DERİN ÖĞRENME TABANLI SÜRÜCÜSÜZ ARAÇ SİSTEMLERİ

Koray AKİ

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME TABANLI SÜRÜCÜSÜZ ARAÇ SİSTEMLERİ

Koray AKİ
Orcid No: 0000-0002-3661-3058

Doç. Dr. Ahmet Emir DİRİK
Orcid No: 0000-0002-6200-1717

(Danışman)

YÜKSEK LİSANS
BİLGİSAYAR MÜHENDİSLİĞİ

BURSA – 2019

TEZ ONAYI

Koray AKİ tarafından hazırlanan "Derin Öğrenme Tabanlı Sürücüsüz Araç Sistemleri" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Doç. Dr. Ahmet Emir DİRİK
Orcid No: 0000-0002-6200-1717
İmza

Başkan : Doç. Dr. Ahmet Emir DİRİK
Orcid No: 0000-0002-6200-1717
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi,
Bilgisayar Mühendisliği Anabilim Dalı

Üye : Doç. Dr. Ersen Yılmaz
Orcid No: 0000-0002-6620-655X
Bursa Uludağ Üniversitesi
Mühendislik Fakültesi
Elektrik-Elektronik Mühendisliği Bölümü
İmza

Üye : Doç. Dr. Cemal HANİLÇİ
Orcid No: 0000-0002-9174-0367
Bursa Teknik Üniversitesi
Mühendislik ve Doğa Bilimleri Fakültesi
Elektrik-Elektronik Mühendisliği Bölümü
İmza

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
../../2019

U.Ü. Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

16/09/2019

Koray AKİ

ÖZET

Yüksek Lisans Tezi

DERİN ÖĞRENME TABANLI
SÜRÜCÜSÜZ ARAÇ SİSTEMLERİ

Koray AKİ

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ahmet Emir DİRİK

Makine öğrenmesi (Machine Learning) ve özellikle derin öğrenme (Deep Learning) alnındaki gelişmeler pek çok farklı alanda ve özellikle de karmaşık problemlerin çözümü gibi birçok uygulamanın geliştirilmesine olanak sağlamaktadır. Makine öğrenmesi otomotiv endüstrisi üzerinde ve sürücüsüz araçların geliştirilmesinde büyük bir etkiye sahiptir. Sürücüsüz araç, insan müdahalesi olmadan kendi kendine hareket edebilen bir araçtır. Son yirmi yılda, sürücüsüz araçlar askeri, lojistik ve endüstriyel üretimdeki potansiyel uygulamaları ile hem akademiden, hem de endüstriden büyük ilgi görmeye başlamıştır. Sürücüsüz araçların geliştirilmesi, ölüm sayısını ve günümüz trafiğinin çevresel etkilerini azaltmak gibi birçok konuda toplumsal fayda sağlamaktadır. Sürücüsüz araç herhangi bir insan etkileşimi olmadan kendi kendini yönlendirebilmektedir. Sürücüsüz araçlar navigasyon için GPS, çarpışmaları önlemek için sensörler ve nesne tespit etmek için kameralar gibi çeşitli teknolojiler kullanmaktadırlar. Derin öğrenme ve PID kontrol ile otonom sürüş yapılabilmektedir. Bu tez çalışmasında simülasyon ortamında sürücüsüz araç eğitimi gerçekleştirilmiştir. Aynı zamanda PID kontrol ile aracın otonom hareket etmesi sağlanmış ve derin öğrenme ile eğitilen aracın otonom hareketi ile PID kontrol ile otonom hareket eden aracın performansları karşılaştırılmıştır.

Anahtar Kelimeler: Derin öğrenme, sürücüsüz araç, evrişimsel sinir ağları
2019, ix + 95 sayfa.

ABSTRACT

MSc Thesis

DEEP LEARNING BASED
AUTONOMOUS VEHICLE SYSTEMS

Koray AKİ

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

Supervisor: Assoc. Prof. Dr. Ahmet Emir DİRİK

Developments in Machine Learning and in particular Deep Learning focuses on the complexity of applications in a wide range of areas and in various issues related to solving problems. Machine learning has a significant impact on the automotive industry and in the development of autonomous vehicles. Autonomous vehicles is a vehicle which can drive itself without human intervention. Over the last two decades, autonomous vehicles have been receiving considerable interest from both academia and industry, with potential applications in military, logistics and industrial production. The development of autonomous vehicles provides social benefits in many aspects, such as reducing the number of deaths and reducing the environmental impact of today's traffic. The autonomous vehicle can steer itself without any human interaction. Autonomous vehicles use various technologies such as GPS for navigation, sensors to avoid collisions, and cameras for object detection. Autonomous driving can be performed with Deep Learning and PID control. In this study, autonomous vehicle training was conducted in a simulation environment. At the same time, the autonomous movement of the vehicle is achieved with PID control and the performances of the autonomous movement of PID control and the autonomous movement of the vehicle trained with deep learning are compared.

Key words: Deep learning, autonomous vehicle, convolutional neural networks
2019, ix + 95 pages.

TEŞEKKÜR

Bu tez çalışmasının gerçekleştirilmesinde, değerli bilgilerini benimle paylaşan, kendisine danıştığımda bana kıymetli zamanını ayırıp sabırla ve büyük bir ilgiyle bana faydalı olabilmek için elinden gelenden fazlasını sunan, güler yüzünü ve samimiyetini benden esirgemeyen, iş disiplini ve profesyonelliğiyle bana örnek olan ve her sorun yaşadığımda yanına çekinmeden gidebildiğim danışmanım Doç. Dr. Ahmet Emir DİRİK'e, sürüş testlerinin değerlendirilmesinde yardımcı olan Emmanuel KIEGAING'e ve bu zorlu süreçte en büyük desteği ve sevgiyi vererek her zaman yanımda olup, çalışmalarımı büyük bir sabırla destekleyen, beni motive eden sırdaşım, yoldaşım, biricik eşim Serpil AKİ'ye teşekkürlerimi sunarım.

Koray AKİ
16/09/2019

İÇİNDEKİLER

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
β	Momentum
α	Öğrenme Katsayısı
σ	Varyans
b	Bias
ϕ	Bernoulli Parametresi
J	Maliyet Fonksiyonu
L	Kayıp Fonksiyonu
π	Politika
K_P	Oransal Kontrolör Kazancı
K_i	İntegral Kontrolör Kazancı
K_D	Türev Kontrolör Kazancı

Kısaltmalar	Açıklama
ML	Makine Öğrenmesi
DL	Derin Öğrenme
NN	Sinir Ağı
CNN	Evrişimsel Sinir Ağı
PID	Oransal İntegral Türev Kontrol
YSA	Yapay Sinir Ağları
GPU	Grafiksel İşlem Birimi
YZ	Yapay Zeka
SVM	Destek Vektör Makinesi
CTE	Çapraz Yol Hatası

Çeviriler	Açıklama
Machine Learning	Makine Öğrenmesi
Deep Learning	Derin Öğrenme
Reinforcement Learning	Pekiştirmeli Öğrenme
Neural Network	Sinir Ağı
Convolution Neural Networks	Evrişimsel Sinir Ağı
Support Vector Machine	Destek Vektör Makinesi
Graphical Processing Units	Grafiksel İşlem Birimi
Proportional	Oransal
Integral	İntegral
Derivative	Türev
Artificial Intelligence	Yapay Zeka
Artificial Neural Network	Yapay Sinir Ağı
Back Propagation	Geriye Yayılım

Supervised Learning	Denetimli Öğrenme
Reinforcement Learning	Pekiştirmeli Öğrenme
Batch	Yığın
Dropout	Seyreltme
Node	Düğüm
Threshold	Eşikleme
Activation	Aktivasyon
Gradient Descent	Gradyan İnişi
Linear	Doğrusal
Convolution	Evrişim
Pooling	Ortaklama
Fully Connected	Bütün Bağlı
Learning Rate	Öğrenme Katsayısı
Zero Padding	Sıfır Ekleme
Architecture	Mimari
Stride	Kaydırma
Early Stopping	Erken Durdurma
Epoch	Adım
Multiple	Çoklu
Least Squared Error	En Küçük Kareler Hatası
Least Mean Squares	En Küçük Ortalama Kareler Algoritması
Logistic Loss	Lojistik Kayıp
Hinge Loss	Menteşe Kaybı
Cross Entropy	Çapraz Entropi
Regresyon	Bağlanım
Locally Weighted Regression	Yerel Ağırlıklı Regresyon
Optimal Margin Classifier	Optimal Marj Sınıflandırıcı
Kernel	Çekirdek
Generative Learning	Üretici Öğrenme
Confusion Matrix	Çapraz Tahmin Tablosu
Regularization	Düzenleştirme
Stochastic	Rasgele
Bias	Önyargı (Eşik)
Exploration	Keşif
Exploitation	Sömürü
Cross Track Error	Çapraz Yol Hatası
Batch Normalization	Yığın Normalizasyonu

