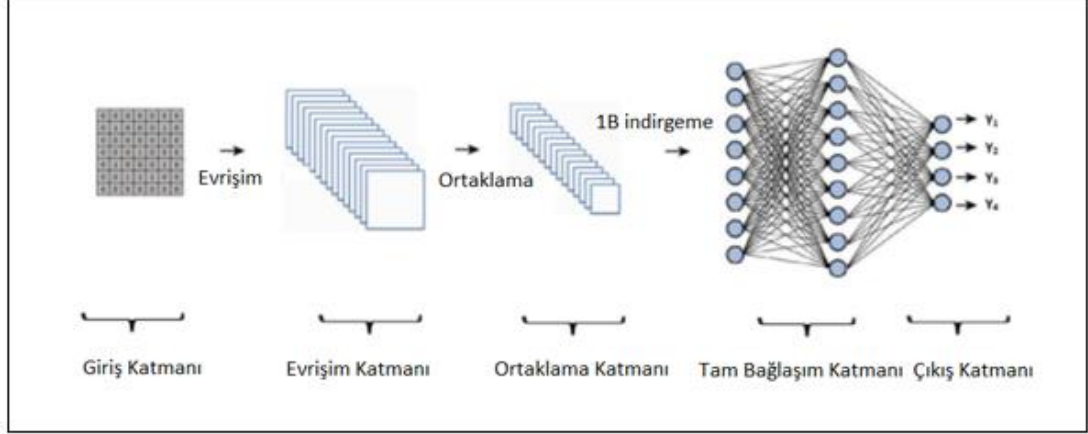


Şekil 2.16: Çapraz park alanı yukardan görünüşü.

Şekil 2.15 ve Şekil 2.16 de sırasıyla dik park alanı ve çapraz park alanı gösterilmiştir. Tasarlanan araç bu parkurlarda uzman kullanıcı tarafından park işlemi yaptırılarak eğitim verileri toplama işlemi gerçekleştirilmiştir.

3. EVRİŞİMLİ SİNİR AĞLARI

3.1. Genel bilgiler



Şekil 3.1: Evrişimli sinir ağı genel mimari gösterimi.

Evrişimli sinir ağı (ESA), dünyadaki en gelişmiş duylardan biri olan görme duylusunun çalışması prensibi temel alınarak yaratılmış bir derin öğrenme algoritmasıdır. Çok katmanlı sinir ağlarının bir alt alanı olan evrişimli sinir ağı'nın mimarisi diğerlerinden farklı olarak ekstra katmanlara sahiptir. Bu ağı mimarisi Yann LeCun tarafından 1998 yılında ortaya konulmuştur [19]. Bu ilk çalışmada el yazısıyla yazılmış olan rakamlar üzerinde oldukça başarılı sonuçlar alınmıştır. Bu başarı derin öğrenme modellerinin günümüzde çokça kullanılmasının temelini atmıştır. Resim 3.1 de Evrişimli sinir ağlarının temel mimarisi görülmektedir. Bu ilk mimari, 2 evrişim ve 2 ortaklama katmanı olmak üzere 4 katmanlı Lenet modeli olmakla birlikte, yıllar içinde gelişen işlemci gücü sayesinde daha başarılı mimariler geliştirilmiştir. Bunlardan ilki 2012 yılında ImageNet (ImageNet Large Scale Visual Recognition Challenge – ILSVRC) yarışmasında veri seti olarak 1000 sınıflı 1,2 milyon eğitim verisi kullanılmıştır. Bu yarışmada en iyi model Lenet modelden daha derin bir mimariye sahip olan 8 katmanlı AlexNet olmuştur [20]. 2014 yılında ise 19 katmanlı VGGNet mimarisi, hata oranını %6,8' ye kadar düşürmeyi başarmıştır [21]. İnsan performansının da %5 de olduğu bilindiğinden insana yakın bir başarı sağlanmıştır. [22]. VGGNet aynı zamanda kendi içinde 11,13,16 katmanlı olarak da kullanılabilir. Yine aynı yıl içinde Google tarafından geliştirilen GoogleNet

mimarisi de hatayı %6,7'e düşürerek aynı başarıyı yakalamıştır [23]. 2015 yılında ise Microsoft tarafından geliştirilen 152 katmanlı ResNet mimarisi hatayı %3.57 düşürerek insandan bile daha iyi bir performansa ulaşmıştır [24].

3.2. Evrişimli Sinir Ağ Mimarisi

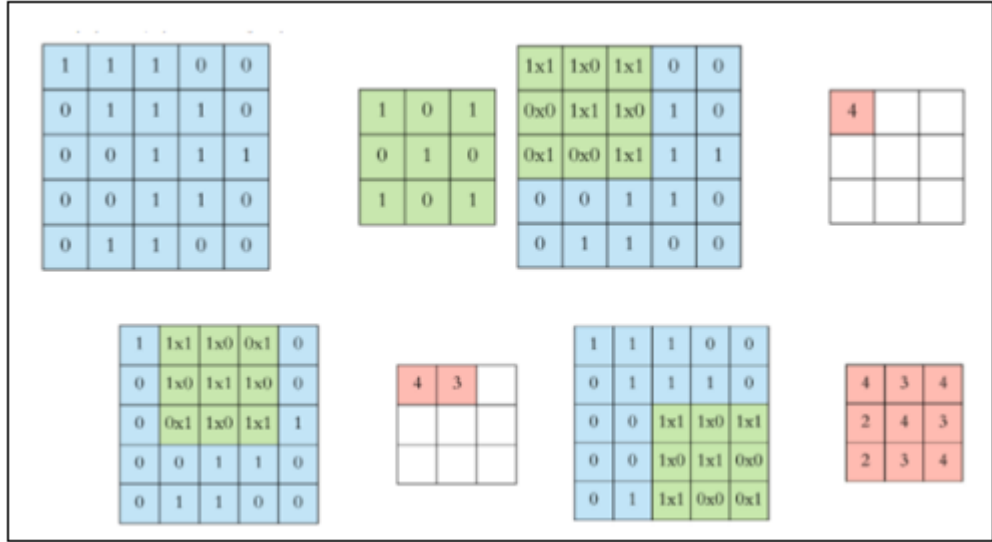
ESA mimarisinde temel olarak giriş katmanı, evrişim katmanları, ortaklama katmanları, tam bağlantı katmanları ve son olarak çıkış katmanı bulunmaktadır. Bunların yanında aktivasyon fonksiyonları, seyreltme işlemleri, yığın normalizasyonları ve filtreleme işlemleri yapılmaktadır.

3.2.1. Giriş Katmanı

Giriş katmanı, verilerin uygun biçimde bulundurulması, veriler üzerinde özellik ve etiket işlemleri, daha az hesaplama yapması için verilerin çözünürlüğünün düşürülmesi, veri normalizasyonu yapılması, bunlara ek olarak verilerin çoğaltılması gibi veri üzerinde birçok işlemin yapıldığı katmandır.

3.2.2. Evrişim Katmanı

Giriş katmanında etiketlenmiş veriler evrişim katmanında belirlenen katman, kernel filtresi ve adım değerleri kadar işlemde geçirilerek öznitelik haritasının çıkarıldığı katmandır. Bu katman en fazla hesaplama gücünün olduğu ve enerjisinin harcandığı katmandır.



Şekil 3.2: 5x5 girdi ve 3x3 filtre ile öznetelik çıkarımı.

3.2.3. Ortaklama Katmanı

Ortaklama katmanı, tipik olarak evrişim katmanından sonra uygulanan örnekleme işlemidir. Öznetelik haritasının boyutunu azaltmak ve işlem gücü düşürmek amacıyla maksimum veya ortalama değer alma işlemleri yapılır. Böylece daha hızlı eğitim yapılmasına yardımcı olur.



Şekil 3.3: 4x4 girdi ile maksimum ve ortalama havuzlama.

3.2.4. Tam Bağlantı Katmanı

Tam bağlantı katman, her girişin tüm nöronlara bağlı olduğu bir giriş üzerine çalışır. Önceki katmanlardan elde edilen öznitelik haritası bu katmanda kullanılarak tek boyutlu ve sınıflardan oluşan çıktılardan meydana gelmektedir. Eğer varsa, FC katmanları genellikle CNN mimarisinde softmax katmanından hemen önceki katmanda bulunur. Bundan sonraki katmanda ise sadece sınıflandırma işlemi yapılır.

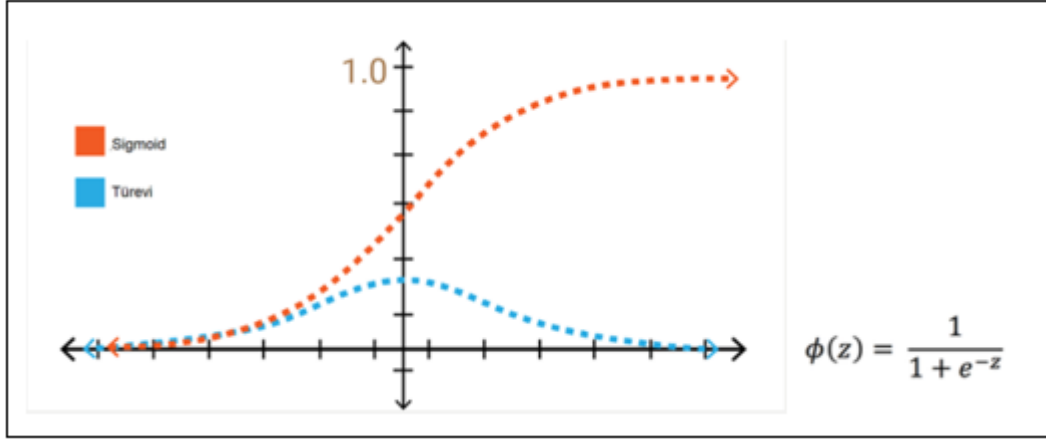
3.2.5. Çıkış Katmanı

Çıkış katmanı, tam bağlantı katmanından istenen sınıf sayısına indirgenmesi ile belirlenmiş sınıfların softmax fonksiyonu yardımı ile çıkışta 1 ya da 0 değer aralığında değerler çıkararak sınıflandırma işleminin yapılmasının sağlandığı katmandır. Sigmoid gibi sınıflandırma işlemlerinde oldukça başarılı olan softmax fonksiyonları 2'den fazla sınıflandırma gerektiğinde sigmoidlerden daha iyi performans göstermektedir. Bu yüzden de çok katmanlı ağların çıkış katmanlarında sıklıkla kullanılmaktadır.

3.2.6. Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları, nöronların öğrenmesi için hayati öneme sahiptir. Aktivasyon fonksiyonları olmadan öğrenme işlemi yapılamaz. Aktivasyon fonksiyonları çıktının 0 ile 1 arasında olasılıksal değerler arasında bırakarak daha iyi bir öğrenme işlemi sağlar. Doğrusal ve doğrusal olmayan aktivasyonları olarak kendi içlerinde ayrılırlar. Bu bölümde en çok kullanılan aktivasyon fonksiyonları gösterilmiştir [31].

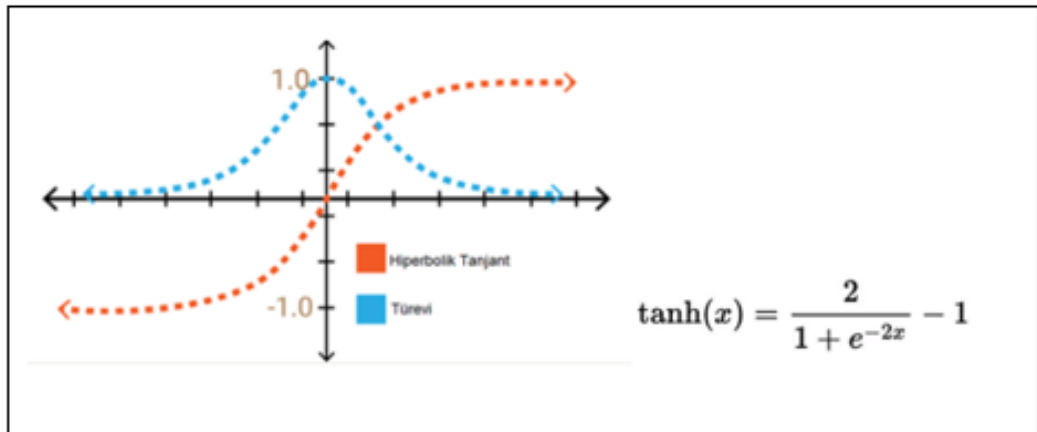
3.2.6.1. Sigmoid Fonksiyonu



Şekil 3.4: Sigmoid fonksiyonu ve türevi.

Sigmoid fonksiyonu, doğrusal olmayan türevlenebilir bir fonksiyondur. X eksenindeki değerinin ufak değişimleri ile y de etkisi çok büyük olabilmektedir. Bu da öğrenme işlemine olumlu yönde etkilemektedir. Bu yüzden sık kullanılan aktivasyon fonksiyonlarından biri olmayı başarmıştır. Bununla beraber sigmoid fonksiyonun türevindeki uç noktalara bakıldığında 0'a yakınsamaktadır. Buna gradyan ölmesi ya da gradyan kaybolması denmektedir. Bu durumda öğrenme olayı en düşük seviye de kalır. Yavaş öğrenme olduğunda hatayı minimum yapan optimizasyon algoritması lokal minimum değerlerine takılabilir ve yapay sinir ağı modelinden maksimum performans alınmaz.

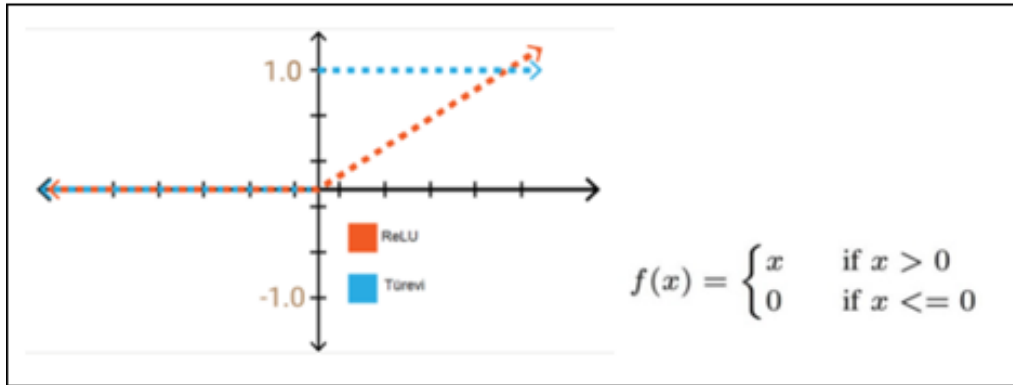
3.2.6.2. Hiperbolik Tanjant Fonksiyonu



Şekil 3.5: Hiperbolik Tanjant fonksiyonu ve türevi.

Sigmoid fonksiyonu ile benzerlik göstermektedir. Sigmoid fonksiyonu 0 ile 1 arasında değer alırken, hiperbolik tanjant fonksiyonu -1 ile 1 arasında değer almaktadır. Bu fonksiyonun türevi daha dik olması ile öğrenme aralığı artacağından daha verimli bir öğrenme işlemi yapılabilir. Ancak türevinden uçlarındaki gradyan ölmesi problemi devam etmektedir.