ŞEKİLLER DİZİNİ

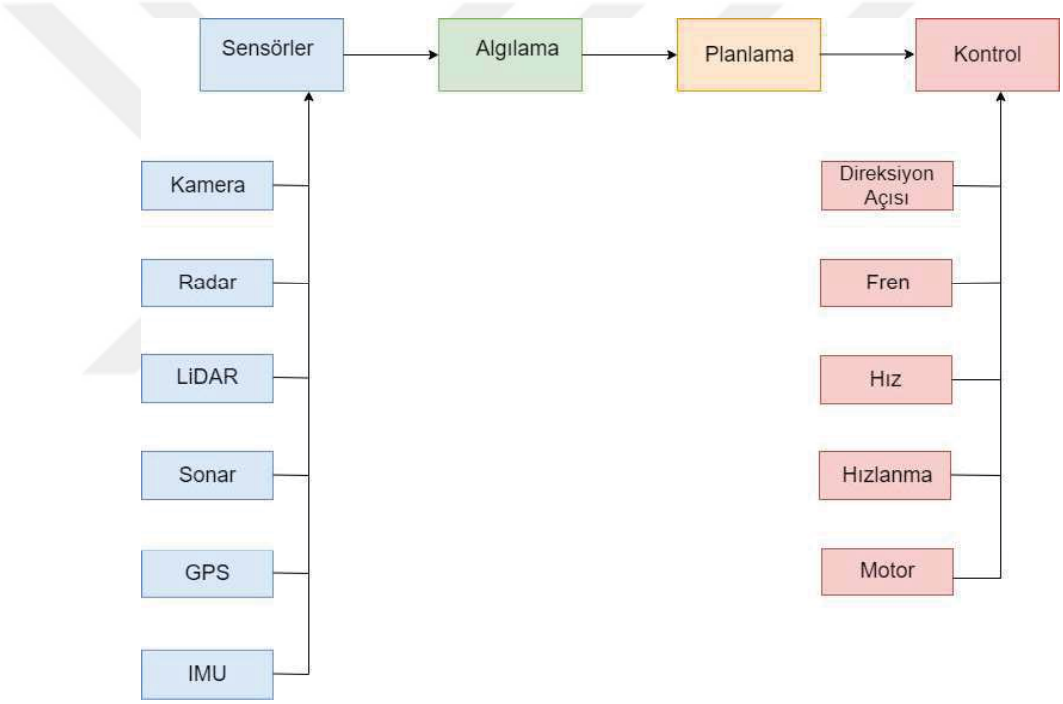
Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Makine öğrenmesi (Machine Learning, ML) ve Derin Öğrenme (Deep Learning, DL) alanındaki gelişmeler, pek çok farklı alanda ve özellikle de karmaşık problemlerin çözümü gibi birçok uygulamanın geliştirilmesine olanak sağlamaktadır. Makine öğrenmesi otomotiv endüstrisinde sürücüsüz araçların geliştirilmesinde çok büyük etkiye sahiptir. Makine öğrenmesi, nesne tespiti, yol planlaması ve haritalama gibi birçok sürücüsüz araç alt sisteminde kullanılmaktadır. 2000'li yıllardan itibaren sürücüsüz araç gelişimi oldukça hızlanmıştır. Araçların otomatik park etmesi, şerit takibi, trafik ışıklarını tanıma ve nesnelerin tespiti gibi birçok alanda kullanılmaktadır.



Şekil 1.1. Sürücüsüz Araç Alt Sistemleri

Sürücüsüz araçlar genel olarak sensörler, algılama, planlama ve kontrol olmak üzere dört modüle ayrılabilir. Şekil 1.1'de sürücüsüz araçlara ait sistem diyagramı gösterilmektedir. Sürücüsüz araçların kararlarını almak için kullandığı bilgiler; kamera, radar ve LiDAR gibi sensörlerden oluşur. Bu tip sensörler genellikle büyük miktarlarda veri üretir. Sensörlerden gelen bilgiler, algılama modülüne gelir. Algılama modülü sensör verilerini anlamlı bilgilerle birleştirir. Planlama modülü, algılama modülü çıktısını, davranış planlama ve hem kısa, hem de uzun menzilli yol planlaması için kullanır. Kontrol modülü,

aracın planlama modülü tarafından üretilen verilerle yolu takip etmesini sağlar ve araca kontrol komutları gönderir.

Makine Öğrenmesinin alt dalı olan Derin Öğrenme, girdi olarak yüksek boyutlu verileri olan sorunları çözmek için kullanılmaktadır. Derin Öğrenme yöntemleri, yüksek boyutlu verileri, Sinir Ağı (Neural Network, NN) olarak adlandırılan çok katmanlı bir mimari kullanan fonksiyon belirleyiciden geçirip, yüksek boyutlu verilerden ilgili özellikleri çıkarabilir. Son zamanlarda meydana gelen gelişmeler nedeniyle derin öğrenmeye ilgi oldukça artmıştır. Evrişimsel Sinir Ağlarının (Convolution Neural Networks, CNN) ve DL mimarilerinin geliştirilmesine en önemli katkı ImageNet yarışmasında DL algoritmalarının sağladığı iyileştirmelerdi (Russakovsky ve ark. 2015). ImageNet yarışmasında fotoğraftaki objelerin sınıflandırılmasında 2011 yılında %75 civarında olan başarım seviyesini 2012 yılında %85 seviyelerine çıkartmışlar ve 2015 yılında ise daha yüksek bir performans sergileyerek insanlardan daha iyi performans göstermiştir. Diğer bir gelişme ise sinir ağlarının daha hızlı eğitilmesini sağlayan grafiksel işlem birimlerinin (Graphical Processing Units, GPU) hesaplama kapasitesinin arttırılmasıdır. Birkaç yıl içinde ImageNet yarışmasını kazanan algoritmalar NN ve DL mimarilerinin gelişmesinde araştırmacılara ilhan kaynağı olmuştur. Geliştirilen mimarilerden bazıları; Toronto Üniversitesi'nden SuperVision grubu tarafından tasarlanan 1,2 milyon görüntüyü 1000 sınıfa ayırabilen AlexNet (Krizhevsky ve ark. 2012), Oxford Üniversitesi'nden VGG (Visual Geometry Group) tarafından tasarlanan VGG-16 modeli (Simonyan ve Zisserman 2015), Google araştırmacıları tarafından tasarlanan GoogLeNet (Szegedy e ark. 2015), 152 katman derinliği ile Microsoft araştırmacıları (2016) tarafından tasarlanan ResNet (Residual Neural Network) ve Politecnico di Milano ve Montreal Üniversitesi araştırmacıları (2015) tarafından tasarlanan ReNet (A Recurrent Neural Network)'tir. Bu gelişmelere ek olarak AlphaGo isminde bilgisayar programının kısa sürede çözülemeyeceği iddia edilen, Go oyununda dünya şampiyonu olan Lee Sedol'u, Mart 2016'da yenmeyi başardı. Go oyununda elde edilen bu büyük ilerleme derin öğrenme çalışmalarına ilişkin yeniden bir tartışma başlattı ve derin öğrenmeyi daha popüler hale getirdi.

Derin öğrenmenin dezavantajlarından biri, bu ağların büyük miktarda veriye ihtiyaç duymasıdır. Veri arttırma işlemleri için pek çok yöntem mevcuttur, ancak bu yöntemlerle bile, yeterince büyük veri kümesi oluşturmak zor olabilir. Bundan dolayı, eğitim sırasında ML algoritmaları için sentetik verilerin kullanımı aktif bir araştırma alanı olmuştur (Alzantot ve ark. 2017, Johnson-Roberson ve ark. 2017, Zhang ve ark. 2015). Otonom sürüş için, eğitim verilerinin elde edilmesi zordur. Sentetik veriler kullanılarak eğitim gerçekleştirildiğinde bu zorluğun üstesinden gelinebilmektedir.

Bu tez çalışmasında sürücüsüz bir aracın özerk hareketi için Derin Öğrenme modeli oluşturularak, bir sinir ağı eğitimi gerçekleştirilmiş; ayrıca PID (Proportional Integral Derivative) kontrol kullanılarak da aracın özerk hareketi simülasyon ortamında gerçekleştirilmiştir. Sürücüsüz araç eğitimi için veriler sürüş simülatöründe toplanmıştır. NN'i eğitmek için direksiyon açıları ve kamera görüntüleri kullanılmıştır. Eğitim sonunda, simülasyon ortamında gerçekleştirilen testlerde, sürücüsüz araca çıkış olarak direksiyon açısı verilmiş ve araç hızı PID kontrol ile sabitlenerek aracın otonom hareket etmesi sağlanmıştır. Simülasyon ortamında sürüş pistinde bir tam turda aracın yolun orta noktasından sapması incelenerek PID kontrol ve derin öğrenme algoritması karşılaştırılmıştır.

Bu tez çalışması 5 bölümden oluşmaktadır. Giriş bölümünde genel bir bilgilendirme yapılmıştır. Kuramsal Temeller ve Kaynak Araştırması bölümünde sürücüsüz araç için PID kontrol ve derin öğrenme algoritmaları kullanılarak gerçekleştirilen otonom sürüş örnekleri ile ilgili literatür detaylıca incelenmiş ve sürücüsüz araç örnekleri verilmiştir. Materyal ve Yöntem bölümünde, sürücüsüz araç için derin öğrenme tabanlı ve PID kontrol ile araç hareketi detaylıca incelenmiş ve kullanılan yöntemlerden bahsedilmiştir. Bulgular bölümünde elde edilen sonuçlar ve gerçekleştirilen karşılaştırmalı deneylerden bahsedilmiştir. Tartışma ve Sonuç bölümünde tez çalışmasında kullanılan yöntemlerin eksik yanlarından pozitif ve negatif yönleri belirtilerek, sonraki yapılacak çalışmalara ilhan kaynağı olması açısından yorumlanmıştır.

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

Ülkemizde ve dünya genelinde her geçen gün araç kullanımı artmaktadır. Bugün hemen hemen her evde en az bir araç vardır. Bununla birlikte, otomobillerin yaygın kullanımı, trafik kazalarına, hava kirliliğine ve trafik sıkışıklığı gibi hayatımızı olumsuz etkileyecek durumlara yol açmaktadır. Sürücüsüz araçlar bu sorunları azaltma ve yüksek maliyetli harcamalardan tasarruf etmeyi sağlayabilmektedir. Araç sürücüleri dikkatsizlik sonucu trafikte kazaya karışabilmektedir. Bu kazalar yaralanmalarla ve bazen de ölümlerle sonuçlanabilmektedir. Birçok çalışma sürücülerin hatasının kazaların ana nedeni olduğunu göstermektedir. Eygü (2018), 2010-2015 yılları arasında trafik kazalarını etkileyen faktörler üzerine bir inceleme gerçekleştirmiş ve hava şartları, yol ve çevre özelliklerinin yanı sıra sürücü faktörünün de kazaya %75 etkisinin olduğunu tespit etmiştir. Çavdar ve ark. (2008), Türkiye'de son on yılda gerçekleşen trafik kazalarına neden olan kusur oranları üzerine bir inceleme gerçekleştirmiş ve sürücü hatalarından kaynaklanan kazaların toplam kazalara oranının %95 olduğunu tespit etmiştir. Sungur ve ark. (2014), Türkiye'deki trafik kazalarının analizi üzerine bir inceleme gerçekleştirmiş ve istatistiklere göre %95 oranı ile en büyük kusurun sürücülerde olduğu tespit edilmiştir. Dünya geneline de bakıldığında aynı durum söz konusudur. Indiana Üniversitesi'nde Treat ve ark. tarafından (1979) trafik kazalarının nedenleri üzerine yapılan bir çalışmada, hızlanma, dikkatsizlik ve sürücü faktörlerinin tüm kazaların %92,6'sını oluşturduğu tahmin edilmektedir. Beyaz Saray tarafından (2014) gerçekleştirilen bir araştırmada trafik kazalarında sürücü faktörü oranının %90 ile %95,4 arasında değiştiği görülmüştür. Trafik kazaları dışında sürücülerin araba kullanırken ya da arabayı park ederken harcadığı zaman da göz önünde bulundurulduğunda sürücüsüz araçlar vazgeçilmez bir hale gelmeye başlamıştır. Shanker ve ark. (2013) tarafından hazırlanan raporda ABD'deki sürücüler her yıl yaklaşık 75 milyar saat araba kullanıyor. Sürüş sırasında ve park etme konusunda harcanan zaman kaybının yanında araç yakıtı da maliyet yaratmaktadır.

Trafik kazalarının çoğu sürücülerin hatalarından kaynaklanmaktadır. Sürücülere yardımcı olacak veya kazalarda ölüm oranlarını azaltacak teknolojileri kullanmak tarihsel sürece bakıldığında sürüşü daha güvenli hale getirmiştir. Trafik kazalarında 1968 yılından önce emniyet kemeri olmadığında ölüm oranları yüksek olmaktaydı. Emniyet kemerlerinin kullanılmasıyla birlikte ve yakın zamanlarda, gelişmiş hava yastıkları, lastik

basıncı izleme sistemleri ve fren sistemlerinin gelişmesi kazalarda ölüm oranının azalmasını sağlamıştır (Chen 2016). Araçlardaki teknolojik gelişmeler sürüş güvenliğini arttırmaktadır. Tesla otomobillerinde otopilot kullanımı çarpışma oranlarında %39'luk bir azalmaya yol açmıştır (Habib 2017). IIHS'ye göre (2016), otomatik frenleme sistemi aracın arkasından çarpmaları ortalama %40, ileriye doğru çarpışma uyarısı ise kazaları %23 azalttığı görülmüştür. Ayrıca, otomatik fren sistemi kazalarda yaralanmaları %42 oranında düşmüştür. Herkesin sürücüsüz araç kullanması durumunda kazaların %90 civarında azalması beklenmektedir. Sürücüsüz araçlar trafik tıkanıklığını ortadan kaldırarak yakıt tüketimini ve sera gazı emisyonlarını da azaltacaktır.

Bugün piyasadaki birçok araç, uyarlanabilir hız kontrolü, şerit kontrolü ve park yardımı gibi bazı otomasyon seviyelerini içerir (Çizelge 2.1). Otonom araçları Seviye 0'dan Seviye 5'e kadar altı seviyede inceleyebiliriz. Seviye 0'da, sürücü aracın frenini, direksiyonunu, gaz kelebeğini ve hareket gücünü kontrol eder. Seviye 0'da tüm sorumluluk sürücüye aittir. Seviye 1'de, bazı kontrol fonksiyonları otomatiktir ancak birbirlerinden bağımsız olarak çalışır. Sürücü bu otomatik sistemlere kontrol verebilir veya sistemler sürücünün kontrolünü otomatik olarak alabilir. Seviye 1 araçtaki sistemlere örnek olarak hız kontrolü ya da direksiyon yönlendirmesinde sürücüye yardımcı olabilir. Seviye 1'de de tüm sorumluluk sürücüye aittir. Seviye 2'de, iki veya daha fazla kontrol işlevi birlikte çalışabilir. Sistem hızı ve direksiyonu aynı anda kontrol edebilir ama bu seviyede de sorumluluk sürücüye aittir. Seviye 3'te araçlar "çevresel algılama" kabiliyetlerine sahiptir ve yavaş hareket eden bir aracın hızlanması gibi kararlar alabilir. Seviye 3'te, sürücü araca tam kontrol verebilir. Sistem hızı ve direksiyonu kontrol ederken yolu sensörleriyle izler ancak sürücünün gerekli durumlarda hazır olarak beklemesi gerekmektedir. Seviye 4'te, sistem tüm seyahat boyunca kontrolü ele alabilir, gerekli durumlarda sürücüden yardım ister fakat yanıt alamasa bile sürüşe devam eder. Seviye 4 araçlar otonom sürüş modunda çalışabilir. Ancak, mevzuat ve altyapı gelişinceye kadar otonom araçlar sınırlı bir alanda (genellikle en yüksek hızların ortalama 30mph'a ulaştığı kentsel bir ortam) içinde kullanılabilmektedir. Bazı firmalar Seviye 4 araç testlerini gerçekleştirmeye başlamıştır.

- Bir Fransız firması olan NAVYA, ABD'de elektrikle çalışan ve 55mph hıza ulaşabilen Seviye 4 araçları üretiyor ve satıyor.

- Alphabet's Waymo geçtiğimiz günlerde Arizona'daki Seviye 4'te sürücüsüz taksi hizmetini tanıttı. Bir yıldan fazla süre ve 10 milyon milden fazla test gerçekleştirmiştir.
- Kanadalı otomotiv tedarikçisi Magna, hem şehir ortamında, hem de otoyolda MAX4 adında Seviye 4 araç geliştirmiştir.
- Sadece birkaç ay önce, Volvo ve Baidu, Çin'de robotaksi pazarına hizmet edecek olan Seviye 4 elektrikli araçları ortaklaşa geliştirmek için stratejik bir ortaklık yaptıklarını açıkladı.

Son olarak Seviye 5'te araçta bir direksiyon olmasa bile sensörlerle yolu takip edebilir ve sürücüye ihtiyaç duymadan yola devam edebilir. Seviye 5 araçlarda direksiyon veya gaz ya da fren pedalları bile olmayacak. Kamuya açık olmayan alanlarda test sürüşleri gerçekleşmektedir (Aldana 2013, Anonim 2013, Blanco ve ark. 2015, Anonim 2019a, Anonim 2019b).

Çizelge 2.1. Otonom Sürüş Seviyeleri

Seviye	Sürücü	Araç	Kontrol
0	Tam kontrol	Kontrol Yok	Sorumluluk sürücüde
1	Sorumluluk sürücüye aittir	Gaz ya da direksiyon kontrolü olabilir	Sorumluluk sürücüde
2	Sorumluluk sürücüye aittir	Gaz ve direksiyon kontrolü olabilir	Sorumluluk sürücüde
3	Gerekli durumlarda sürücü hazır olmalıdır	Tam kontrol olabilir	Sorumluluk araçta
4	Gerekli durumlarda araca yardım edebilir	Sürücü etkileşimi gerektirmez	Sorumluluk araçta
5	Kontrol yok	Tam kontrol	Sorumluluk araçta

Herhangi bir seviyedeki otonom araç, sürüş ortamı hakkında bilgi toplayabilmeli ve bu bilgilere dayanarak kontrol kararları verebilmelidir. Araçlarda direksiyon, gaz ve fren pedalı vardır ve bu bileşenlerle araç kontrol edilebilir. Otonom araç hareket halindeyken radar, LiDAR ve kamera olmak üzere kullanılan üç ana sensör vardır. Her sensör farklı bilgiler sağlar ve bazı dezavantajları da vardır. Sırasıyla radyo dalgaları ve lazerler kullanan Radar ve LiDAR, aracın yakınındaki nesnelerin mesafelerini algılar, ancak nesnelerin görünümü hakkında bilgi vermez. LiDAR, kar ve yağmurdan da etkilenebilir

(Chen 2016). Kamera temel olarak insan gözüyle aynı bilgileri sağlar ve bu nedenle sürüş için uygun olmalıdır. Bununla birlikte, bir görüntüden sürüş için gereken bilgileri çıkarmak çok daha zordur.