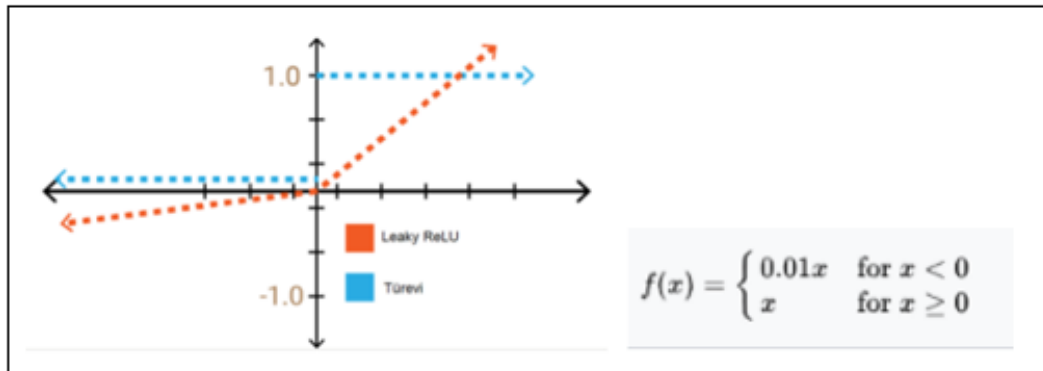
3.2.6.3. ReLU Fonksiyonu



Şekil 3.6: ReLU fonksiyonu ve türevi.

ReLU fonksiyonu, evrişimli sinir ağılar mimarisinde en sık kullanılan aktivasyon fonksiyonlarından biridir. Bunun sebebi ReLU fonksiyonun türevi pozitif değerler için 1 olduğundan çok katmanlı ağlarda çok iyi bir öğrenim sağlar. Ancak negatif tarafta kat sayısı 0 olduğundan öğrenilme olmaz.

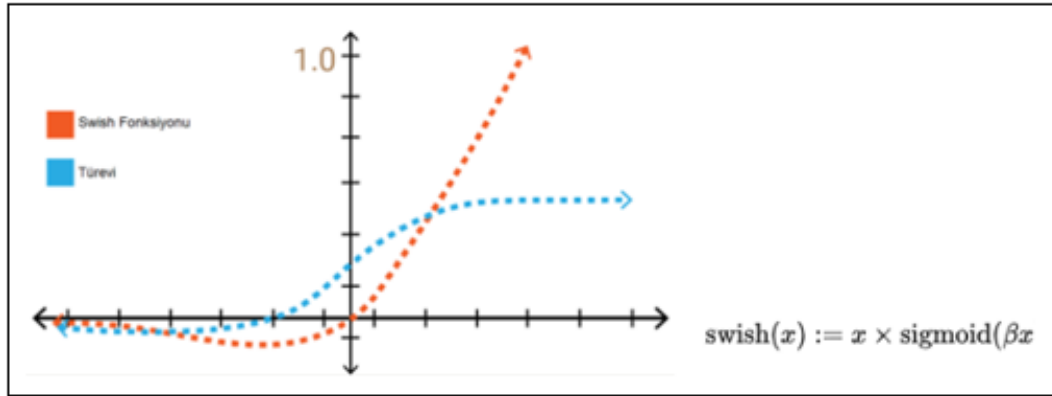
3.2.6.4. Sızıntı ReLU Fonksiyonu



Şekil 3.7: Sızıntı ReLU fonksiyonu ve türevi.

Sızıntı ReLU aktivasyon fonksiyonu, ReLU fonksiyonunun öğrenme yapmadığı negatif tarafta sızıntı denilecek kadar katsayı sağlamaktadır. Bu katsayı sayesinde negatif tarafta da öğrenilme işlemi devam etmiş olur. Böylece çok katmanlı ağlarda daha iyi öğrenme performansı sağlanır.

3.2.6.5. Swiss Fonksiyonu



Şekil 3.8: Swiss fonksiyonu ve türevi.

Swiss fonksiyonu, sızıntı Relu fonksiyonundan farklı olarak negatif bölgede aldığı değerler doğrusal değildir. Bu açıdan bakıldığında negatif yerlerde oluşan gradyan ölmesine çözüm bulunmuştur. Ancak pozitif tarafta beta çok büyük seçilirse doğrusal davranır ve öğrenme seviyesi düşebilir.

3.2.7. Seyreltme Katmanı

Evrişimli sinir ağları gibi çok katmanlı ağlarda oluşabilecek en büyük sorunlardan biri modele eğitim sırasında aşırı öğrenme işlemi yapılmasıdır. Aşırı öğrenme sonucunda model eğitim veri setinde çok iyi performans gösterecektir. Ancak test verisinde performansı çok düşük olacaktır. Bunun sebebi modelin eğitim verilerini ezberlemiş olması oluşudur. Bu problemi engellemek adına en çok yapılan işlemlerin başında seyreltme işlemi gelmektedir. Bu işlem eğitim sırasında rastgele şekilde nöronları dondurma işlemi yapılarak olur. Böylece her batch adımda farklı veri seti kullanılarak ezberleme engellenmiş olur. Bunu yaparken dikkat edilmesi gereken nokta çok fazla seyreltme işlemi yapılırsa eğitime girecek verilerin azalmasından dolayı modelin iyi eğitilememesine ya da çok düşük alınırsa da ezberlemesine çözüm

getirmemesine sebep olabilir. Sonuç olarak seyreltme katsayısı belirlenirken bu şartlar dikkate alınmalıdır.

3.2.8. Paket Normalizasyonu

Nığın Normlar, evrişimsel sinir ağlarını daha verimli ve düzenli hale getirmek için kullanılan bir diğer yöntemdir. Eğitim sırasında meydana gelen gradyan kaybolması durumuna direnç göstererek eğitim süresinin kısalmasına ve bu sayede modelin daha iyi performas göstermesine olanak sağlar.

3.2.9. Maliyet ve Optimizasyon Fonksiyonları

Maliyet fonksiyonu tahmin edilen değerin, gerçek değerinden ne kadar uzak olduğunu hesaplamak için kullanılır. Yapılması istenen bu değeri sıfıra yaklaştırmaktır. Daha sonra da optimizasyon ile model güncellenmeye devam eder. Maliyet fonksiyonlarına L2 loss fonksiyonu ve cross entropy loss fonksiyonu örnek verilebilir. Loss fonksiyonlarında en önemli parametre öğrenme hız oranıdır. Öğrenme hızı oranı küçük seçilirse belirlenen yönde çok küçük adımlar atılır ve bu da hesap yükü olarak geri dönecektir. Ancak öğrenme oranı büyük seçilirse büyük adımlar atılacak olup kayıp büyük olacaktır ve lokal minimum değerine ırsayacaktır. Genel olarak 0'a yakın bir öğrenme oranı seçilmektedir. Maliyet fonksiyonlarında lokal minimumu bulmak için farklı optimizasyon fonksiyonları kullanılmaktadır. Bunların arasından en başarılı olarak da Stokastik gradyan iniş (SGD) fonksiyonu göze çarpmaktadır. Bunun sebeplerinden birisi veri setini mini gruplara bölünerek 1'den fazla iterasyon yaparak kaybı minimuma indirir. İdeal iterasyon değeri ve grup sayıları belirlendiğinde lokal minimum değerine ulaşma olasılığı o kadar artar. Bunlara ek olarak adam, RMS, Adadelata gibi alternatif optimizasyon fonksiyonları da kullanılmaktadır.

3.3. Nesne Tanıma ve Sınıflandırma

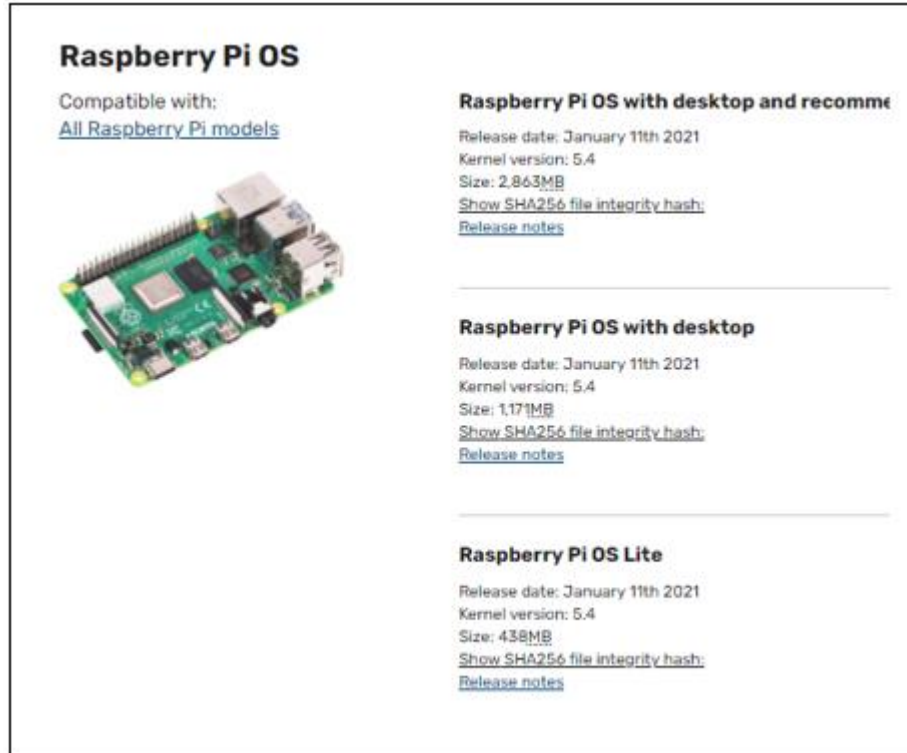
Evrişimli sinir ağları görüntü işleme alanında çok başarılı sonuçlar vermektedir. Bu sinir ağlarının oldukça iyi çalıştığı konulardan biri de nesneleri tanıma ve sınıflandırma çalışmalarıdır. Nesne tanıma ve sınıflandırma için birçok farklı mimariye sahip çok katmanlı sinir ağları modelleri geliştirilmiştir. Bu konudaki çalışmalarda en çok kullanılan mimarilerden bazıları R-CNN, hızlı R-CNN, daha hızlı R-CNN, SSDlite MobileNet ve diğer gelişmiş versiyonlarıdır. Bu modeller arasında en yüksek başarı oranına sahip model Faster R-CNN'dir. [25]. Bu mimarilerin tercih edilmesi, hızın önemli olduğu ya da başarının önemli olduğu uygulamalara göre farklılık gösterebilir. Örnek olarak, gerçek zamanlı sistemler uygulamalarında, hızın ve başarının ön planda olduğu uygulamalarda Faster R-CNN mimarisi kullanılabilir. Ancak Rasppery Pi gibi işlemci gücü daha düşük sistemlerin kullanıldığı uygulamalarda, hızın önemli olmaması ve başarının önemli olması sebebiyle SSDlite MobileNet mimarisi tercih edilmektedir [26].

4. MODEL EĞİTİMİ VE YAZILIM GELİŞTİRİLMESİ

Bu bölümde tasarlanmış aracın işletim sisteminin kurulumu, yazılım geliştirme için gerekli kütüphanelerin ve ara yüzlerin yüklenmesi, veri toplanması, etiketlenmesi, model geliştirmesi ve eğitilmesi ve son olarak da karşılaştırmalar anlatılacaktır.

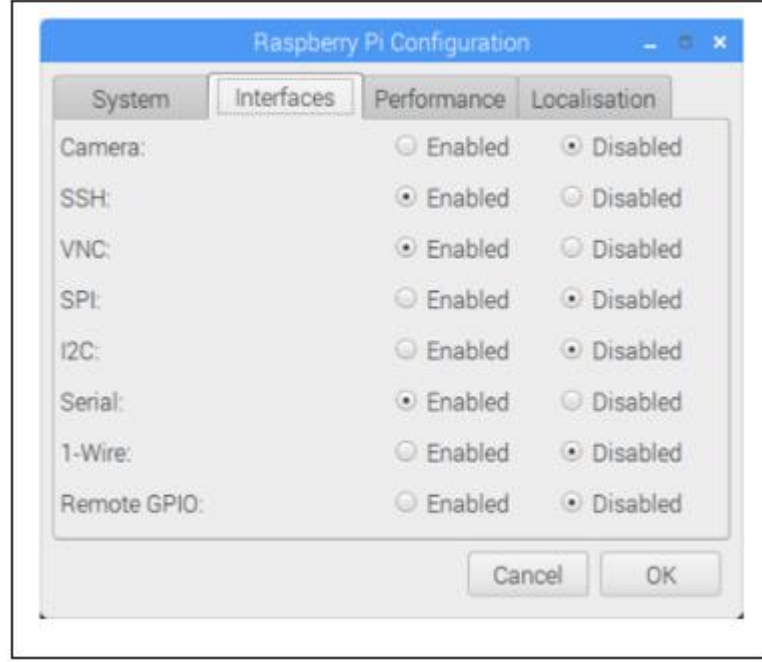
4.1. RPİ Kurulumu

Raspberry Pi kendi işletim sistemi olan bir mikrobilgisayardır. Bu yüzden ilk olarak gerekli sürücü dosyaları indirilmelidir. Şekil 4.1. de görüldüğü <https://www.raspberrypi.org/software/> üzerinden istenilen sürücü indirilir. Bu indirilen dosyalar Micro SD karta yüklenmesi için Win32DiskImager dosyası yüklenmesi gerekmektedir. Bu program ile indirilen imaj dosyası SD karta yazdırılarak işlem tamamlanır.



Şekil 4.1: Raspberry pi os ve çeşitleri.

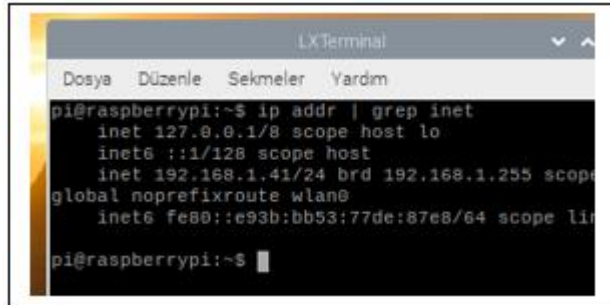
İlk açılışta ayarlar yapabilmek için hdml kablosu yardımı ile bir ekrana bağlantı yapılarak, mikrobilgisayarın açılışı görülebilir. Klavye ile fare takıldıktan sonra ayarlama işlemine hazır hale getirilir. İlk olarak Şekil 4.2. deki gibi konfigürasyon ara yüzüne girilir. Birden fazla uzaktan bağlantı seçeneği mevcuttur. Bu çalışma da uzaktan bağlantı ayarı VNC Viewer programı yardımıyla yapılacağı için VNC seçeneği aktif edilir



Şekil 4.2: Raspberry pi konfigürasyon ara yüzü.

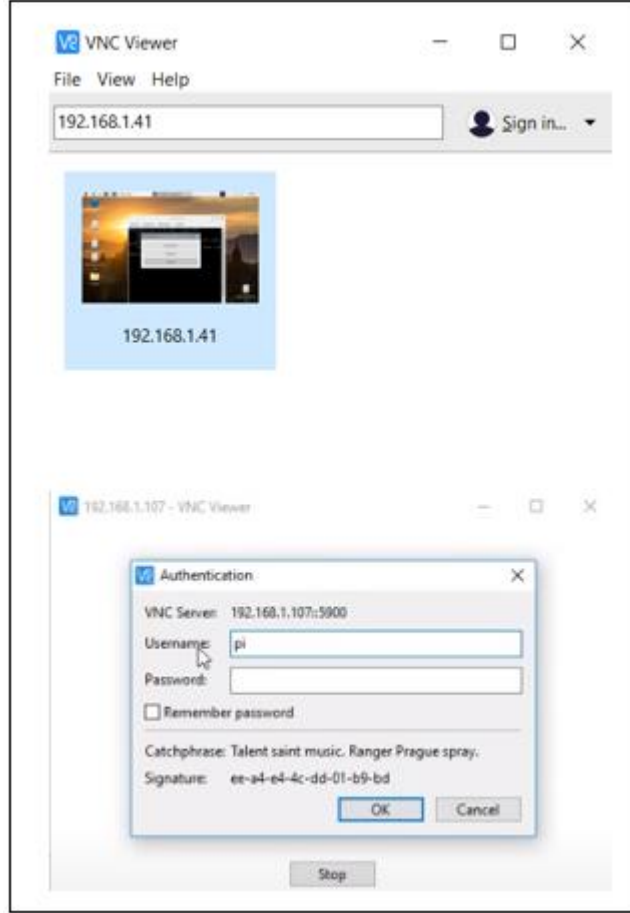
Raspberry Pi'ya uzaktan bağlanabilmek için VNC viwer programı <https://www.realvnc.com/download/vnc/> adresinden indirilir.

VNC programı ile mikrobilgisayara bağlanmak için kullanılacak ip adresi komut satırına `ip addr | grep inet` yazılarak görülmektedir.



Şekil 4.3: RPi ait ip adresi.

IP adresi öğrenildikten sonra Şekil 4.4. deki arama yerine IP adresi yazıldıktan sonra giriş tuşuna basılır. Program kullanıcıdan kullanıcı adı ve şifre isteyecektir. Eğer başlangıç şifre bilgileri değiştirilmediyse;



Şekil 4.4: VNC ait ara yüz ekranı.

- Kullanıcı adı: pi
- Şifre: raspberry

şeklinde yazılarak RPi'ya uzaktan bağlanma başarılı şekilde yapılır.

4.2. Python ve Kütüphane Seçimi

Bu bölümde tasarlanan aracın uyum içerisinde çalışabilmesi için geliştirilecek yazılım da kullanılacak olan yazılım dili ve derin öğrenme kütüphanelerinden bahsedilecektir.

4.2.1. Yazılım Dili Seçimi

Python, nesne yönelimli, modüler ve etkileşimli yüksek seviyeli bir programlama dilidir. Python programlama dili yaratıcısı Hollandalı Gudio Van Rossum tarafından 1990 yılında geliştirilmeye başlanmıştır [27].

Python programlama dili, RPİ içerisinde kolayca yüklenip kullanılabilir. Aynı zamanda derin öğrenme uygulamalarında en çok kütüphane ve uygulama geliştirilen dildir. Bunun dışında python programlama bilimsel çalışmalarda sıkça kullanılmakla birlikte Arduino ile de uyumlu şekilde çalışmaktadır.

Python ile model geliştirmek için gerekli araçlar ve kütüphaneler komut satırından kolayca yüklenebilmektedir. Bir sonraki adımda kütüphaneler ile ilgili bilgiler verilecektir.

4.2.2. Derin Öğrenme Kütüphaneleri

Kütüphane kurulumu hem ana bilgisayarda hem de raspberry pi içerisinde yapılacaktır. Bunun sebebi RPİ işlemci gücü modeli eğitmek için zayıf kaldığından, modeli Windows ortamında eğitmek gereklidir. Bu yüzden ilk olarak kütüphane kurulumu bilgisayara yapılır. Windows ortamında eğitilen bu modeli pi içerisinde çalıştırabilmek içinde aynı kütüphanelerin raspberry pi içerisinde de yüklenmesine ihtiyaç duyulmaktadır. Aşağıdaki tablo 4.1 ve tablo 4.2 hem bilgisayarda hem de RPİ içerisine indirilmiş olan yazılım araçları ve kütüphaneler gösterilmiştir.

Bütün bu kütüphanelerin sorunsuz yüklenmesi ve birlikte çalışması için sanal ortam yaratılmalıdır. Sanal ortam yaratılmadan önce klasör oluşturulur ve bunun içerisine girilir. Komut satırından aşağıdaki kodlar yazılarak sanal ortam yaratılmış olur.

- Windows için: `python -m venv env`
- Linux için: `python3 -m venv env`

Bu işlemten sonra sanal ortamı aktif etmek gerekir. Aktif etme işlemi aşağıdaki gibi yapıldıktan sonra sanal ortam kütüphane yüklenmesine hazır hale gelir.

Tablo 4.1: Yazılım ve Donanım Araçları Bilgisi.

İşletim Sistemi	Windows	Windows 10
İşletim Sistemi	Ubuntu	18.04 LTS
Grafik Kart	GeForce	GTX 1650 Ti
Yazılım Dili	Python	3.7.3
IDE	Pycharm Google Colab	2019.3.5

Tablo 4.2: Kullanılan Ana Kütüphaneler.

Kütüphane	Versiyon
OpenCV	3.4.6
Tensorflow	2.3.0
Keras	2.4.3
Matplotlib	3.3.3
Numpy	1.18.5

4.3. Veri Seti Oluşturulması

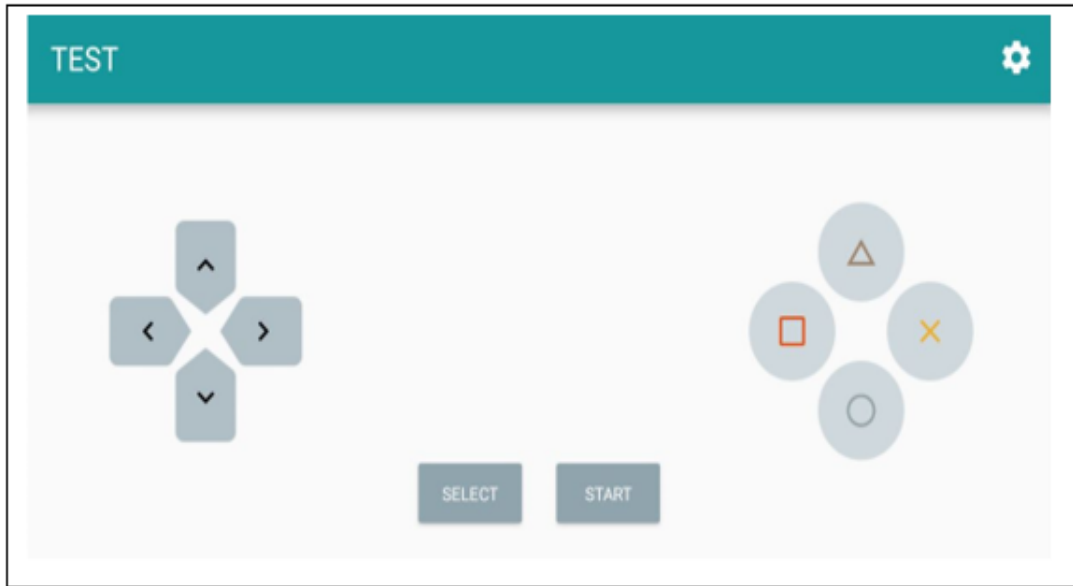
Gerçek zamanlı nesne tanıma ve sınıflandırma uygulamalarında model geliştirilirken en önemli konulardan biri de veri setinin istenilen uygulama için gerçekçi olmalıdır. Hazır veri seti kullanılarak eğitilen modeller aynı performansı canlı uygulamalarda bazen gösterememektedir. Bunun sebebi kullanılan veri setinin test anındaki veri seti ile benzerlik göstermemesidir. Bu durum başarısız deneyler ve zaman kaybetmesine neden olmaktadır.

4.3.1 Veri Seti Toplanması

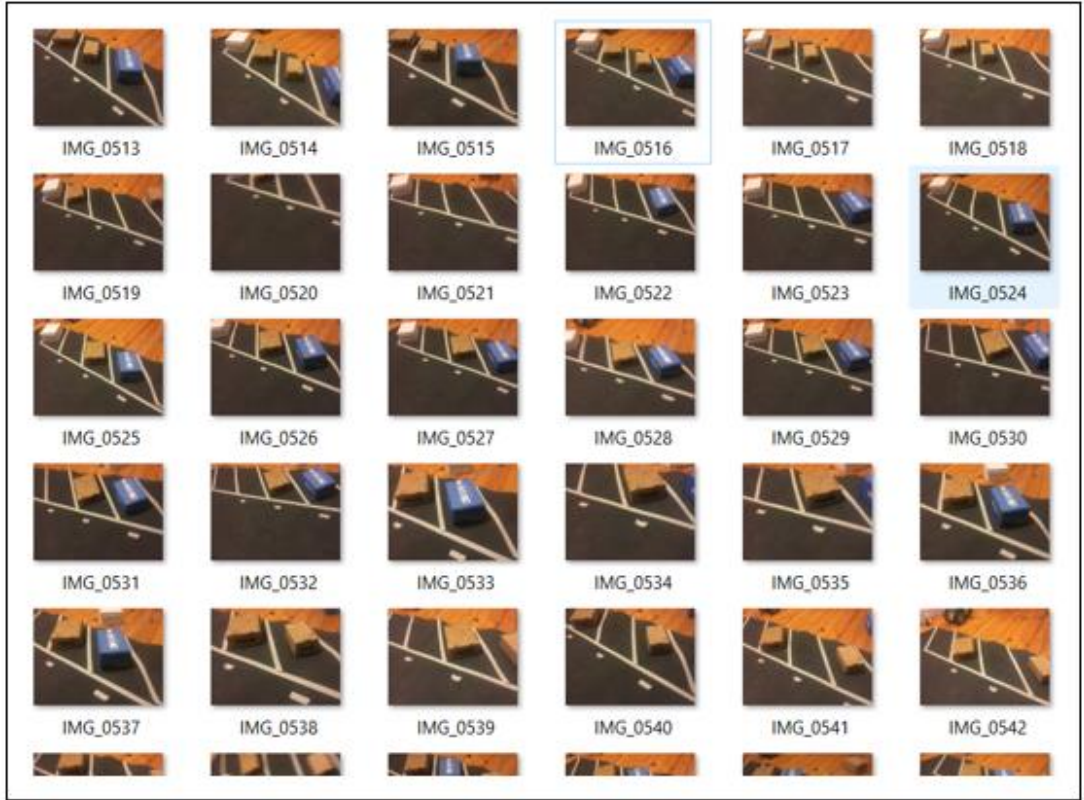
Bu çalışmada bu problemi çözmek adına parkur kurulmuş olup, uzman kullanıcı tarafından araca park işlemi yapılırken görüntüler kaydedilmiştir. Bu görüntülerden işe yarar olan 866 adet görüntü eğitim verisi olarak alınmıştır. Bu görüntülerin birçoğundan birden fazla etiketleme işlemi ile toplamda 2000’den fazla veri etiketlendirilmesi yapılmıştır. Buna ek olarak 2 farklı kameradan toplanmış 12 binden

fazla görüntüyü içeren hazır CNRPark veri seti kullanılmıştır [28]. CNR veri setinden 4 bin görüntü alınarak toplamda 5 binin üzerinde etiketli veri seti ile modelin eğitilmesine olanak sağlanmıştır. Bu verilerin %75'i eğitim verisi olarak %25 i ise test verisi olarak kullanılmak üzere bölünmüştür. Görüntülerden oluşan bu veri seti iki sınıflı bir yapıya sahip olup etiket işlemleri bos ve dolu şeklinde ayarlanmıştır.

Uzman kullanıcı raspberry pi ya bağlı olan arduino üzerindeki bluetooth sensörüne Bölüm 2.2 de bahsedildiği gibi konfigürasyon ayarları yaparak telefon aracılığıyla haberleşme sağlamaktadır. Bağlantı kurulduktan sonra Şekil 4.5' deki gibi bir kullanıcı ara yüzü gelecektir. Bu ara yüz ile aracı kontrol ederek yazılan yazılım ile veri toplama işlemi gerçekleştirilir. Şekil 4.6 da uzman kullanıcı tarafından farklı açılardan toplanan verilerin bir kısmı görülmektedir.



Şekil 4.5: Telefon ile veri toplamak için gerekli kullanıcı ara yüzü.

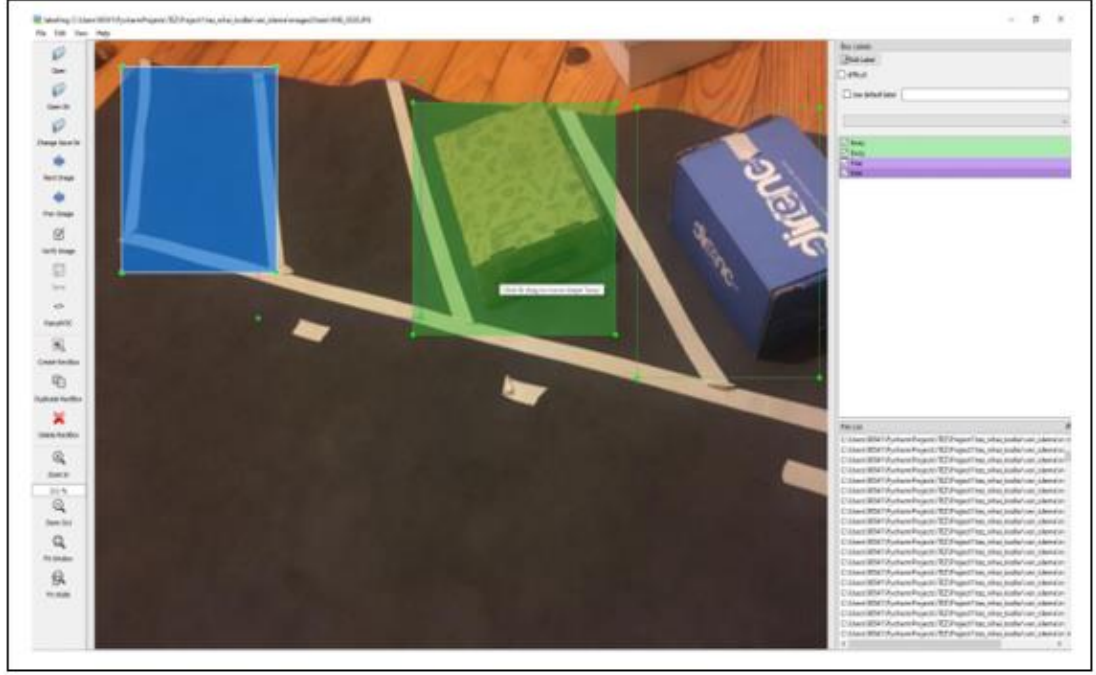


Şekil 4.6: Uzman kullanıcı tarafından toplanan görüntülerin bir kısmı.

4.3.2 Veri Seti Etiketlenmesi

Başarılı bir model eğitimi için veri seti toplamak ne kadar önemli ise veri setinin doğru ve düzgün şekilde etiketlenmesi bir o kadar önem arz etmektedir. Bir görüntü de birden çok etiketlenecek verilerde olabilmektedir. Bu yüzden ne kadar farklı ortamlarda ne kadar çeşitli etiketlenmiş veri seti elde edilirse modelin başarısı sansı o kadar yükselmektedir.

Veri seti etiketleme işlemi bu çalışmada labelImg isimli python uygulamasına komut satırından ulaşarak yapılmıştır. Program çalıştırıldıktan sonra Resim 4.7'deki gibi bir ara yüz karşımıza çıkmaktadır. Burada yapılacak işlem ilk önce programda ham verilerin olduğu klasörü programda göstermektir. Daha sonraki işlem ise etiketlenmiş verinin olacağı klasörü göstermektir. Bu arada önemli nokta ise etiketlenmiş verinin ham veri ile aynı klasörde olma zorunluluğudur. Aksi halde verilerin etiketlenmesinin bir önemi kalmayacaktır. Yine Şekil 4.7 de görüldüğü üzere klasörleri belirlenmiş verilerden örnek olarak etiketlenmiş bir görüntü gösterilmektedir.