Otonom araçların geliştirilmesine yönelik ilk başarılı girişim 1950'lerde başladı. İlk tamamen otonom araçlar 1984'te (Kanade ve ark. 1986, Wallace ve ark. 1985) ve 1987'de (Dickmanns ve Zapp 1987) geliştirildi. DARPA (Defense Advanced Research Projects Agency's) yarışmasında, 2004 ve 2005 yıllarında Grand Challenge etkinliklerinde (Thrun ve ark. 2006, Montemerlo ve ark. 2006) sırasında otonom araçlar alanında önemli atılımlar yapıldı. 2004 yılında yarışa giren 15 araçtan hiçbiri yarışı tamamlamadı. 2005 yılında gerçekleştirilen etkinlikte, bu sefer 23 takımdan 5'i bitiş çizgisine ulaştı. 2007 yılında gerçekleştirilen yarışmada, simüle edilmiş bir şehir ortamında araçların otonom bir şekilde hareket etti. Tamamen otonom hareket eden araçlardan altı tanesi yarışı tamamladı (Buehler ve ark. 2009). Bu yarışmada sürücülerin karmaşık görevlerini, makinelerin bağımsız olarak yerine getirebileceği gösterilmiştir. DARPA'nın yarışmasından sonra otonom araç testleri hız kazanmıştır. Dikkat çeken örnekler arasında 2009'dan 2013 yılına kadar gerçekleşen otonom araç yarışmaları (Xin ve ark. 2014), 2010'da Hyundai tarafından düzenlenen otonom araç yarışması (Cerri ve ark. 2011), 2010'da VisLab kıtalararası otonom araç yarışması (Broggi ve ark. 2012), 2013'te kamuya açık alanlarda otonom araç testleri (Broggi ve ark. 2015) ve Bertha Benz'in otomotiv tarihindeki ilk kara yolculuğunu tamamlamasından 125 yıl sonra, aynı rota tamamen otonom olarak tamamlanmıştır (Ziegler ve ark. 2014) . Aynı zamanda, araştırma hem akademik ortamda hem de endüstride hız kazanmaya devam etmiştir. Google'ın sürücüsüz aracı ve Tesla'nın otopilot sistemi son zamanlarda oldukça dikkat çekmektedir (Anonim 2016a, Anonim 2016b). Waymo, 2009 yılında Google ile sürücüsüz araç projesi başlattı. Bugün, mevcut en gelişmiş sürücüsüz araç üreten şirkettir. Waymo, bilgisayar simülasyonlarında milyarlarca mil, 25 şehirde 5 milyon milden fazla otonom sürüş gerçekleştirmiştir. Şirket ulaştırma sektöründe, sürücüsüz araçlar kullanarak sürücüsüz servis hizmeti vermeyi planlıyor. Tesla, Tesla otopilotlarının ilk versiyonunu 2014 yılında Model S olarak piyasaya sürdü. Otomotiv sektöründe faaliyet gösteren diğer şirketlerden daha genç olsa da, Tesla bugün sürücüsüz araç pazarına liderlik ediyor. Bu günlerde caddelerde test edilen otonom araçlar olsa da, halen bazı zorluklarla karşılaşılmaktadır.

Bojarski ve ark. (2016) aracın önüne yerleştirilen tek bir kameradan alınan görüntüler ile CNN'i eğitmişlerdir. Elde edilen eğitim verileriyle sistem, şerit çizgileri olan veya olmayan yollarda ve otoyollarda trafikte sürüş yapmayı öğrenmiştir. Eğitim için bir NVIDIA DevBox ve Torch 7 kullanılmıştır, ayrıca aracın nereye gideceğini belirlemek için Torch 7 çalıştıran bir NVIDIA DRIVE (PX) otonom çalışan araç bilgisayarı kullanılmıştır. Sistem saniyede 30 kare hızında (FPS) çalışmıştır. Gerçekleştirilen modeldeki eksikliklerini gidermek için Bojarski ve ark. (2017) yoldan elde edilen görüntüler ile direksiyon açılarını tahmin eden PilotNet adında sinir ağı tabanlı bir sistem geliştirmişlerdir. PilotNet sürücü tarafından toplanan görüntülerle eşleşmiş direksiyon açılarıyla eğitilmiştir. Gerçekleştirilen testlerde PilotNet'in yoldaki şerit çizgileri olmadan da çeşitli sürüş koşullarında şeridi başarıyla takip etmiştir. Yiping ve ark. (2018) Venodyle-HDL32E lazer ve kamera kullanarak sürücü davranışlarıyla birlikte yolun kamera görüntüsünü kaydetmişlerdir. Lazer bilgisinin kullanılmasının, sadece video karelerinin kullanımına kıyasla tahmin doğruluğunu büyük ölçüde arttırdığını belirtmişlerdir. Xu ve ark. (2018) otonom araçta 3 boyutlu bir nesne algılama sistemi geliştirmişlerdir. Geliştirdikleri sistemde kamera ve LiDAR kullanmışlardır. Oh ve Kang (2016), araçlarda LiDAR yalnızca görünür mesafesindeki engelleri ölçebildiğinden, çalışmalarında aracın nerede olduğunu bulmak için LiDAR ve sterio kamerayı birlikte kullanmışlardır. Chavez-Garcia ve Aycard (2016) hareketli nesneleri tespit etmek için radar, LiDAR ve kamerayı birlikte kullanmışlardır. Cho ve ark. (2016) gerçek dünyada sürücüsüz araçların yanlarından geçen yayaları ve araçları tespit edebilmeleri için kamera, radar ve LiDAR'ı birlikte kullanmışlardır. Ravankar ve ark. (2018) otonom robot ve araçların virajları daha yumuşak bir şekilde dönmeleri ve bu alanda yaşanan zorluklar üzerine bir inceleme gerçekleştirmişlerdir.

Morgan Stanley, otonom araçların tasarımı ve geliştirilmesi ile ilgili 5 büyük teknolojik sorun tespit etmiştir. İlk ikisi doğrudan LiDAR kullanımı ve kamera kullanımı ile ilgilidir. Sürücüsüz araçlarda radar ve sensör özellikleri etkisiz hale getirildiğinde kar, sis ve yağmurda ne yapılacağının belirlenmesi gerekmektedir. Otonom sürüş sırasında yol kenarlarında gerçekleşen değişimlerde aracın nasıl yönlendirileceğinin bilmesi gerekmektedir. Sürücüsüz araçlarda bu sorunların üstesinden nasıl gelineceği araştırma konusudur (Shanker ve ark. 2013).

Sürücüsüz aracın yolda hareketi sırasında şerit takibi önemli bir konudur. Otonom sürüş sırasında aracın şeridin ortasında kalmasını sağlayacak kontrol sistemleri de araştırma konusu olmuştur. Araştırmacılar "Kamera kullanarak otoyol şeridinin ortasında sürücüsüz bir aracı tutmak için bir kontrol sistemi nasıl tasarlanabilir?" sorusuna cevap aramaktadırlar. Kontrol sistemleri; ısıtma sistemlerini düzenlemek, robotları dengelemek, barajlardaki su seviyelerini dengelemek gibi çeşitli sistemler için kullanılır (Yang ve ark. 2008, Khare ve ark. 2010, Velazquez ve ark. 2011, Zimit ve ark. 2018, Chowdhary ve ark. 2018, Zhou ve ark. 2019, Copot ve ark. 2019, Omijeh ve ark. 2015). PID kontrolörler, istenen değeri hesaplamak için, mevcut ölçülen değer ile istenen değer arasındaki hataya bağlı olarak ağırlıklı oransal (Proportional, P), integral (Integral, I) ve türev (Derivative, D) bileşenlerinin bir kombinasyonunu kullanan dijital kontrol sistemleridir. Sürücüsüz araçlarda hız kontrolü, acil frenleme gibi kritik zaman kontrol sistemlerini kontrol etmek ve şerit takibi için PID kontrolörü kullanılabilmektedir. Zhao ve ark. (2012) uyarlanabilir PID tabanlı kontrol sisteminde testlerini gerçekleştirecekleri 1.6 Tiggo3 SUV araca, iki tane 4 çekirdekli bilgisayar ve bir sensör seti yerleştirmişlerdir. Gerçekleştirdikleri çalışma algılama, navigasyon, karar ve kontrol sistemi olmak üzere dört alt sistem içermektedir. PID kontrol sistemiyle direksiyon açılarını hesaplamışlardır ve aracın hızını sabit tutarak hareket etmesini sağlamışlardır. Hazırladıkları kontrol sistemi bir yıldan fazla bir süre boyunca geliştirilmiş ve 2010 ve 2011 yıllarında Çin'de gerçekleştirilen sürücüsüz araç yarışmasında aracın testini gerçekleştirmişlerdir. Testler sonucu kontrol sistemindeki eksiklikler düzeltilmiş ve aşamalı olarak iyileştirmeler gerçekleştirilmiştir. Testler ve yarışmalar sırasında, yörünge izleme sistemi yüksek kontrol doğruluğu göstermiştir. Uyarlanabilir PID tabanlı araç 60 km/s hızda sürücüsüz olarak sürüş yapabilmektedir. 2011'deki yarışmada 10 km'lik yolculuğu istenen sürede (50 dakika) tamamlayan üç sürücüsüz araçtan biri olmuştur. Jigang ve ark. (2017) motor hız kontrolü için bulanık PID kontrol geliştirmişlerdir. Geleneksel PID denetleyici ile bulanık PID denetleyici sonuçları karşılaştırıldığında bulanık PID kontrol denetleyicisinin daha iyi performans gösterdiğini belirtmişlerdir. Önerilen denetleyicinin motorun tepki hızını arttırabileceği ve sarma önleme yeteneğini güçlendirebileceği sonucuna ulaşmışlardır. Chandni ve ark. (2017) direksiyon açısı, fren ve aracın hızını ayarlamak için PID denetleyici geliştirmişlerdir. Geliştirilen mimari yol takibi, algılama ve kontrol modülü