Şekil 4.7: Etiketlenmiş bir örnek görüntü.

Şekil 4.7 de tek bir görüntüde var olan nesneler park yerindeki durumlarına göre boş ya da dolu olarak 4 farklı şekilde etiketleme işlemi yapılmıştır. Bunun gibi 866 toplanan veri seti de bu şekilde etiketlenme işlemlerinden geçerek 2000’den fazla etiketlenmiş veri oluşturulmuştur.

4.4. Evrişimli Sinir Ağı Model Eğitimi

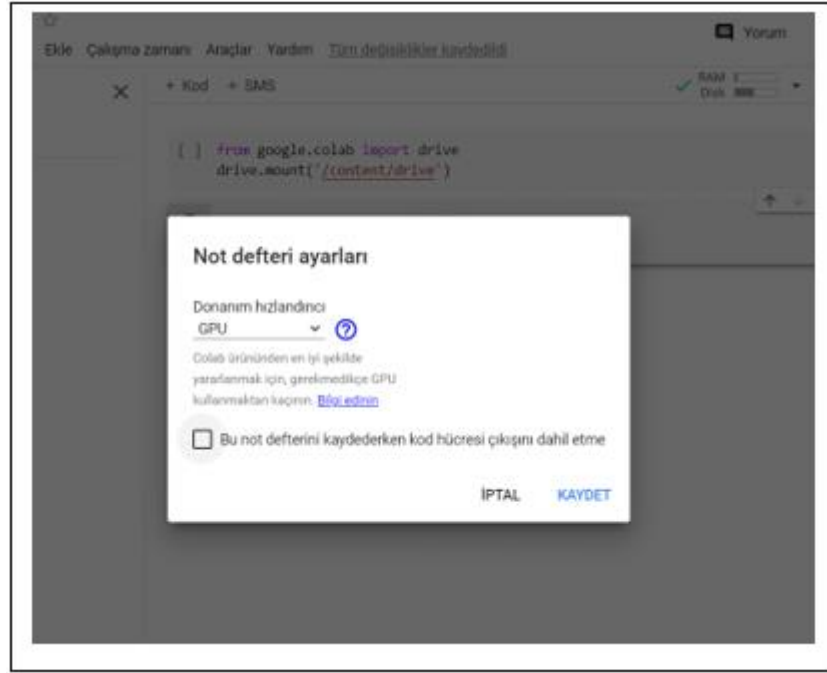
Bu bölümde modeli daha hızlı eğitiminin Google colab'daki gpu desteği ile aynı zamanda bilgisayar üzerinden nasıl yapıldığı anlatılacaktır. Devamında ise modelin eğitimi hakkında bilgiler gösterilecektir.

4.4.1. Google Colab Üzerinde Model Eğitimi

Google colab’a <https://colab.research.google.com/> sitesinden girilerek yeni proje dosyası açılır. Bu dosyaya aşağıda gösterilen kod parçası girilir. Bu kod parçası ile colab, google drive ile bağlanmış olur. Bu sayede drive da bulunan veri setine kolayca ulaşarak veri çekme işlemi yapılır.

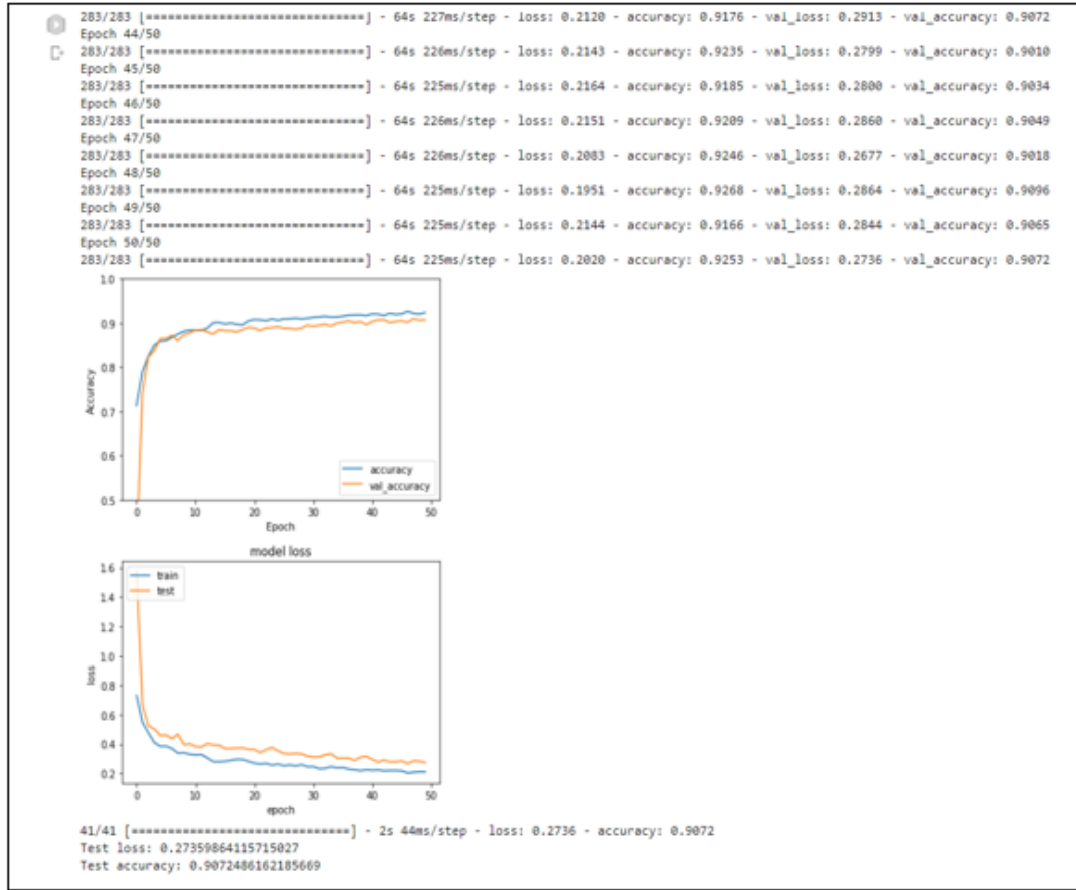
- from google.colab import drive
- drive.mount('/content/drive')

Şekil 4.8 de gösterildiği üzere çalışma zamanı üzerinden çalışma zamanı değiştir bölümüne girildikten sonra ekranda not defteri ayarları çıkmaktadır. Buradan donanım hızlandırıcı GPU seçilerek model eğitim işlemleri daha hızlı yapılacaktır.



Şekil 4.8: Colab çalışma zamanı değiştirme işlemi.

Şekil 4.9 da google colabta ücretsiz gpu üzerinde eğitilmiş bir modelin performansı görülmektedir. Oldukça başarılı performansın yanı sıra 50 epokluk bir eğitim işlemini bir iki dk içerisinde yapmıştır.



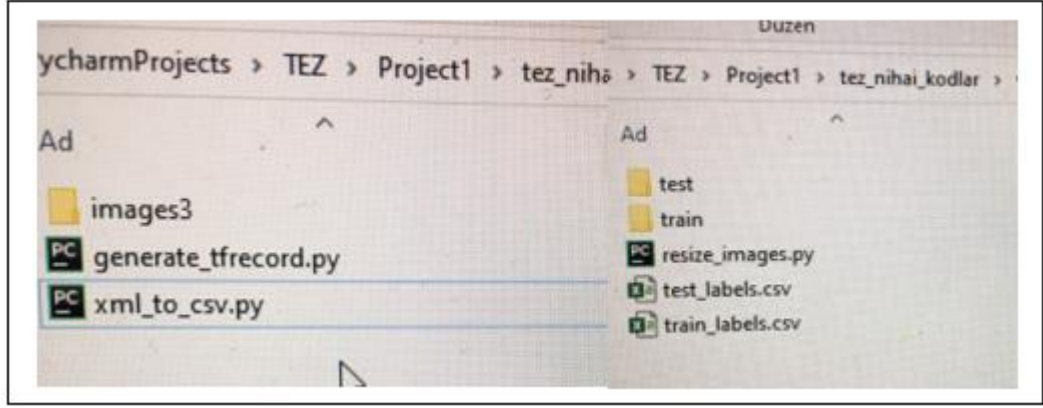
Şekil 4.9: Colab da gpu üzerinden eğitilmiş bir modelin görüntüsü.

4.4.2. Windows CPU Üzerinde Model Eğitimi

Bölüm 4.3.1 de anlatılan veri etiketleme işlemi yapıldıktan sonra veriler eğitim ve test verileri olmak üzere iki ayrı klasörlerin içine aktarılır. Bu etiketlenmiş verilerin .xml uzantılı dosyalardır. Bu uzantılı dosyalar terminalden

– python xml_to_csv.py

şeklinde çalıştırılarak csv dosyaları üretilir. Şekil 4.10 da xml den dosyalarından hem eğitim seti için hem de test verileri için üretilmiş olan csv dosyaları görülmektedir.



Şekil 4.10: Csv uzantılı veri seti oluşturulması.

Oluşturulan csv uzantılı dosyalar record uzantılı dosyaları çevrilerek veriler eğitime girmek için hazır hale gelecektir. Bu işlem de aşağıdaki şekilde yapılabilir.

- python generate_tfrecord.py --csv_input=images3\train_labels.csv --image_dir=images3\train --output_path=train.record
- python generate_tfrecord.py --csv_input=images3\test_labels.csv --image_dir=images3\test --output_path=test.record

Record dosyaları hem eğitim verisi için hem de test verisi için ayrı ayrı yapılmıştır. Bu işlem sonrasında geriye kalan son adım sınıf etiketi dosyası oluşturmaktır. Bu çalışmada iki sınıflı veriler olduğu için dosyanın içine 1. Numaralı id ye boş, 2 numaralı id'ye dolu ataması yapılmıştır. Böylece eğitim sonucunda eğitilmiş bir modele girdi verildiğinde 1 ya da 2 diye çıktı alınacaktır. Bu sayede modeldeki karmaşıklığın önüne geçilmiş olunmuştur. Bu elde edilen veriler ile oluşturulan dosyaların dizini park.config isimli dosyanın içerisine gömülür. Bu dosyanın içerisinden veriler okunarak model eğitim yapılmaktadır. Windows üzerinde model eğitimi aşağıdaki gibi yapılmaktadır. Şekil 4.11 da modelin eğitim sırasında alınmış bir görüntüsü gösterilmiştir.

- python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/park.config

Not: Toplanan görüntü verilerinin çözünürlüğü yüksek olduğu için eğitim süresi çok uzun olacaktır ya da bellek boyutu aşılacağından eğitim başlamayacaktır. Bu sorunu çözmek için verileri etiketleme işlemi yapılmadan önce çözünürlüğü yüksek olan

resimler geliştirilmiş olan `resize_images.py` programı ile resimlerin boyutu küçültülmüştür. Boyutu azaltılmış veriler etiketleme aşamasına geçmiştir.

```
10117 01:12:00.147484 7160 learning.py:512] global step 12353: loss = 2.5795 (14.017 sec/step)
INFO:tensorflow:Recording summary at step 12353.
10117 01:12:08.029699 7376 supervisor.py:1050] Recording summary at step 12353.
INFO:tensorflow:global step 12354: loss = 2.3004 (10.031 sec/step)
10117 01:12:16.202221 7160 learning.py:512] global step 12354: loss = 2.3004 (10.031 sec/step)
INFO:tensorflow:global step 12355: loss = 2.0747 (10.480 sec/step)
10117 01:12:26.693504 7160 learning.py:512] global step 12355: loss = 2.0747 (10.480 sec/step)
INFO:tensorflow:global step 12356: loss = 2.2779 (9.603 sec/step)
10117 01:12:36.299515 7160 learning.py:512] global step 12356: loss = 2.2779 (9.603 sec/step)
INFO:tensorflow:global step 12357: loss = 2.0657 (8.566 sec/step)
10117 01:12:44.868583 7160 learning.py:512] global step 12357: loss = 2.0657 (8.566 sec/step)
INFO:tensorflow:global step 12358: loss = 2.6923 (9.412 sec/step)
10117 01:12:54.282631 7160 learning.py:512] global step 12358: loss = 2.6923 (9.412 sec/step)
INFO:tensorflow:global step 12359: loss = 2.1829 (9.820 sec/step)
10117 01:13:04.105989 7160 learning.py:512] global step 12359: loss = 2.1829 (9.820 sec/step)
INFO:tensorflow:global step 12360: loss = 2.2454 (9.112 sec/step)
10117 01:13:13.220225 7160 learning.py:512] global step 12360: loss = 2.2454 (9.112 sec/step)
INFO:tensorflow:global step 12361: loss = 1.8896 (9.223 sec/step)
10117 01:13:22.444658 7160 learning.py:512] global step 12361: loss = 1.8896 (9.223 sec/step)
INFO:tensorflow:global step 12362: loss = 2.6502 (9.242 sec/step)
10117 01:13:31.778516 7160 learning.py:512] global step 12362: loss = 2.6502 (9.242 sec/step)
INFO:tensorflow:global step 12363: loss = 2.2262 (8.287 sec/step)
10117 01:13:40.067251 7160 learning.py:512] global step 12363: loss = 2.2262 (8.287 sec/step)
INFO:tensorflow:global step 12364: loss = 2.3337 (8.404 sec/step)
10117 01:13:48.472230 7160 learning.py:512] global step 12364: loss = 2.3337 (8.404 sec/step)
INFO:tensorflow:global step 12365: loss = 2.2056 (8.269 sec/step)
10117 01:13:56.745311 7160 learning.py:512] global step 12365: loss = 2.2056 (8.269 sec/step)
INFO:tensorflow:Recording summary at step 12365.
10117 01:14:06.846301 7376 supervisor.py:1050] Recording summary at step 12365.
INFO:tensorflow:global step 12366: loss = 2.0520 (14.334 sec/step)
10117 01:14:11.089928 7160 learning.py:512] global step 12366: loss = 2.0520 (14.334 sec/step)
INFO:tensorflow:global step 12367: loss = 2.4119 (9.694 sec/step)
10117 01:14:20.783500 7160 learning.py:512] global step 12367: loss = 2.4119 (9.694 sec/step)
INFO:tensorflow:global step 12368: loss = 2.8770 (9.079 sec/step)
10117 01:14:29.864544 7160 learning.py:512] global step 12368: loss = 2.8770 (9.079 sec/step)
INFO:tensorflow:global step 12369: loss = 2.1911 (9.360 sec/step)
10117 01:14:39.226166 7160 learning.py:512] global step 12369: loss = 2.1911 (9.360 sec/step)
INFO:tensorflow:global step 12370: loss = 2.4839 (10.040 sec/step)
10117 01:14:49.269618 7160 learning.py:512] global step 12370: loss = 2.4839 (10.040 sec/step)
INFO:tensorflow:Recording summary at step 12370.
```

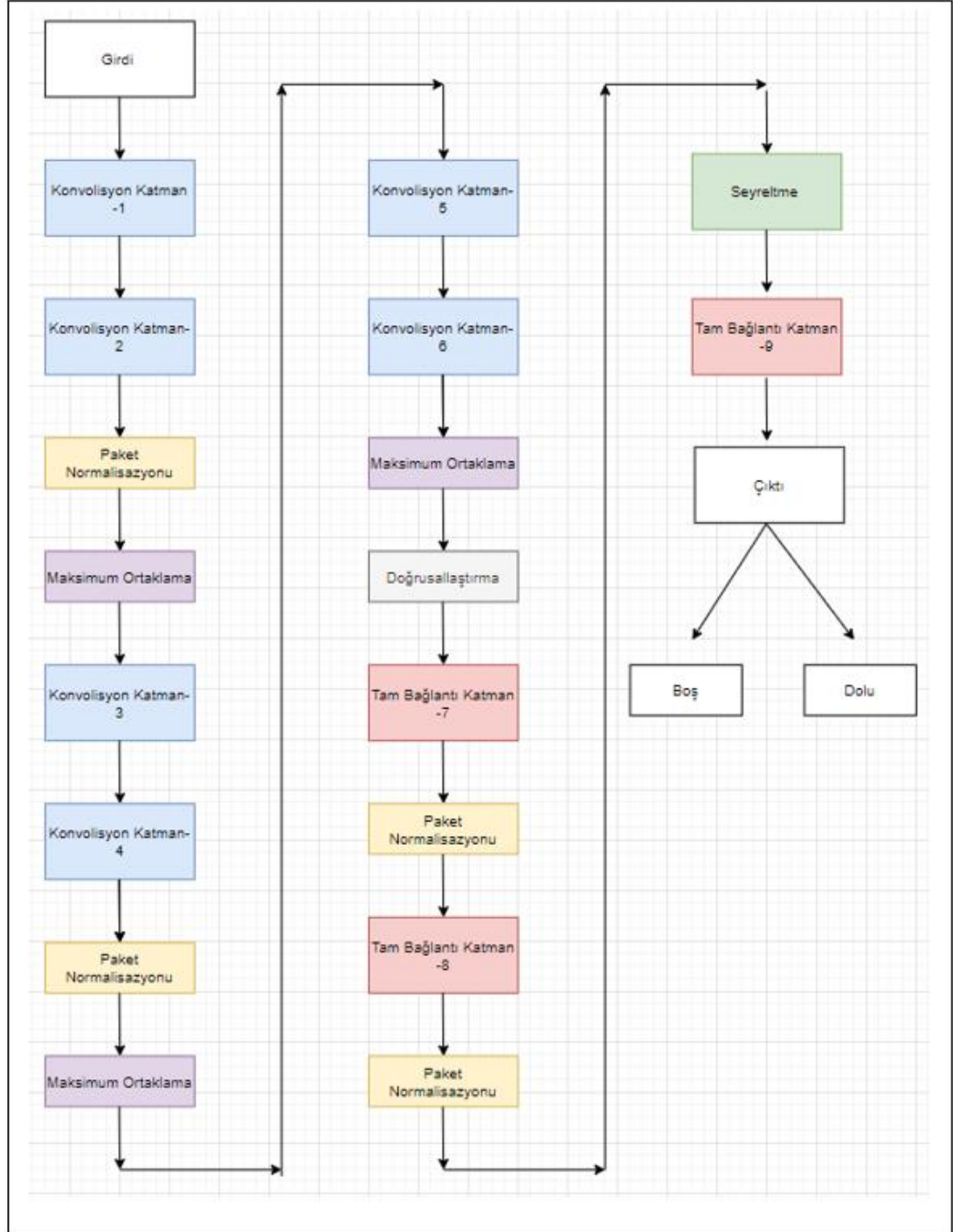
Şekil 4.11: Windows 10 cpu üzerinde eğitilmekte olan bir modelin görüntüsü.

4.5. Önerilen Sinir Ağı Modelinin Mimarisi ve Performansı

Yukarıdaki bölümlerde anlatılan ve derin öğrenme uygulamalarında oldukça iyi performans gösteren modeller açıklanmıştır. Ancak bu modeller çok fazla katmanlı ve hesap gücü çok yüksek olması sebebiyle daha güçlü işlemci gücüne ihtiyaç duymaktadır. Bu durumda oldukça yüksek maliyet getirmektedir. Bu nedenle daha düşük işlemci gücü ile çalışabilen diğer modellerden daha az katmanlı olarak tasarlanan Acinet isimli derin öğrenme modeli önerilmiştir. Bu bölümde önerilen modelin performansı incelenecektir.

4.5.1. Önerilen Modelin Mimarisi

Şekil 4.1 de önerilen ağ mimarisinin yapısı gösterilmektedir. İlk bakıldığında 8 katmanlı bir mimari olduğu gözükmemektedir. Bunun sebebi başvurulacak uygulamada sınıflandırma işlemi sadece 2 sınıflı olduğu için gereksiz işlem gücünü tüketmemektir.



Şekil 4.12: Önerilen ağ mimarisi.

Şekil 4.12 de önerilen ağ mimarisinin yapısı gösterilmektedir. İlk bakıldığında 8 katmanlı bir mimari olduğu gözükmemektedir. Bunun sebebi başvurulacak uygulamada sınıflandırma işlemi sadece 2 sınıflı olduğu için gereksiz işlem gücünü tüketmemektir. Önerilen ağ mimarisinde girdiler bellekte fazla yer kaplamaması için 32x32 boyutlarına indirgenmiştir. Veriler eğitim girdiklerinde 32 gruplu paketler halinde eğitim işlemine tabi tutulmaktadır. Evrişim filtre boyutu 3x3 seçilmiş olup evrişim katmanında 32 ile 64, tam bağlantı katmanında ise 1024 çıkış katmanında ise 2 sınıflı bir çıktı üretileceğinden dolayı son da 2 seçilerek katman değerleri tanımlanmıştır.

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	320
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization (Batch Normalization)	(None, 32, 32, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_1 (Batch Normalization)	(None, 16, 16, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_4 (Conv2D)	(None, 8, 8, 64)	36928
conv2d_5 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 64)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 1024)	1049600
batch_normalization_2 (Batch Normalization)	(None, 1024)	4096
dense_1 (Dense)	(None, 1024)	1049600
batch_normalization_3 (Batch Normalization)	(None, 1024)	4096
dropout (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 3)	3075
Total params: 2,249,699		
Trainable params: 2,245,411		
Non-trainable params: 4,288		

Şekil 4.13: Önerilen modelin tüm yapısı.

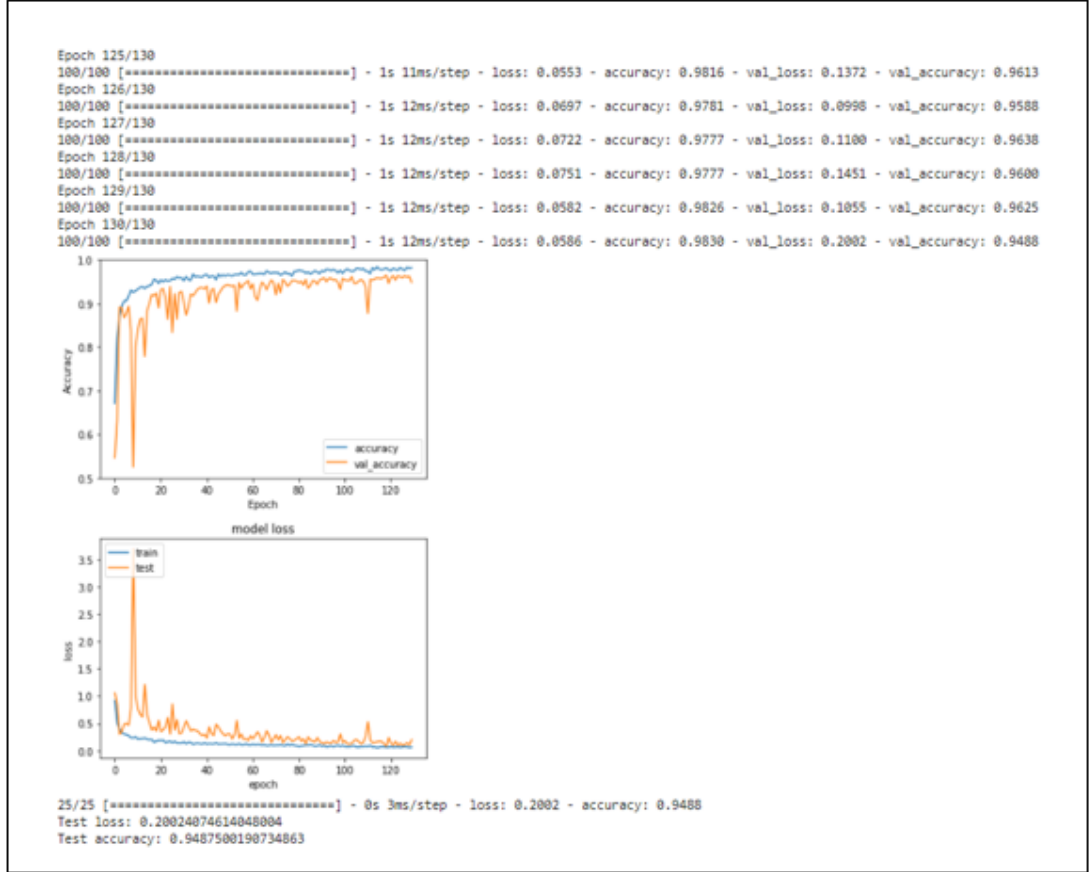
Bu model yapısında diğer modellerden farklı olarak derin öğrenme işlemlerinde çok kullanılan seyreltme işlemi kullanılmıştır. Bunun sebebi modelin aşırı ezberlemesine gerektirecek kadar veri seti bulunmamasıdır. Seyreltme işlemi sadece son yapılarak modelin az öğrenmesinin de önüne geçilmiştir. Bu uygulama çerçevesinde her evrişim katmanından sonra seyreltme yapmak yerine düzenleme yapmanın deneysel olarak daha iyi olduğu tespit edilmiştir. Bu yüzden ilk iki evrişim katmanından sonra ve son iki evrişim katmanlarından önce paket normalizasyonu yapılarak en iyi başarıyı göstermesi amaçlanmıştır.

AcıNet modelinde kullanılan maliyet fonksiyonu olarak derin öğrenme uygulamalarında kullanılan cross entropy fonksiyonu kullanılmıştır. Optimizasyon fonksiyonu seçimi içinde, veriler modelin içine 32’li mini paketler halinde gireceği için SGD optimizasyonu bu iş için en uygun olandır. Hatanın lokal minimuma noktasına ulaşabilmesi için öğrenme katsayısı genel olarak 0.001, 0.0001, 0.00001 gibi değerler alınır. Bu çalışmada en öğrenim yapan model için 0.0001 seçilmiştir. Modelin eğitim süresi 130 epok işleminden oluşmaktadır. Şekil 4.13 de AcıNet modelinin tüm yapısı gösterilmektedir.

AcıNet modelinin VGG-16, AlexNet gibi modellerden en önemli farklarından biri de katman sayısının ve toplam kullanılan parametrelerin daha düşük olmasıdır. Bu sayede işlem gücü düşük tutularak gerçek zamanlı uygulamalarda daha az maliyetli işlemler yaptırılabilir.

4.5.2. Önerilen Modelin CNR Park Veri Seti Üzerindeki Performansı

Bu bölümde ilk olarak hazır veri seti olan 4000 adet CNR veri setinde modelin performansı gözlenecektir. Daha sonra da bütün veri seti üzerinde modelin performansı incelenecektir.



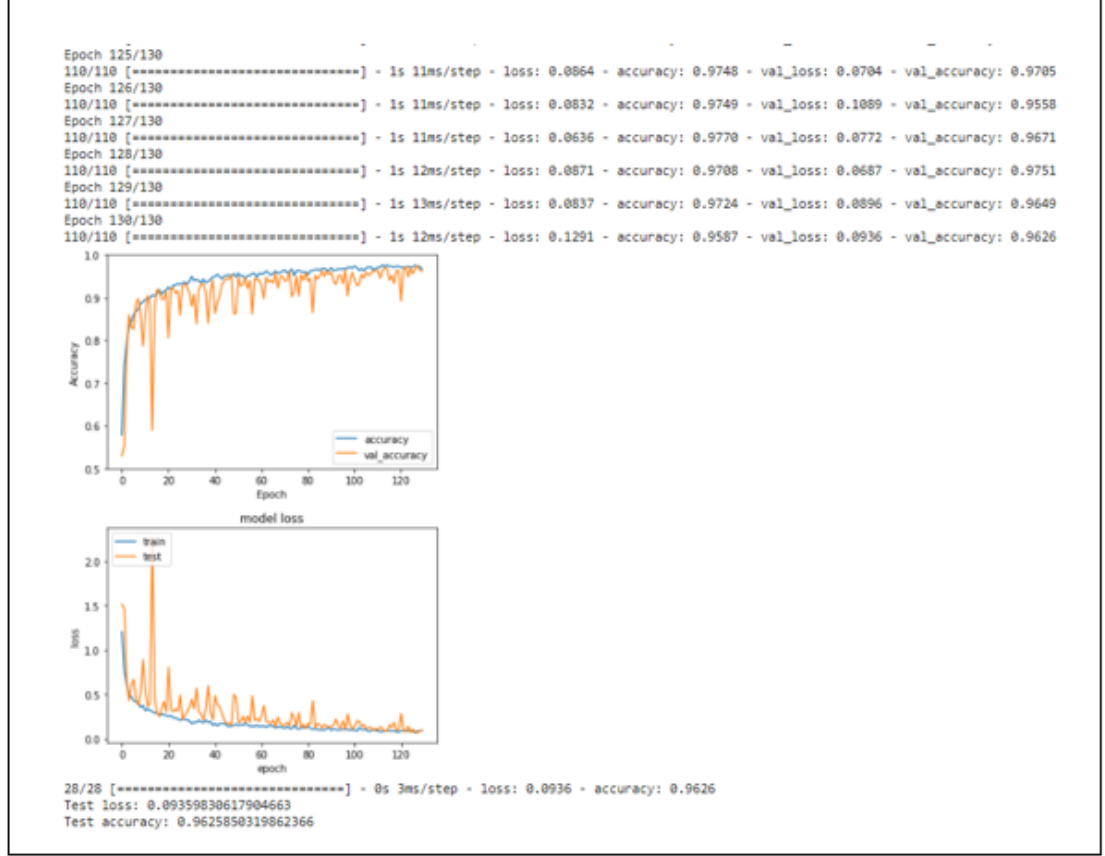
Şekil 4.14: Acinet'in CNR veri seti üzerindeki performansı.

Tablo 4.3: Acinet CNR üzerindeki toplam performans sonuçları

Epok	130
Eğitim Başarısı	0.98
Eğitim Kaybı	0.05
Test Başarısı	0.94
Test Kaybı	0.20

4.5.3. Önerilen Modelin Bütün Veri Seti Üzerindeki Performansı

Bu bölümde toplanan 4866 adet veri seti üzerinde modelin performansı gözlenecektir.



Şekil 4.15: AcinetNet'in bütün veri seti üzerindeki performansı.

Tablo 4.4: AcinetNet bütün verilerin üzerindeki toplam performans sonuçları.