olmak üzere üç başlığa ayrılmıştır. Algılama modülü olan şerit algılama ve engel algılama modülleri ile navigasyon modüllerinden konum ve yön bilgileri kontrol sinyallerini tetikler. Bu kontrol sinyalleri, sürücüsüz aracın direksiyon, gaz pedalı, fren ve debriyajını kontrol etmek için ilgili motorların çalışmasını sağlar. Tasarlanan kontrol sistemi sürücüsüz araç için uygulanmış ve test edilmiştir. Direksiyon açısı değerleri, fazla gecikme olmadan mikro denetleyiciye gönderilmiş, böylece araç, yolun merkez şeridinden sapma olduğunda her zaman düz ve pürüzsüz iki şeritli bir yolun merkez şeridini izlemiştir. Direksiyon kontrolü hem açık döngü kontrol modunda, hem de kapalı döngü PID kontrol modunda gerçekleştirilmiştir. Tekerlek kodlayıcısından gelen geri bildirim ile kapalı döngü yöntemi, merkez şeridinin sorunsuz izlenmesini sağlamıştır. Alonso ve ark. (2013) trafikte farklı hız ve mesafede ilerleyen bir aracı takip etmek için çevrimiçi ayarlanmış bir PID kontrol cihazı önermiştir. Farklı yollarda on saatlik bir sürede testler gerçekleştirmişlerdir ve gerçekleştirilen denetleyici öndeki aracı salınım yapmadan takip edebilmiştir. Lie ve ark. (2014) araçlarda güvenliği arttırmak için PID kontrol tabanlı araç çarpışma önleme sistemi geliştirmişlerdir. Çevreyi gerçek zamanlı olarak algılayan araçta kullanılan sensörler doğru ve güvenilir sürüş bilgilerini elde etmektedir. Yayalar ve öndeki araçlar sensörler tarafından tespit edildikten sonra, güvenli bir mesafede durmalıdır. Aksi takdirde, tehlike olarak değerlendirildiğinde kontrol edilmesi gerekir. Gerçekleştirilen çalışmada, çarpışmadan kaçınma senaryolarını sadeleştirmişlerdir. İlk deneme aynı şerit üzerine yerleştirilen engeller dikkate alınmıştır, Sonra aracın önüne başka bir araç veya yaya ortaya çıktığında şerit değiştirme yerine otonom fren manevrası yapılmıştır. Jiao ve ark. (2008), direksiyon açısını kontrol etmek için Geliştirilmiş Parçacık Sürü Optimizasyon Algoritması (Improved Particle Swarm Optimization Algorithm, IPSO) temelli kendinden uyarlamalı bir PID kontrol sistemi geliştirmişlerdir. Gerçekleştirilen kontrol sistemi, geleneksel PID kontrol yöntemine kıyasla, daha az salınım gerçekleştirmiştir. Sürücüsüz araçlar, trafik koşulları değişse bile araçları şeritte tutarak istenilen hedefe seyahat etmek üzere tasarlanmıştır. Surendharan ve Jennifer Ranjani (2016) sürücüsüz araçların ulaştırma sistemlerine entegrasyonu için bir yaklaşım geliştirmişlerdir. Geliştirdikleri yöntemde, sürücüsüz araçlar üç alt sistemden oluşmaktadır. Ortamı ve mesafe ölçümlerini algılamak için bir kamera ve ultrasonik sensör, yoldaki nesneleri ayırt etmek için duyusal verileri analiz etme yeteneğine sahip sistem ve bilgiyi yorumlayan ve aracı istenilen yolda tutmak için uygun

kararı veren bir PID kontrol ünitesi kullanılmıştır. Önerilen prototip, giriş ve işlem birimi olarak Raspberry Pi kullanılarak geliştirilmiştir. Kontrol ünitesi olarak düşük maliyetli bir Arduino kartı kullanılmıştır. Kamera, çevrenin gerçek zamanlı görüntüsünü yakalamak için kullanılmıştır. Çekilen video veya görüntüler Raspberry Pi'ye, yani işlem ünitesine gönderilmiştir. PID kontrol cihazına dayalı sürücüsüz aracın performansı analiz edilmiştir. Gerçek mesafe ve ölçülen mesafe arasındaki fark ölçülerek, sistemin %90 başarım oranının olduğu belirtilmiştir. Nie ve ark. (2018), çalışmalarında sürücüsüz aracın hızı ve direksiyon açısını kontrol eden PID kontrol için radyal temelli sinir ağı geliştirmişlerdir. Geliştirilen PID kontrolünün performansı geleneksel yöntemlerden daha iyi performans göstermiştir. Emirler ve ark. (2014) çalışmalarında, sürücüsüz araçlarda direksiyon kontrolü için PID denetimini simülasyon ortamında gerçekleştirmişlerdir. Aracın 15 m/s hızında istenen yörüngeyi başarıyla takip ettiği belirtilmiştir. Han ve ark. (2017) çalışmalarında, şerit kontrolü için direksiyon açısını belirleyen sinir ağı tabanlı PID kontrolü kullanmışlardır. Deneysel simülasyon sonuçlarında, önerilen model ve algoritmanın, şerit takibi kontrolünde gerçek zamanlı olarak ve değişen ortamlarda doğru sonuç verdiği belirtilmiştir.

Literatür incelendiğinde sürücüsüz araçların yollarda kontrollü bir şekilde ilerlemesi için DL algoritmaları ve PID denetimleri kullanılmıştır. Bu tez çalışmasında sadece kamera görüntüleri kullanılarak sürücüsüz aracın sürüş simülatöründe, hem DL tabanlı, hem de PID kontrolü kullanılarak otonom bir şekilde hareket etmesi sağlanmıştır. Sürücüsüz araçlar için kullanılan birçok sensör olmasına rağmen, kamera sürücüsüz araçlarda vazgeçilmez bir sensördür. Kameralar sürücüsüz aracın çevresini görselleştirmesini sağlar. Kameralar yol dokusunun sınıflandırılmasında çok etkilidir ve radar veya LiDAR gibi algılama için kullanılan diğer sensörlerden daha ekonomiktir. Kameraların kısıtı veriler işlenirken gösterdiği performans yani işlem gücüdür. Kamera görüntüleri CNN'in eğitimi ve PID kontrol için kullanılmış ve çıkış olarak direksiyon açısı elde edilmiştir.

Son yıllarda Yapay Zeka (Artificial Intelligence, YZ) ilgi duyulan araştırma konularından biri olmuştur. Kaynak araştırmasında da bahsedildiği gibi ML, DL ve YZ teknoloji yayınlarının dışında sayısız makaleye konu olmuştur. Önümüzdeki günlerde sürücüsüz

araçlar ve sanal asistanlar hayatımızda yerini alacak, ayrıca ekonomik faaliyetlerin çoğu robotlar veya YZ ajanları tarafından gerçekleştirilecektir.

YZ, makinelerin insanların gerçekleştirdikleri görevleri taklit edebilmesi üzerine geliştirilmiştir. Sürücüsüz araçların geliştirilmesinde YZ teknolojilerine ihtiyaç duyulmaktadır (Chollet, 2018). Sürücüsüz araçların otonom hareketleri için direksiyon açılarının belirlenmesi ve otonom sürüş sırasında aracın hızının ayarlanması gibi karar verme veya bu değerlerin tahmin edilmesi süreci ML ile gerçekleşmektedir. Bu tür çözülmesi zor ve karmaşık problemler DL yöntemleri kullanılarak çözülmektedir. YZ, DL ve ML'yi içeren genel bir alandır. Her DL algoritması bir ML algoritmasıdır ve her ML algoritması bir YZ algoritmasıdır, fakat bunun tersi söz konusu değildir. Şekil 2.1'de DL'nin kapsam haritası verilmiştir.

Şekil 2.1. Derin Öğrenme, kapsam haritası (Chollet, 2018).

ML'de programcıların veri işleme kurallarını elle oluşturması yerine, bir bilgisayar verilere bakarak bu kuralları otomatik olarak öğrenebilir. ML ile insanlar verilerden beklenen cevapların yanı sıra veriyi ve kuralları girer. Bu kurallar daha sonra orijinal cevapları üretmek için yeni verilere uygulanabilir. ML matematiksel istatistiklerle ilişkilidir, ancak istatistiklerden farklılık gösterir. İstatistiklerden farklı olarak, ML, Bayesian analizi gibi klasik istatistiksel analizin uygun olamayacağı büyük, karmaşık veri kümeleriyle ilgilenir. Sonuç olarak, ML ve özellikle DL, nispeten az matematik teorisi sergiler ve mühendislik odaklıdır. ML'de sınıflandırma performansını ölçmek için, verilerin ne kadar doğrulukla sınıflandırılabildiği bilgisine ihtiyaç duyulmaktadır. Sınıflandırma işleminde farklı performans değerlendirme ölçütleri kullanılmaktadır.

Çizelge 2.2. Çapraz Tahmin Tablosu

	Tahmin Edilen Sınıf	
Gerçek Sınıf	**Pozitif**	**Negatif**
Pozitif	TP	FN
Negatif	FP	TN

Her bir sınıf için doğru ya da yanlış örneklerinin sayısı bir matris ile gösterilebilmektedir. Bu gösterime Çapraz Tahmin Tablosu (Confusion Matrix) denir. Çizelge 2.2'de Çapraz Tahmin Tablosu gösterilmiştir.

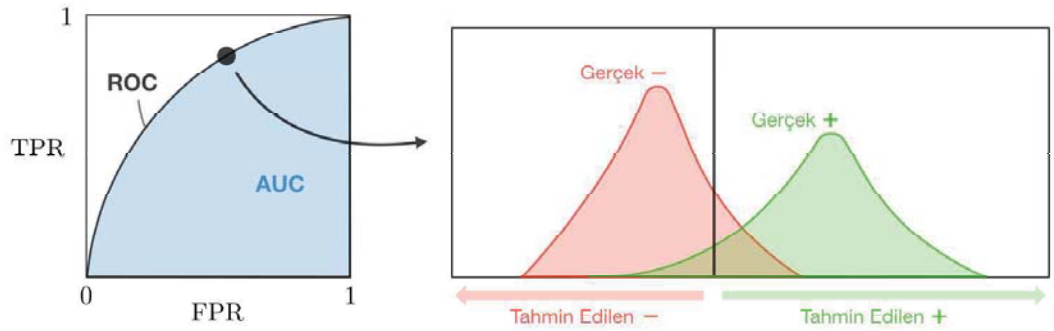
Çizelge 2.3. Sistem Başarım Ölçütleri

Metrik	Formül	Açıklama
Doğruluk	$\frac{\lvert TN\rvert + \lvert TP\rvert}{\lvert FN\rvert + \lvert FP\rvert + \lvert TN\rvert + \lvert TP\rvert}$	Doğru tahminleri gösterir.
Kesinlik	$\frac{\lvert TP\rvert}{\lvert FP\rvert + \lvert TP\rvert}$	Her bir sınıf için, doğru tahminlerin ne kadar kesin olduğunu gösterir.
Belirlilik	$\frac{\lvert TN\rvert}{\lvert FP\rvert + \lvert TN\rvert\rvert}$	Gerçek negatif örneklerinin oranını gösterir.
Geri Çağırma	$\frac{\lvert TP\rvert}{\lvert FN\rvert + \lvert TP\rvert}$	Gerçek pozitif öreklerin oranını gösterir.