Epok	130
Eğitim Başarısı	0.96
Eğitim Kaybı	0.12
Test Başarısı	0.96
Test Kaybı	0.09

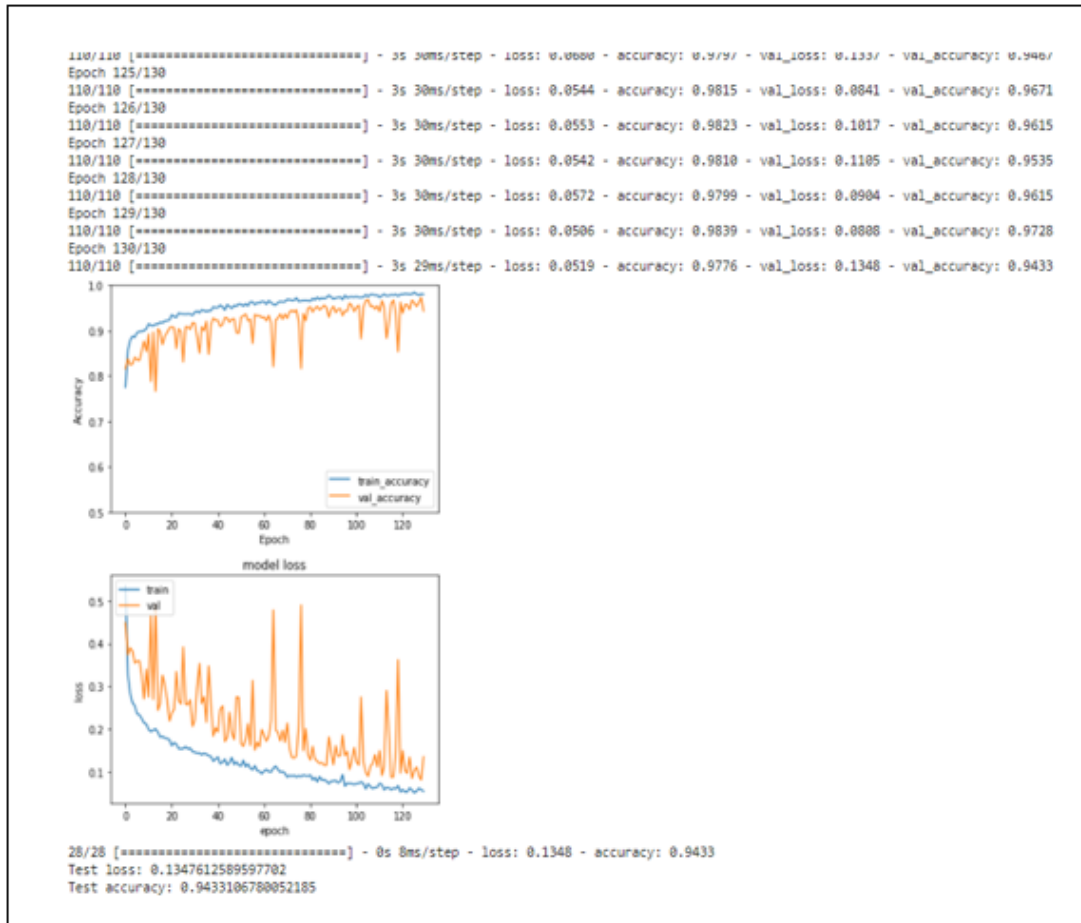
Tablodaki sonuçları bakıldığında hazır Acinet modelinin bütün veri seti üzerindeki performansı hazır veri setindeki performansa göre %2 oranında daha iyi başarı göstermiştir. Aynı zamanda %1'lik kayıpta da azalma olmuştur. Önerilen model

de ki bu fark manuel olarak toplanan veri setinin hazır verilere eklenmesi ile çeşitlik kazandırmasından dolayıdır.

4.6. AcıNet Modeli ile Diğer Modellerin Karşılaştırılması

Bu bölümde, daha önceki bölümlerde anlatılan derin öğrenme modellerinin ve bu çalışmada önerilen model ile karşılaştırması yapılacaktır. Bu karşılaştırmayı yapabilmek için bütün modeller aynı parametreler ile eğitilecektir.

VGG-11:

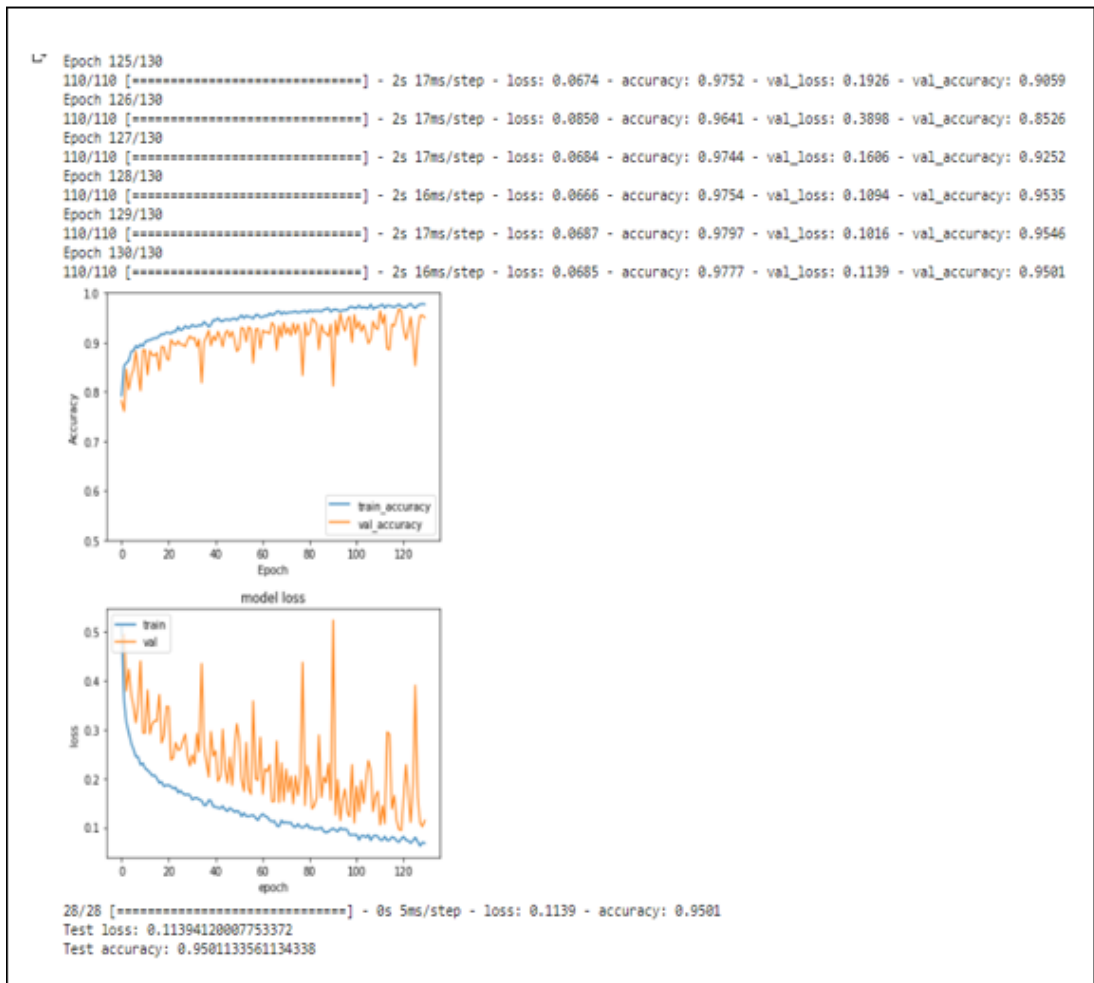


Şekil 4.16: VGG-11 modelinin bütün veri seti üzerindeki performansı.

Tablo 4.5: VGG-11 bütün veriler üzerindeki toplam performans sonuçları.

Epok	130
Eğitim Başarısı	0.97
Eğitim Kaybı	0.05
Test Başarısı	0.94
Test Kaybı	0.13

AlexNet:

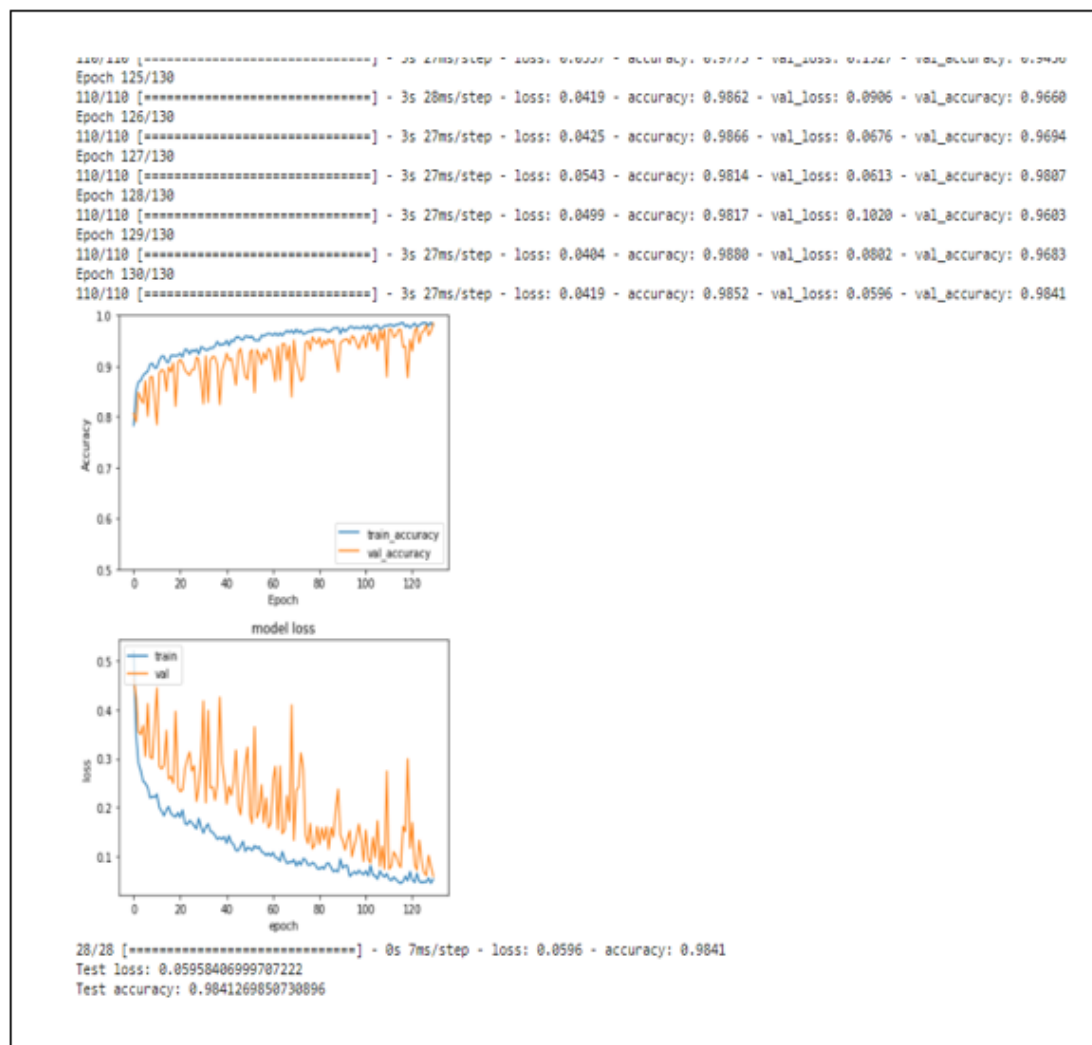


Şekil 4.17: AlexNet modelinin bütün veri seti üzerindeki performansı.

Tablo 4.6: AlexNet modelinin bütün veriler üzerindeki toplam performans sonuçları.

Epok	130
Eğitim Başarısı	0.97
Eğitim Kaybı	0.06
Test Başarısı	0.95
Test Kaybı	0.11

VGG-13:

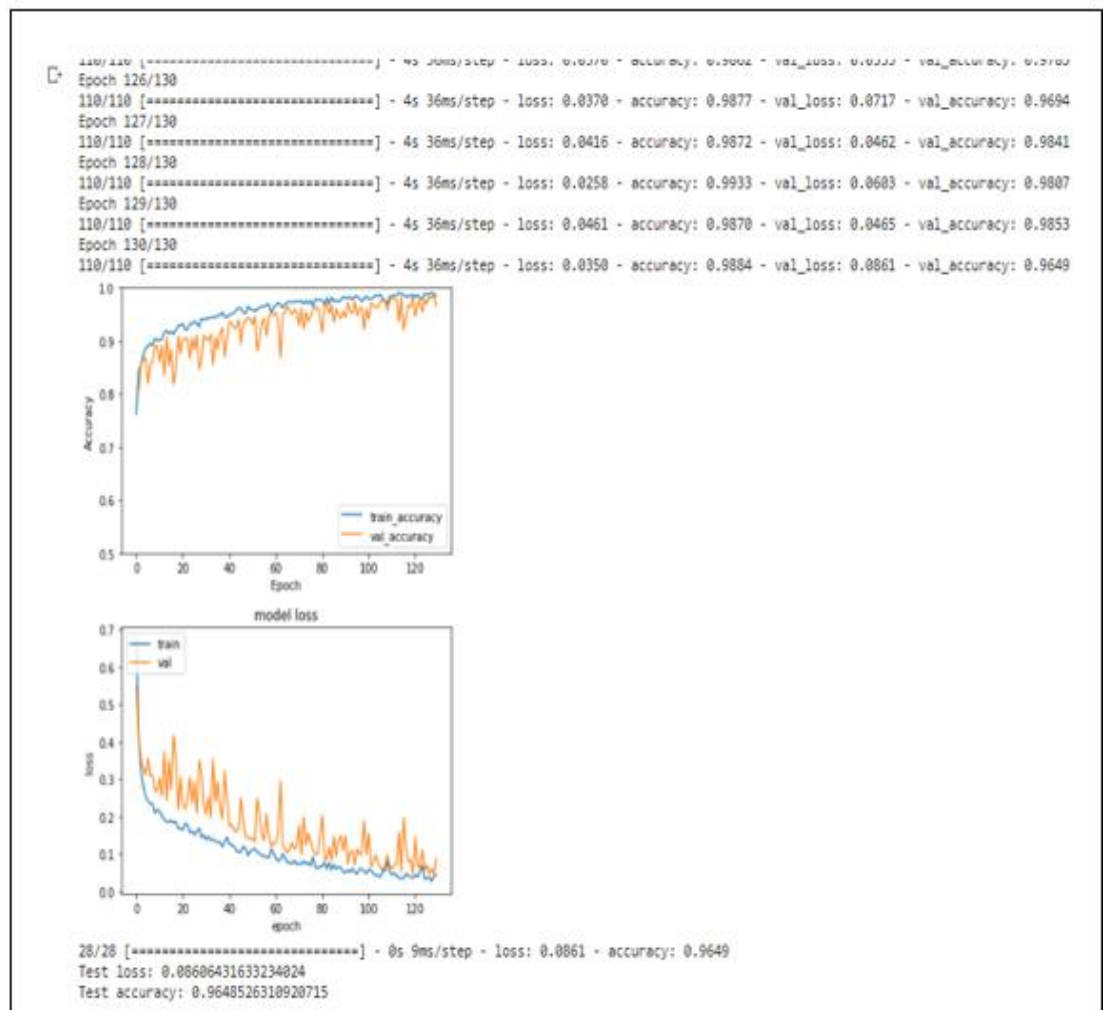


Şekil 4.18: VGG-13 modelinin bütün veri seti üzerindeki performansı.

Tablo 4.7: VGG-13 modelinin bütün veriler üzerindeki toplam performans sonuçları.

Epok	130
Eğitim Başarısı	0.98
Eğitim Kaybı	0.04
Test Başarısı	0.98
Test Kaybı	0.06

VGG-16:



Şekil 4.19: VGG -16 modelinin bütün veri seti üzerindeki performansı.

Tablo 4.8: VGG-16 modelinin bütün veriler üzerindeki toplam performans sonuçları.

Epok	130
Eğitim Başarısı	0.98
Eğitim Kaybı	0.03
Test Başarısı	0.97
Test Kaybı	0.08

Tablo 4.9: Bütün Mimarilerin Karşılaştırılması.

Mimariler	Katman Sayısı	Parametre Sayısı	Basarı	Kayıp
AcıNet	8	2.249.699	0.962	0.093
VGG-11	11	10.582.675	0.943	0.134
AlexNet	8	25.657.955	0.950	0.113
VGG-13	13	32.386.403	0.984	0.060
VGG-16	16	37.696.099	0.964	0.086

5 farklı mimariye sahip derin sinir ağları, aynı sayısal değerler kullanılarak kendine has katmanlar ile birlikte aynı veri seti üzerinde, aynı epok sayısında, aynı kayıp fonksiyonları ile aynı optimizasyon fonksiyonları ile eğitim yaptırılmıştır. Performanslar Tablo 4.7’deki gibi görülmektedir.

Tablolar ve grafikler dikkatlice incelendiğinde en yüksek başarıyı VGG-13 göstermiştir. Tabloya katmanlar açısından bakıldığında önerilen AcıNet modeli, AlexNet ile aynı katmana sahip olmasına rağmen yaklaşık 10 kat daha az parametre ile daha çok başarı göstermiştir. Aynı şekilde parametre sayısına bakıldığında en az iki parametreye sahip AcıNet ve VGG-11 mimarisidir. Yine AcıNet, VGG-11 mimarisine göre yaklaşık 5 kat daha az parametreye sahip olmasına rağmen, yaklaşık %2 daha iyi performans göstermiş aynı şekilde hata ise %4 daha az olmuştur. Tablo 4.9’a bakıldığında AcıNet ile en başarılı model olan VGG-16 ya bakıldığında, VGG-16 mimarisi %2 oranında daha başarılı olurken aynı zamanda parametre sayısı da yaklaşık 15 kat fazladır. Buna rağmen Şekil 4.15 ve Şekil 4.18’e dikkatli bakıldığında test verileri üzerinde AcıNet’in bütün veri seti üzerinde çok daha kararlı, çok daha güvenilir bir sonuç çıkardığı ortaya çıkmıştır.

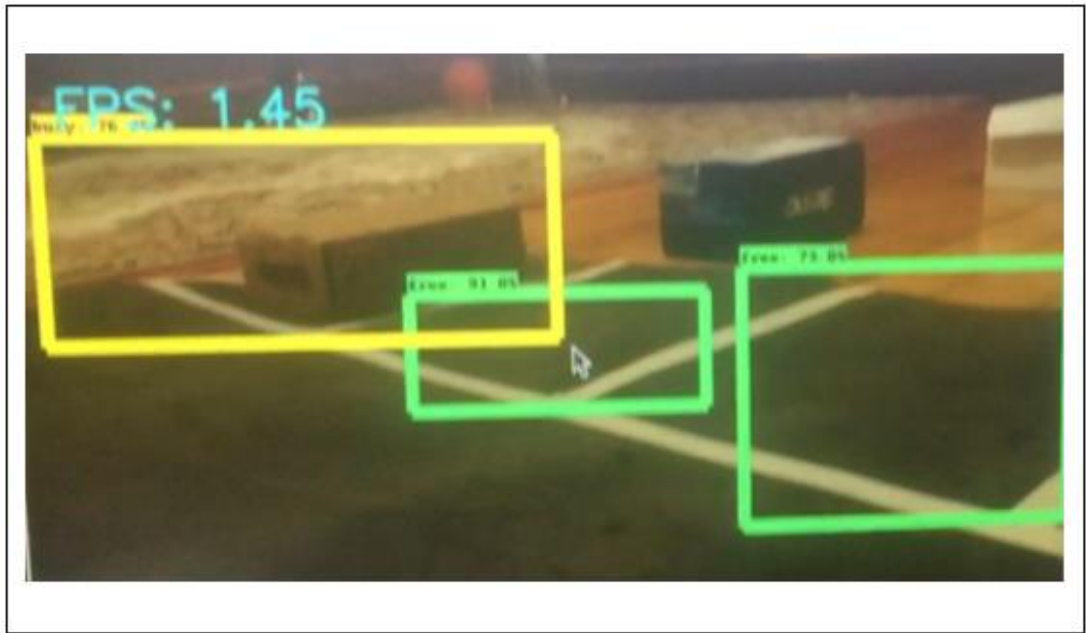
Sonuç olarak, derin öğrenme uygulamalarında en çok kullanılan mimarilerle kıyaslandığında AcıNet çok başarılı bir performans ortaya koymuştur. Bu model tasarlanırken hedeflemiş olan daha az işlemci gücü tüketme ve başarı, yapılan çalışmalar neticesinde tatmin edici şekilde karşılanmıştır.

4.7. Öğrenme Aktarımı

Gerçek zamanlı nesne tanıma ve sınıflandırma gibi uygulamalarından karşılaşılan sorunlardan biri modellerin başarı oranı çok yüksek olsa bile eğitilen verinin gerçek hayatta başarılı olacak kadar yeterli olmamasıdır. Bu gibi durumlarda yapılacak şeylerden biri ya da gerçek hayat verisini daha çok toplamak ya da elle toplanan az miktarda gerçek zaman veri ile öğrenme aktarımı yapmaktır [29].

Öğrenme aktarımı makine öğreniminde bir problemi çözerken kazanılan bilgiyi depolamaya ve o depolanan bilgiler ve probleme özgü yeni verilerle başka bir problemi çözmek için yapılan çalışmadır [30].

Bu çalışmada uzman kullanıcı tarafından toplanmış 866 adet 1500'den fazla etiketlenmiş veri ile transfer öğrenme işlemi yapılacaktır. Görüntü işleme de nesne tanıma ve sınıflandırma da en çok kullanılan modellerden bölüm 3.3' de bahsedilmiştir. RPI işlemci gücü ve belleği düşük olduğu için öğrenme aktarımında SSDlite MobileNet mimarisi kullanılmıştır.



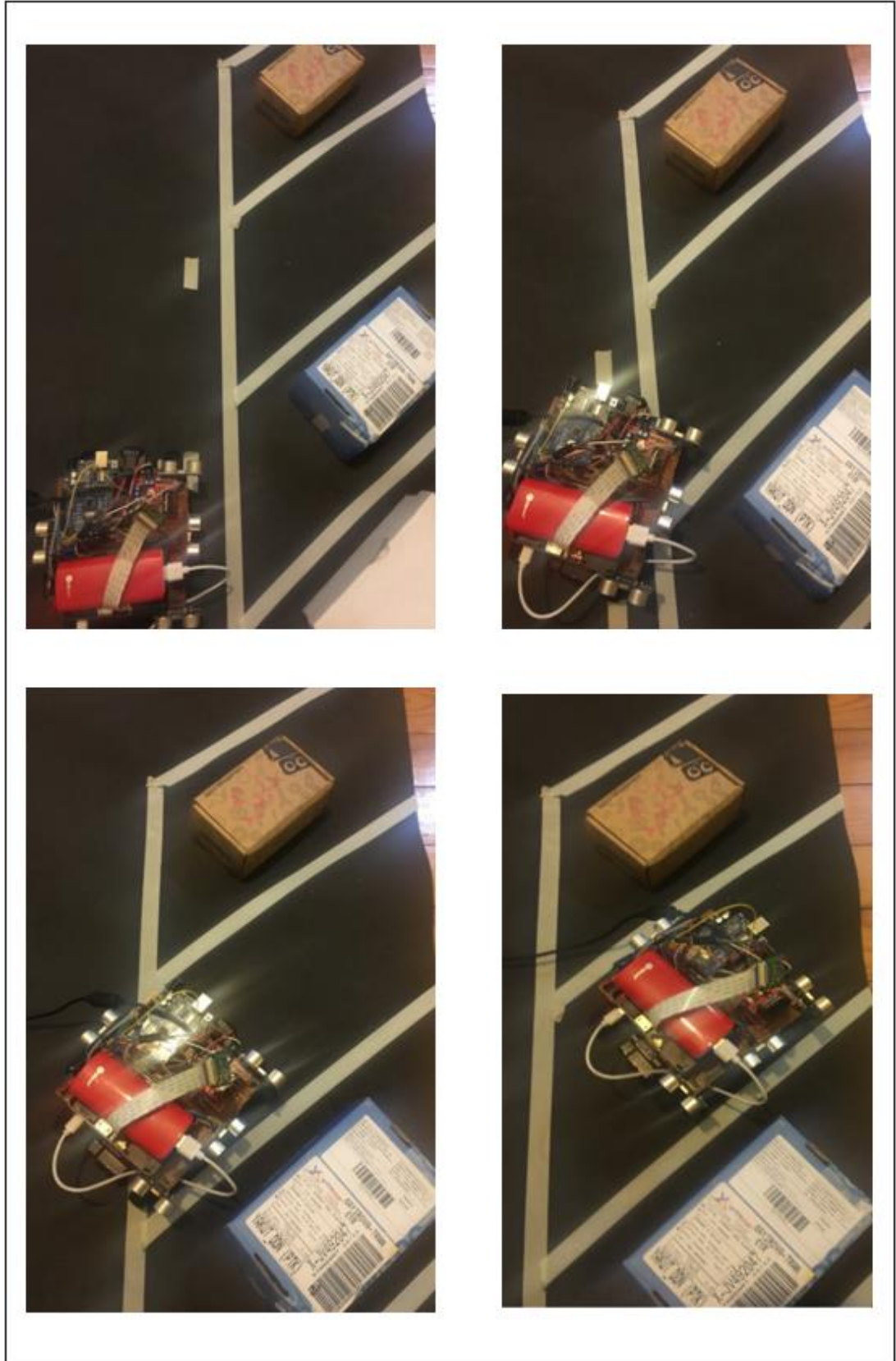


Şekil 4.20: Öğrenme aktarımı ile dik park yapmak için anlık olarak yapılan park yeri tespiti.

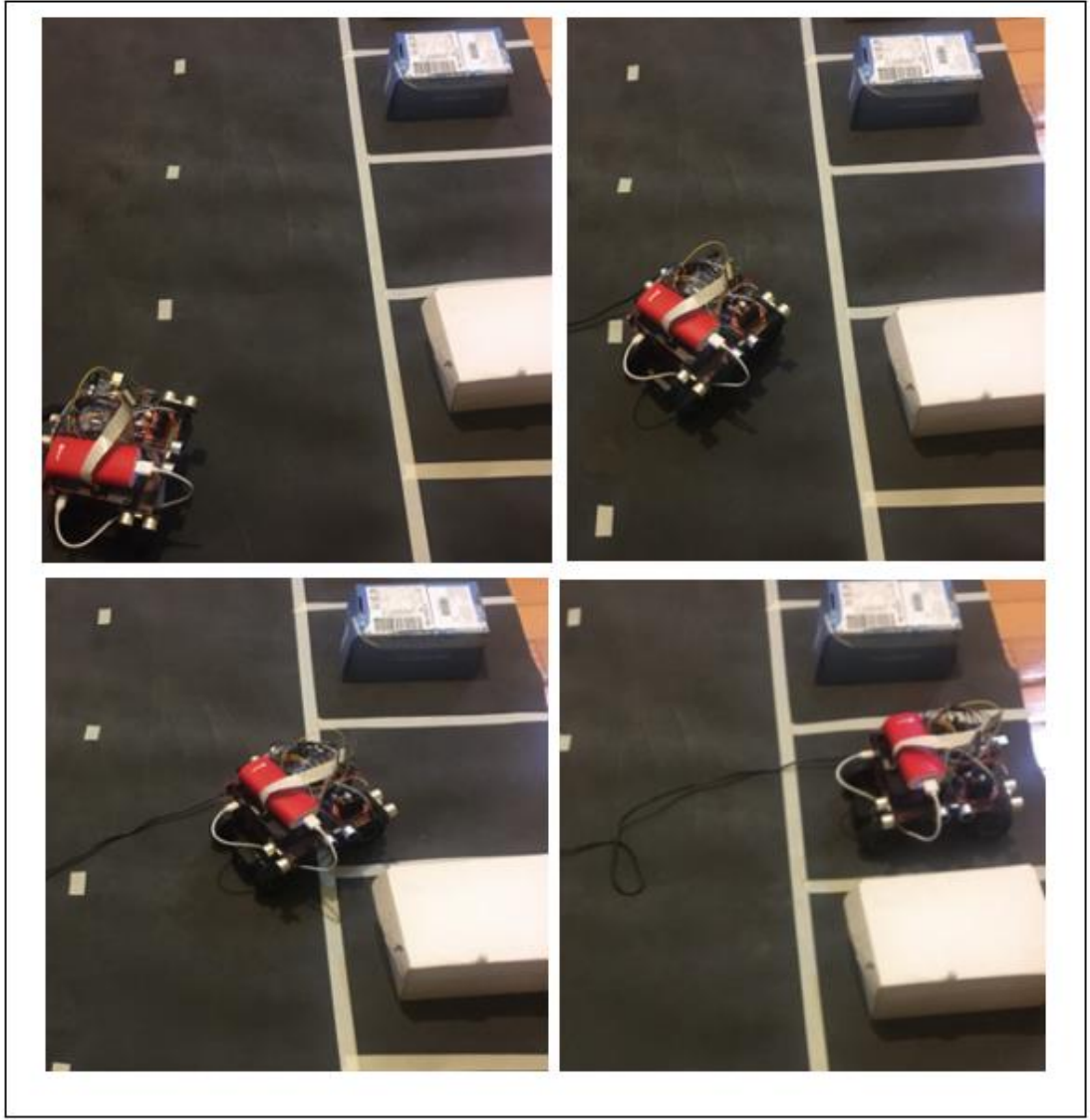
4.8. Park Yapma İşlemi

Öğrenme aktarımı ile Resim 4.18 te gösterildiği gibi park yeri başarılı şekilde yapılmıştır. Ek- A' da deneylerden görseller sunulmuştur.

Bu bölümde araç uygunluk tespiti durumuna göre park manevrası yapılacaktır. Bunu yapmak için araç kameradan aldığı canlı görüntü ile park yerinin boş olup olmadığını tespit edecek ve bu duruma göre aracın kontrolcüsüne veri gönderilecektir. Kontrolcünün içerisinde daha önceden yüklenmiş olan park manevra fonksiyonu yardımıyla alınan bu veriye göre ya park manevrası yapılacak ya da park yeri boşalana kadar beklemede kalınacaktır.



Şekil 4.21: Çapraz park manevrası.



Şekil 4.22: Dik park manevrası.

Şekil 4.21 ve 4.22 de aracın dik park manevrası ve çapraz park manevrası yapıldığı görülmektedir. Hem dik hem çapraz park için yapılan 50 deneme de sırasıyla yaklaşık %71 ve %76'lık oran ile araç park manevrası yapmıştır. Veriler göstermiştir ki aracın canlı sınıflandırma performansı, manevra performansına taşınmıştır.

5. SONUÇLAR VE GELECEK ÇALIŞMALARI

Son yıllarda teknolojik gelişmelerle birlikte yapay zekâ ve otonom sistemlerde gelişmeler hız kazanmıştır. Bu sistemlerin faydalı olduğu problemlerden biri de park yapma problemleridir. Ancak sistemlerin otonom hale gelmesi ve kendi karar mekanizmaları olması bazı beklenmedik kazalara yol açmaktadır. Bu durumda maddi manevi zararlara uğratmaktadır.

Bu tezde park problemine otonom bir çözüm üretmek için çalışmalar yapılmıştır. Bu çalışmada ilk olarak uzaktan kontrol edilen bir araç tasarlanmış olup uzman kullanıcı tarafından veri seti toplanmıştır. Park yerinin uygun olup olmadığını anlamak için toplanan veriler ile yeni bir derin öğrenme modeli geliştirilmiştir. Bu veriler ile aynı zamanda saha da öğrenme aktarımı sayesinde başarılı şekilde nesne sınıflandırması yapılmıştır. Aynı şekilde araçtan anlık olarak alınan görüntüler ile de park yeri boş ise araç kontrolcüsüne veri göndererek park manevrası yapılması sağlanmıştır.