- **Yanlış Pozitif (False Positives, FP):** Negatif sınıftan olan verilerin pozitif olarak yanlış tahmin sayısını gösterir.
- **Yanlış Negatif (False Negatives, FN):** Pozitif olarak tahmin edilmesi gereken örneklerin, negatif olarak tahmin edilme sayısını gösterir.
- **Doğru Pozitif (True Positives, TP):** Pozitif tahmin edilmesi gereken örneklerin pozitif olarak tahmin edilme sayısını gösterir.
- **Doğru Negatif (True Negatives, TN):** Negatif sınıfa ait olan örneklerin doğru tahmin edilme sayısını gösterir.

Sınıflandırma modellerinin performanslarını değerlendirmek için Çizelge 2.3'de verilen sistem başarım ölçütleri kullanılır.

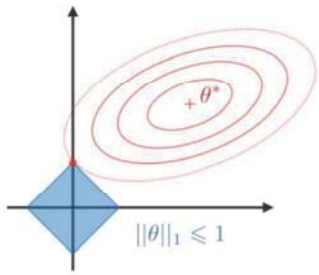
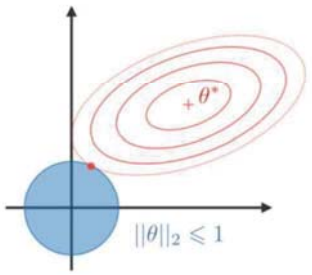
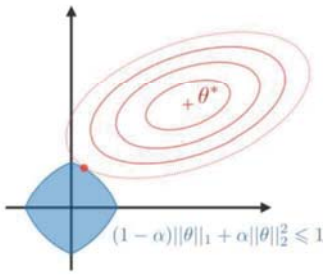
İşlem Karakteristik Eğrisi: İşlem Karakteristik Eğrisi (Receiver Operating Curve, ROC), eşik değeri değiştirilerek Doğru Pozitif Oranı-Yanlış Pozitif Oranı grafiğidir. Şekil 2.2'de İşlem Karakteristik Eğrisi grafiği gösterilmektedir.



Şekil 2.2. İşlem Karakteristik Eğrisi

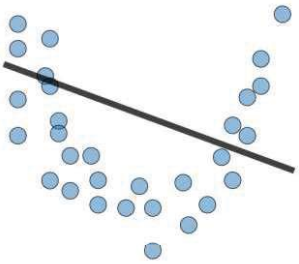
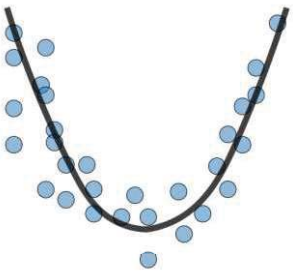
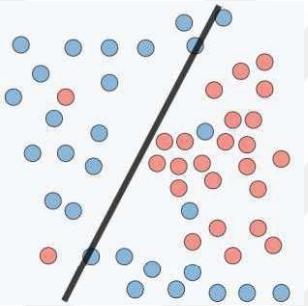
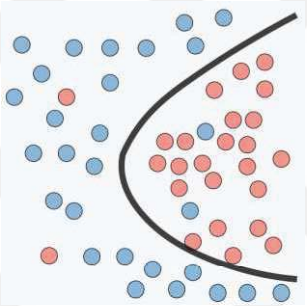
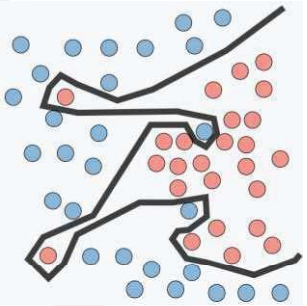
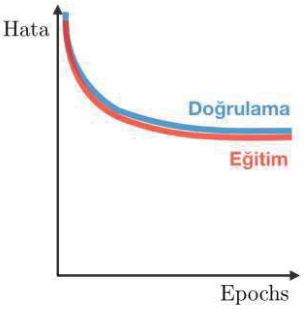
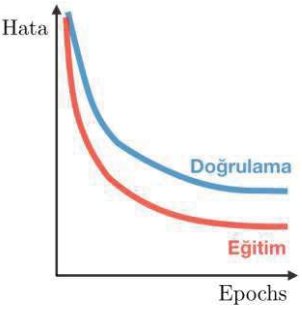
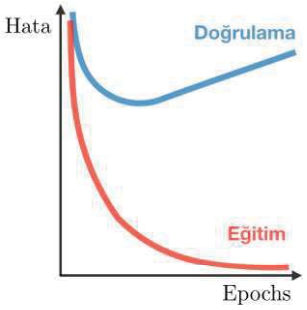
ML'de model seçimi yaparken verilerimizi eğitim veri seti, doğrulama veri seti ve test veri seti olmak üzere üçe ayırırız. Eğitim veri seti verilerimizin %70'i olabilir, doğrulama veri setimiz de genelde verilerimizin %30'u kadar belirlenebilir. Test veri seti ise modelin hiç görmediği verilerden oluşur. Verileri aşırı öğrenmesinden kaçınmak ve dolayısıyla yüksek varyans sorunları ile karşılaşmamak için düzenlileştirme (regularization) yapılır.

Çizelge 2.4. Değişken Seçiminde Kullanılan Bazı Yöntemler (Amidi ve Amidi, 2018)

Lasso	**Ridge**	**Elastic Net**
Değişkenler 0'a kadar küçültüldüğünde, değişken seçimi iyi	Katsayılar daha küçük ayarlandığında	Seçilen değişkenle küçük katsayılar arasındaki çelişki
		
$\ldots + \lambda\|\|\theta\|\|_1$ $\lambda \in R$	$\ldots + \lambda\|\|\theta\|\|_2^2$ $\lambda \in R$	$\ldots + \lambda[(1-\alpha)\|\|\theta\|\|_1 + \alpha\|\|\theta\|\|_2^2]$ $\lambda \in R, \alpha \in [0,1]$

Çizelge 2.4'de, tahmin doğruluğunu ve yorumlanabilirliği arttırmak için yaygın olarak kullanılan düzenlileştirme tekniklerinin farklı türleri verilmektedir. Oluşturacağımız modelde eksik özellikler ya da karmaşıklıklar olabilir. Özelliklerin tam olarak yakalanamaması model yanlılığına neden olabilir. Çizelge 2.5'te yüksek önyargı, doğru öğrenme ve aşırı uyum durumlarının belirtileri gösterilmiştir.

Çizelge 2.5. Yüksek Önyargı, Doğru Öğrenme, Aşırı Uyum

	Önyargı	**Doğu Öğrenme**	**Aşırı Uyum**
Belirtiler	Yüksek önyargı durumunda yüksek eğitim hatası	Eğitim hatasından daha düşük eğitim hatası	Yüksek varyans olduğu durumda, çok düşük eğitim hatası
Regresyon Grafiği			
Sınıflandırma Grafiği			
Derin Öğrenme Grafiği			
Olası Yöntemler	Daha fazla veri kullanarak, model karmaşıklaştığında model daha fazla eğitilebilir.		Daha fazla bilgi elde edip düzenlileştirme gerçekleştirilebilir.

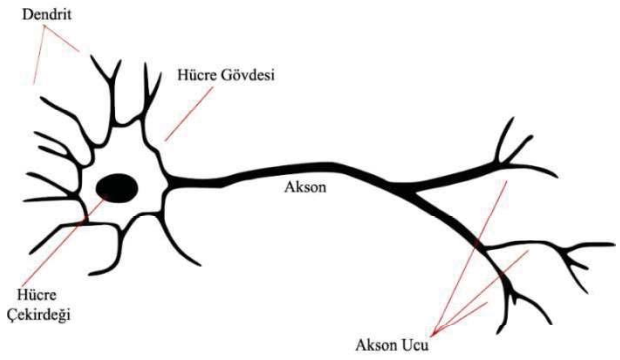
Oluşturulan modellerin tahmin sağlamak için iyi dengelenmiş ve büyük bir örnek veri seti gerekir. Sistem verileri aldıktan sonra ilişkili modeli öğrenir ve model kötü performans gösteriyorsa modeli iyileştirmek gerekmektedir. Modeli iyileştirmek için birkaç yöntem deneyebiliriz. İlk olarak daha çok veri ile eğitim gerçekleştirebiliriz, ama bazen bu durumda da model performansını yükseltemeyebiliriz. Gürültülü özellikleri kaldırıp modeli geliştirebiliriz ya da yeni özellikler ekleyebiliriz. Bir diğer yöntem ise

düzenlileştirmeyi kullanabiliriz. Bazı durumlarda, çok karmaşık ve birçok özellik içeren model kullanmak aşırı uyum sorununa yol açabilir. Model eğitim kümesindeki örnekleri öğrenip çok iyi performans gösterebilir, fakat yeni durumlarda bir tahminde bulunmazsa aşırı uyum gerçekleşmiştir.

Kullandığımız model çok basit olduğunda gerçek verileri iyi bir şekilde temsil edemez ve önyargı durumu oluşur. Bir modelin önyargısı, beklenen tahmin ve verilen veri noktaları için tahmin etmeye çalıştığımız doğru model arasındaki farktır. Kullanılan model çok fazla karmaşık ve esnek ise varyans durumu oluşabilir. Belirli veri noktaları için model tahmininin değişkenliği modelin varyansı olarak belirtilir. Yüksek önyargı durumunda model çok basit olur, varyans yüksek olduğunda model çok karmaşık bir durumda olur. Modelde en iyi durumu yakalayıp, modeli bu şekilde geliştirildiğinde bu durumdan kurtulabilir.