Gerçek zamanlı uygulamalarda derin katmanlı çok başarılı sonuçlar veren birçok derin öğrenme modeli vardır. Ancak bu modeller her uygulama için uygun olmayabilmektedir. Bu çalışmada göstermiştir ki düşük sınıflandırmanın yapıldığı uygulamalarda daha düşük katmanlı modeller de iyi sonuçlar vermektedir. Böylece daha ucuz maliyetli sistemler tasarlanabilir. Sonuçlar göstermiştir ki park problemi için hedeflenen çalışma başarılı şekilde sağlanmıştır.

Gelecek çalışması adına, otonom araç geliştirilirken, işlemci gücü daha yüksek bir kart tercih edilebilir. Aynı şekilde daha güçlü motor kullanarak aracın yavaş ya da hızlı gitmesine olumlu cevap vererek park manevrasındaki hataya azaltılabilir. Bu sayede araç daha hızlı şekilde park yerini tespit edecek ve park manevra başarısını artıracaktır. Bunun dışında daha çok veri toplanarak daha yüksek başarı da derin öğrenme modeli geliştirilebilir.

KAYNAKLAR

- [1] Ozguner U., Acatman T. ve Redmil K., (2011), “Autonomous Ground Vehicles”, USA, Artech House, 3-5.
- [2] Web 1, (2020), <http://www.idsia.ch/~juergen/robotcars.html>, (Eriřim Tarihi: 18.06.2020).
- [3] Pomerleau D.A., (1989), “ALVINN: An Autonomous Land Vehicle in a Neural Network”, In Advances in Neural Information Processing Systems, Morgan Kaufmann, San Mateo.
- [4] ayırođlu İ., řimřir M., (2008), “PIC ve Step Motorla Sürölen Bir Mobil Robotun Uzaktan Kamera Sistemi ile Kontrolö”, Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 24(1-2), 1-16.
- [5] Fong P. E., Yusoff M. A., (2011), “Real-Time Control of Wi-Fi Surveillance Robot”, Proceeding of the International Conference on Advanced Science Engineering and Information Technology, Malasia.
- [6] Kuo G., Cheng C., Wu C., (2014), “Design and Implementation of a Remote Monitoring Cleaning Robot”, International Automatic Control Conference, Taiwan.
- [7] Espes D., Autret Y., Vareille J., Le Parc P., (2014), “Designing a Low-Cost Web-Controlled Mobile Robot for Home Monitoring”, The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Italy.
- [8] etinkaya A., (2017), “Otonom Bir robotun bulanık kontrolör yaklaşımı ile konum kontrolö”, Karatay Üniversitesi, YL Tezi.
- [9] Bingöl S., Kaymak ., Uar A., (2019), “Derin Öđrenme Kullanarak Otonom Araların İnsan Sürüşünden Öđrenmesi”, Fırat Üniversitesi, Fırat Üniversitesi Müh. Bil. Dergisi 31(1), 177-185.
- [10] Alpkıray N., (2019), “Otonom Keřif Amalı Robot Sistemleri İçin Geri dönüş Rotası Hesaplama Algoritması Geliřtirilmesi”, Cumhuriyet Üniversitesi, YL Tezi.
- [11] Notomista G., Ve Botsch M., (2016), “A Machine Learning Approach For The Segmentation Of Driving Maneuvers And Its Application In Autonomous Parking”, JAISCR, 2017, Vol. 7, No. 4, pp. 243 – 255.
- [12] Zhang P., Xiong L., Yu Z., Fang P., Yan S., Yao S., Zhou Y., (2019), “Reinforcement Learning-Based End-to-End Parking for Automatic Parking System”, sensors 2019, 19, 3996.

- [13] Kocic J., Jovicic N., Drndarevic V., (2019), “An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms”, *sensors* 2019, 2064.
- [14] Moon J., Bae I., Kim S., (2019), “Automatic Parking Controller with a Twin Artificial Neural Network Architecture”, *Hindawi* 2019, Article ID 4801985.
- [15] Web 2, (2020), <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>, (Eriřim Tarihi: 25.06.2020).
- [16] Web 3, (2020), <https://store.arduino.cc/usa/arduino-uno-rev3/>, (Eriřim Tarihi: 26.06.2020).
- [17] Web 4, (2020), <https://docs.rs-online.com/2888/0900766b8127db0a.pdf/>, (Eriřim Tarihi: 27.06.2020).
- [18] Web 5, (2020), <https://www.arduino-media.com/arduino-ile-bluetooth-konfigurasyon-ayarlari-bluetooth-at-komutlari/>, Son Eriřim Tarihi: 27.06.2020.
- [19] LeCun Y, Bottou L, Bengio Y, (1998), P., “Gradient-based learning applied to document recognition”. In *Proceedings of the IEEE*, 86(11): 2278-2323.
- [20] Krizhevsky A., Sutskever I., Hinton G., (2012), “ImageNet Classification with Deep Convolutional Neural Networks”, *NIPS* 2012.
- [21] Simonyan K., Zisserman A., (2015), “Very deep convolutional Networks for large-scale image detection”, *ICLR* 2015.
- [22] Web 6, (2020), <http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>, (Eriřim Tarihi: 10.01.2020).
- [23] Szegedy C., Liu W., Jia Y., Erhan D., (2015), “Going deeper with Convolutions”, *CVPR* 2015, Las Vegas, NV, USA.
- [24] He K., Zhang X., Ren S., Sun J., (2016), “Deep residual learning for image recognition”, 770 778., *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 27-30 June 2016, Las Vegas, NV, USA.
- [25] Dař R., Polat B., Tuna G., (2019), “Derin öğrenme ile resim ve videolarda nesnelerin tanınması ve takibi”, *Fırat Üniversitesi Müh. Bil. Dergisi* 31(2), 571-581.
- [26] Web 7, (2020), https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md/, (Eriřim Tarihi: 01.01.2020).
- [27] Web 8, (2020), [https://tr.wikipedia.org/wiki/Python_\(programlama_dili\)](https://tr.wikipedia.org/wiki/Python_(programlama_dili)), (Eriřim Tarihi: 10.01.2020).

- [28] Web 9, (2020), <http://cnrpark.it/>, (Eriřim Tarihi: 10.01.2020).
- [29] Oturak M, (2018), “Öğrenme aktarımı ile sınıflandırma”, Yüksek lisans Tezi, Hacettepe Üniversitesi.
- [30] Goodfellow L., Bengio Y., Courville A., (2016), “Deep learning book”, MIT Press, 536.
- [31] Web 10, (2020), <https://ayyucekizrak.medium.com/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd/>, (Eriřim Tarihi: 20.01.2020).
- [32] Web 11, (2020), <https://www.raspberrypi.org/products/>, (Eriřim Tarihi: 26.06.2020).

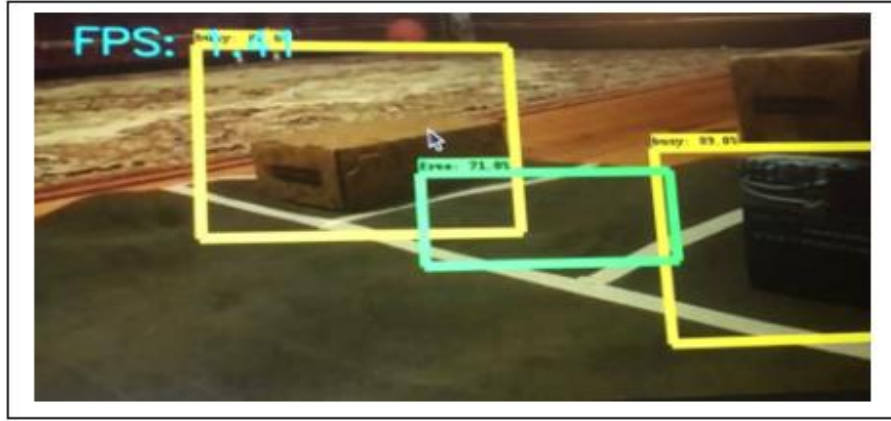
ÖZGEÇMİŞ

Ebubekir Sıddık CEYLAN 1994 yılında Kocaeli’ de doğdu. 2012 yılında başladığı Sakarya Üniversitesi Mühendislik Fakültesi Elektrik Elektronik Mühendisliği Bölümünü 2016 yılında başarıyla tamamlayarak mezun oldu. 2018 yılında Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalında başladı. Şu an da bilişim vadisinde gömülü yazılım mühendisi olarak çalışma hayatına devam etmektedir.

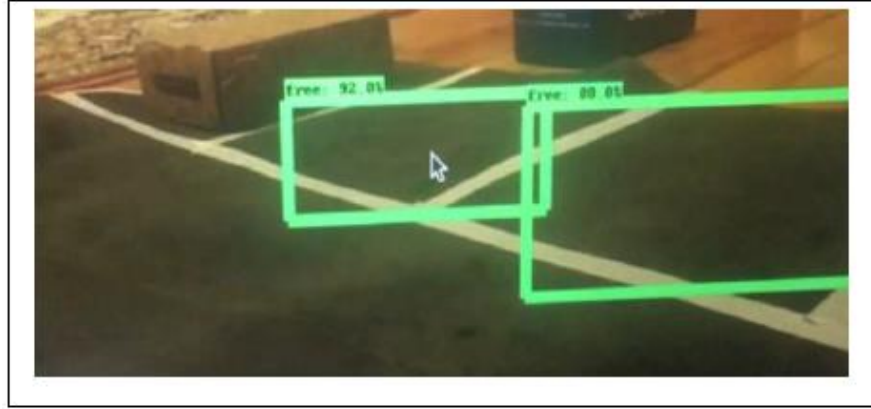
EKLER

Ek A: Nesne Sınıflandırma Konusundaki Çalışmalar

Aşağıda görsellerde aracın farklı test senaryolarındaki park yeri tespiti gösterilmektedir. Her park senaryosunda park yerlerine farklı objeler koyularak eğitilen modelin performansı incelenmiştir.



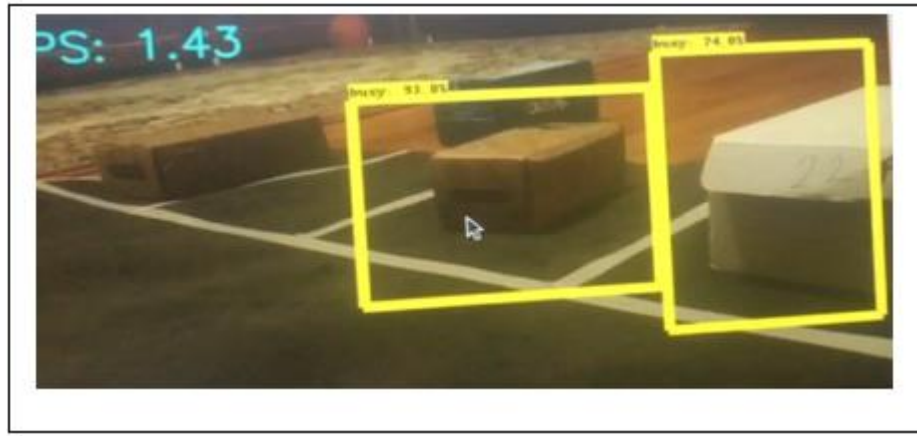
Şekil B1.1: 1. Test nesne sınıflandırma senaryosu



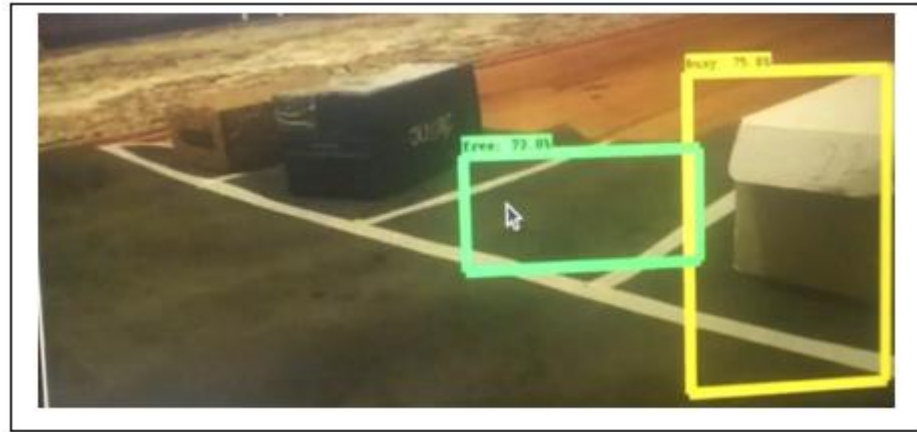
Şekil B1.2: 2. Test nesne sınıflandırma senaryosu



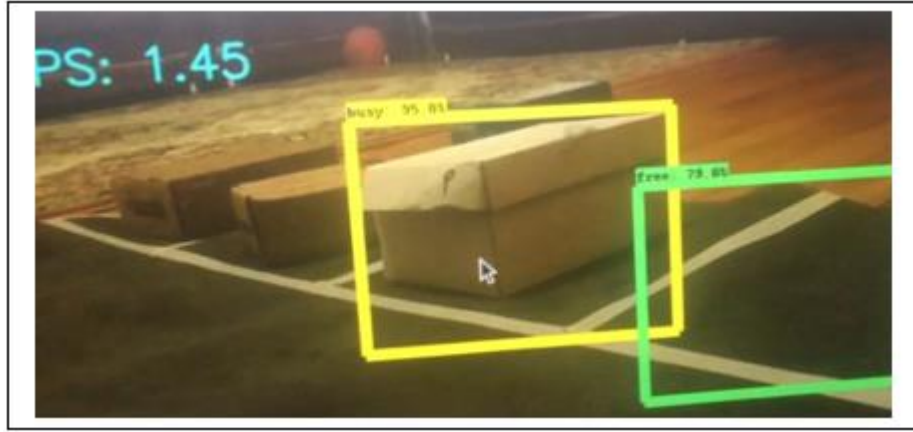
Şekil B1.3: 3. Test nesne sınıflandırma senaryosu



Şekil B1.4: 4. Test nesne sınıflandırma senaryosu



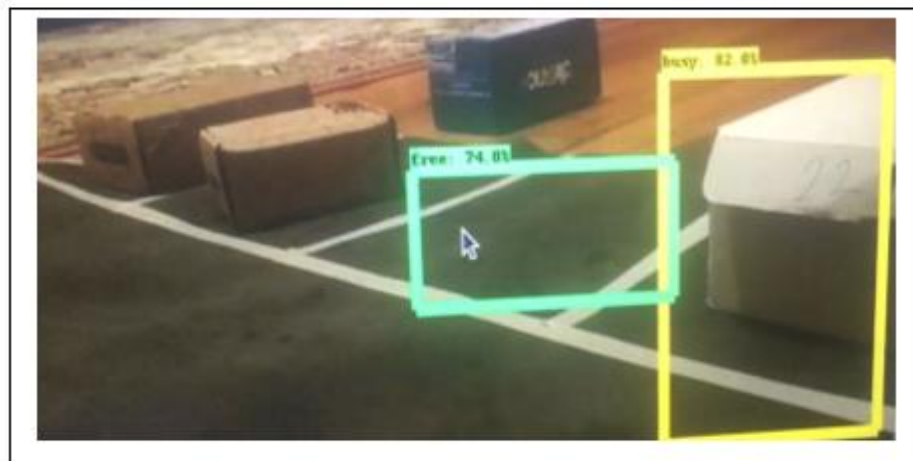
Şekil B1.5: 5. Test nesne sınıflandırma senaryosu



Şekil B1.6: 6. Test nesne sınıflandırma senaryosu



Şekil B1.7: 7. Test nesne sınıflandırma senaryosu



Şekil B1.8: 8. Test nesne sınıflandırma senaryosu