2.1. Yapay Sinir Ağları (Artificial Neural Networks)

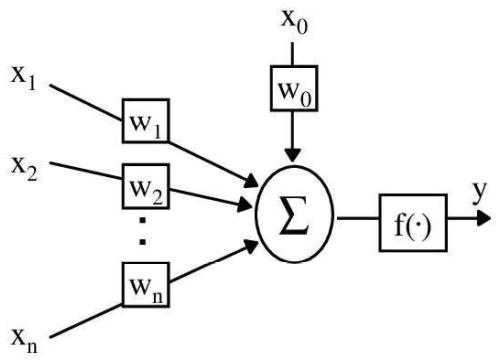
Yapay zeka için sinir ağları, insan beynindeki doğal yapılardan ilham alan ve modern bilgisayarlarda uygulanan matematiksel modellerdir. Tipik uygulamalar, modelleri karmaşık fonksiyon tahmin ediciler olarak kullanmaktır. Yapay Sinir Ağları (Artificial Neural Network, YSA) genelleme yapabilme özellikleri sayesinde esnek ve güçlü çözümler üretebilirler.



Şekil 2.3. Biyolojik Sinir sistemi

YSA bu özellikleri sayesinde karşılaştıkları sorunları farklı örnekler ile öğrenebilme yeteneğine sahiptir. YSA insanların öğrenme, anlama ve yorumlama gibi özelliklerini taklit ederek problem sorunlarını çözebilmektedir. YSA, biyolojik sinir sistemindeki gibi,

gerçek hayatta nesnelerin etkileşimini ve birbirleri arasındaki bağlantıyı ağ yapısıyla tanımlar. YSA sistemi girdiler, ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu ve çıktılardan oluşmaktadır. Şekil 2.3'de biyolojik sinir sisteminde görüldüğü gibi nöronlar işlemci elemanları, sinapslar ağırlıkları, dendritler toplama fonksiyonunu, hücre gövdesi aktivasyon fonksiyonunu ve aksonlar nöronların çıkışını ifade etmektedir. Giriş katmanı dışarıdan gelen verileri işler. Girişi katmanı problemlere etki eden parametrelerden oluşur. Toplama fonksiyonu hücrenin net girişini hesaplamak için hücreye gelen girişlerle, bunların ağırlıklarının çarpımını toplar. Net giriş değeri aktivasyon fonksiyonundan geçtiğinde net çıkış değeri hesaplanır.



Şekil 2.4. Yapay Sinir Ağı Modeli

Her girişin ağırlık değeri vardır. YSA, örnekler gösterildikçe ağırlık değerlerini günceller. YSA'da hedef, giriş verileri için en uygun çıkış değerini bulmaktır. Şekil 2.4'te YSA modeli gösterilmektedir.

$$NET = \sum_{i=1}^{n} x_i w_i \tag{2.1}$$

$$v = f\left(\sum_{i=1}^{n} x_i w_i + b)\right) \tag{2.2}$$

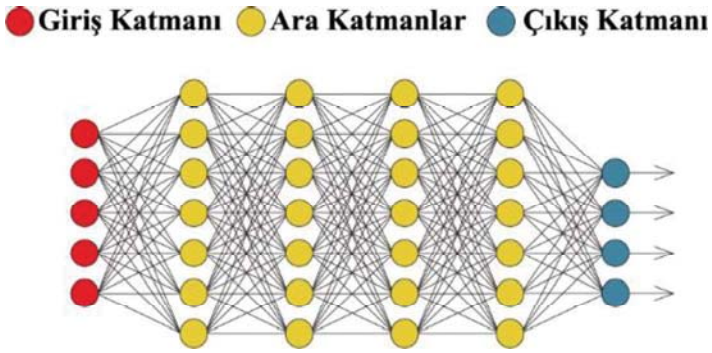
Giriş verileri ağırlık değerleriyle çarpıldıktan sonra, hem sisteme doğrusal olamayan karmaşıklıklar katmak, hem de üretilecek değerin daha anlamlı olmasını sağlamak için aktivasyon fonksiyonundan geçirilerek çıkışa gönderilir. Tercih edilecek aktivasyon fonksiyonları öğrenme ve başarım performansı bakımından farklılık göstermektedir.

Çizelge 2.6. Aktivasyon Fonksiyonlarının Matematiksel İfadeleri

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	$f(x) = 2x\sigma(\beta x) = \begin{cases} \beta = 0 & için \quad f(x) = x \\ \beta \to \infty\ için & f(x) = 2\max(0, x) \end{cases}$	$(-\infty, \infty)$

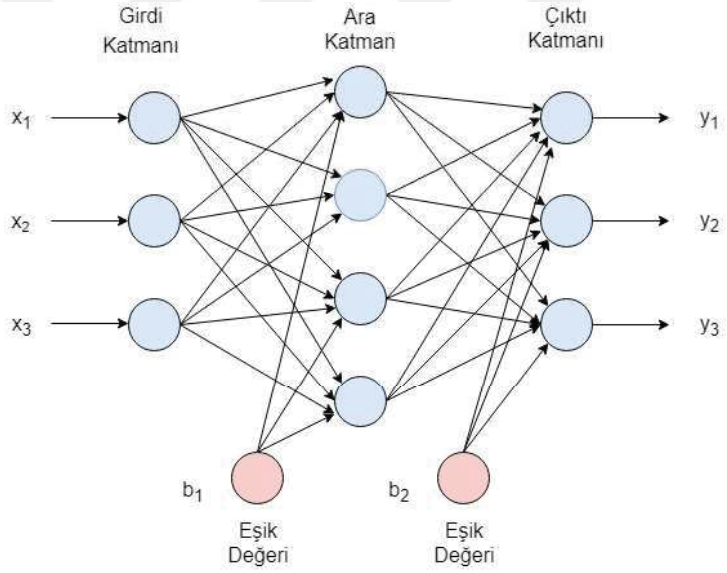
Aktivasyon fonksiyonunun doğru tercih edilmesi sistemin başarımının daha iyi olmasını sağlayacaktır. Çizelge 2.6 ve Çizelge 2.7'de yaygın olarak kullanılan aktivasyon fonksiyonları verilmiştir. Ağın kolay ve hızlı yakınsaması aktivasyon fonksiyonlarının seçiminde ilk kriter olabilir. Çok katmanlı bir ağ kullanılacaksa ReLU hız bakımından diğer fonksiyonlara göre avantajlı olacaktır. ReLU genellikle ara katmanlarda kullanılır.. y değerleri x'teki değişikliklere çok az tepki verdiğinde türev sıfıra yakınsar ve gradyan ölmesi yaşanır. Gradyanların ölmesi problemine ilk çözüm Leaky ReLU olabilir.

Yapay Sinir Ağlarının bir ağ yapısı oluşturabilmesi için birçok düğümün bir araya gelmesi gerekmektedir. Şekil 2.5 örnek YSA modelinde giriş, çıkış ve ara katmanlar gösterilmektedir. Giriş vektörü x_i ve ağırlık matrisi w_i olmak üzere, ara katman için üretilecek değer $y_i = f(\sum_i x_i w_i + b)$ şeklinde yazılır. Ağın düğümlerinde ağırlıkların güncellenmesi için $bias$ (b) değeri kullanılır. $bias$ (eşik) değeri kullanıldığında her giriş vektörü x_i ve w_i ağırlık matrisi için ara katmanlarda üretilen değer $y_i = f(\sum_i x_i w_i + b)$ şeklinde hesaplanır.



Şekil 2.5. Çok Katmanlı Yapay Sinir Ağı Modeli

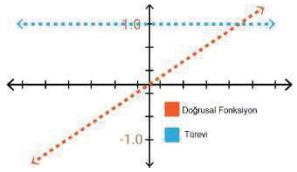
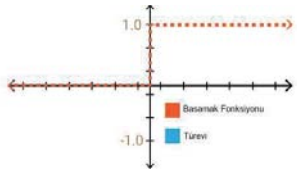
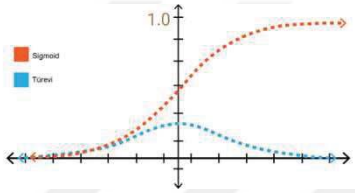
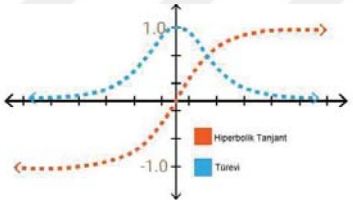
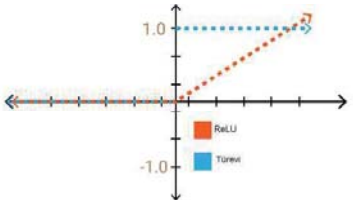
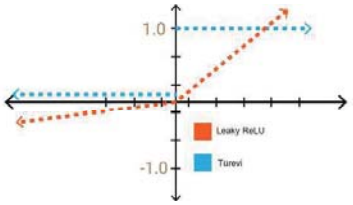
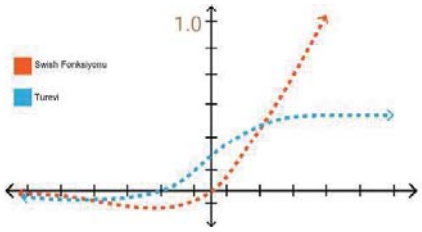
YSA, ileri ve geri beslemeli olmak üzere ikiye ayrılır. İleri beslemeli ağlarda önceki düğünden gelen sinyal sonraki düğüme aktarılır, yani giriş katmanından gelen sinyal ara katmana, ara katmanlardan geçen sinyal çıkış katmanına iletilir. Geri beslemeli ağlarda ise bu durum hem ileriye doğru, hem de geriye doğru olur.



Şekil 2.6. YSA Modeli Eşik Değeri ile Gösterimi

Ağ eğitilirken her düğüm için ağırlıklar güncellenir. YSA kendisine gösterilen her bir örnek için ağırlıklarını günceller. Bu işlem ağa gösterilen tüm örnekler için doğru sonucun üretilmesine kadar devam eder. Katman sayısını arttırmak karmaşık modellerde doğruluk oranını arttırabilir. Sinir ağlarında, çapraz entropi kaybı $L(z, y)$ olarak tanımlanır. Çapraz entropi kaybı veya log kaybı, çıktısı 0 ile 1 arasında bir olasılık değeri olan bir sınıflandırma modelinin performansını ölçer. Öngörülen olasılık, gerçek etiketten ayrıldıkça, çapraz entropi kaybı artar. Mükemmel bir model 0 log kaybı olur. Denklem 2.3'teki gibi formülize edilir.

Çizelge 2.7. Aktivasyon Fonksiyonları (Kızrak 2019a)

Aktivasyon Fonksiyonu	Görsel	Açıklama
Doğrusal Fonksiyon		Türev sabit olduğu için öğrenme gerçekleşmez. Giriş katmanı ile çıkış katmanı arasında hep aynı doğrusal sonuca ulaşılır.
Basamak Fonksiyonu		İkili değer alır ve ikili sınıflayıcı olarak kullanılır. Gizli katmanlarda türevi öğrenme değeri temsil etmediği için kullanılmaz, genellikle çıkış katmanlarında tercih edilir.
Sigmoid Fonksiyonu		(0, 1) aralığında değer üretir. Türevlenebilir olduğundan öğrenme gerçekleşir. En sık kullanılan aktivasyon fonksiyonudur.
Hiperbolik Tanjant Fonksiyonu		(-1, +1) aralığında değer üretir. Türevinin daha dik ve daha çok değer alabilir. Daha hızlı öğrenme ve sınıflama işlemi için daha geniş aralığa sahip olmasından dolayı daha verimlidir.
ReLU		[0, ∞) aralığında değer üretir. Doğrusal değildir ve iyi bir tahmin edicidir. Negatif eksende 0 değerlerini aldığından ağ daha hızlı çalışır. Çok katmanlı ağlarda tercih edilir.
Leaky (Sızıntı) ReLU		Tanım aralığı eksi sonsuza doğru devam etmektedir. Öğrenme negatif bölgelerde de gerçekleşir.
Swish Fonksiyonu		Negatif bölgede aldığı değerler doğrusal değildir. Monoton bir fonksiyondur. Yumuşak bir enterpolasyon sağlar.

Denklem 2.3'te çoklu sınıflandırmaya yönelik bir gözlemde entropi kaybı hesaplanmasına yönelik formül verilmiştir. Burada p öngörülen olasılık değerini ve y ise pozitif ve negatif sınıf değerini gösteren göstergedir. Pozitif sınıflar için $y = 1$, negatif sınıflar için $y = 0$ değerini alır.

$$L(p, y) = -[y\log(p) + (1 - y)\log(1 - p)] \quad (2.3)$$

Öğrenme: YSA'ların öğrenebilme özelliği insanların öğrenebilme kabiliyetleriyle yakından ilgilidir. Gerçekleştirilen bir eylemin belirli bir görevde daha iyi anlaşılması için eğitime ihtiyaç duyulmaktadır ve insanlar bu görevi gerçekleştirmek için kendilerini geliştirmektedir. Benzer şekilde, sinir ağları, giriş sinyaline cevap olarak neyin üretilmesi gerektiğini tanımlamak için eğitime ihtiyaç duyar. Gerçek değer ile ağ tarafından üretilen değer arasındaki farka bağlı olarak hata değeri hesaplanır ve bu hata değeri sistemden geri gönderilir. Ağın tüm katmanları için hata değeri analiz edilir ve bir sonraki giriş sinyali için eşik değeri ve ağırlık matrisini ayarlamak için kullanılır. Bu şekilde sinir ağı, değerleri nasıl analiz edeceğini öğrenmiş olur

Öğrenme oranı, sıklıkla α veya η olarak belirtilir. Öğrenme oranı ağırlıkların hangi performansta güncellendiğini gösterir. Bu derece uyarlamalı olarak değişebilir veya sabit olabilir. "Adam" olarak adlandırılan yöntem ile öğrenme oranı ayarlanabilir.

Geri Yayılım Algoritması: Geri yayılım, kayıp olarak verilen $J(\theta)$ gradyan hesaplanması için bir algoritmadır. YSA eğitim aşamasından sonra, ağın çoklu katmanlarından geçmek ve tüm giriş sinyallerindeki hata değerini en aza indirmek için ağın parametrelerini ve eşik değerini ayarlar. Gerçekleştirilen bu adımlar geri bildirim olarak bilinir ve hata değeri minimum oluncaya kadar işlem ağ üzerinden sürekli olarak uygulanır. Sonrasında, sinir ağı böyle bir eğitim sürecine ihtiyaç duymaz ve güncelleme yapmadan sistemin işleyişine izin verilir. YSA, güncellenmiş ağırlıkları ve eşik değerlerini kılavuz olarak kullanır. Geri yayılım algoritması sinir ağındaki ağırlıkları güncellemek için kullanılır. Güncelleme işlemi gerçekleşirken asıl sonuç ile gerçekleştirilmesi istenilen sonucu hesaba katan bir yöntemdir. Ağırlık w değerine göre Denklem 2.4'deki gibi türev, zincir kuralı kullanılarak hesaplanır. Ağırlık güncellemesi Denklem 2.5'deki gibi hesaplanır. Sinir ağında ağırlıklar güncellenirken ilk olarak bir

eğitim verisi kümesi alınır. Eğitim kümesi alındıktan sonra, denk gelen kaybı elde etmek için, ileri yayılım gerçekleştirilir. Sonra gradyanları elde etmek için kayba geri yayılım uygulanır. Son olarak sinir ağının ağırlıklarını güncellemek için gradyanlar kullanılır.

$$\frac{\partial L(p,y)}{\partial w} = \frac{\partial L(p,y)}{\partial a} \times \frac{\partial a}{\partial p} \times \frac{\partial p}{\partial w} \tag{2.4}$$

$$w \leftarrow w - \alpha \frac{\partial L(w,b)}{\partial b} \tag{2.5}$$

Denklem 2.4 ve Denklem 2.5'te ∂ kısmi türevleri sıradan tek değişkenli türevlerden ayırmak için kullanılır. Burada maliyeti minimize etmek için en uygun w ve b değerlerini bulmaya çalışıyoruz. Gradyan inişi metodu, çok güçlü ve çok genel optimizasyon metodudur. Burada α öğrenme katsayısını ifade etmektedir. $L(p,y)$ kayıp değerini ifade etmektedir.

Eğitim Optimizasyonu ve Seyreltme: Bir modeli öğrenmek veya eğitmek, optimizasyon olarak bilinir. Hata değeri minimum oluncaya kadar, parametreler örnek gradyan yönünde güncellenir. Güncellenen ağırlıklar ile sisteme hiç gösterilmeyen örnekler ile sistem test edilir. Test aşamasındaki başarım oranı sistemin başarısını göstermektedir. Sinir ağı eğitim aşamasında verilen örnekleri yüzde yüz başarı ile doğru bilip, farklı örnekler üzerinde tahmin yapamadığında aşırı uyum gerçekleşir. Sinir ağlarında aşırı uyumu gidermek için seyreltme (dropout) yöntemi uygulanır. Seyreltme işleminde sinir ağı içerisindeki bazı bağlantılar rasgele olarak hesaplaya katılmaz ve bu şekilde değer üretilir.

2.2. Derin Öğrenme (Deep Learning)

Derin Öğrenme ML'nin alt dalıdır. Doğrusal yapıda olan geleneksel ML algoritmalarının yanında DL algoritmalarında uygulanacak olan yapının karmaşıklığına göre değişen bir hiyerarşi vardır. Başarı oranı belli bir seviyeye ulaşıncaya kadar DL süreci devam eder. DL sürecinde kullanılacak verinin DL'ye uygun olup olmadığı belirlenir. Eğer veri uygunsa veri kümesi tanımlanır ve analiz için hazırlanır. Veriler hazırlandıktan sonra uygulanacak DL algoritması seçilir ve tanımlı veriler ile kullanılan algoritma eğitilir. Eğitim süreci bittikten sonra, eğitilen modelin tanımsız veriler ile performansı test edilir.

1950'li yıllardan NN çalışmaları başlamıştır. İnternetin gelişimiyle birlikte, veri miktarı artmış ve artan veriyle birlikte sistemlerin öğrenme, karar verme, akıl yürütme gibi işlevleri gerçekleştirmesi için standart YSA modellerinin yetersiz kalması farklı bakış açılarını beraberinde getirmiştir. LeCun ve ark. (1998) tarafından çevrimiçi el yazısı tanıma için LeNet mimarisi ile CNN kullanmışlardır. Geri yayılım (back propagation) algoritması ile eğitilmiş çok katmanlı sinir ağlarının, başarılı bir gradyan tabanlı öğrenme tekniği olduğu belirtilmiştir. Hinton ve Salakhutdinov (2006), yüksek boyutlu verileri, yüksek boyutlu giriş vektörlerini yeniden oluşturmak için küçük bir merkezi katmanla, çok katmanlı bir sinir ağını eğiterek düşük boyutlu kodlara dönüştürülebilir olduğunu belirtmişlerdir. Bu çalışmadan sonra DL popülerleşmiştir. 2000'li yılların başında popüler olmaya başlayan YSA, bilgisayar işlemcileri dışında, daha fazla işlem gücü gerektiren yapılar için grafik işlemcilerin de kullanılmasıyla birlikte sığ alanlardan derin ağ yapılarına geçiş yapmıştır. Bu yaklaşımla birlikte görüntü işleme uygulamalarından, doğal dil işleme uygulamalarına ve medikal uygulamalar gibi karmaşık problemlerin çözülmesi için farklı alanlarda kullanılmaya başlanmıştır.

Verilerden öğrenebilen yeni bir yaklaşım olan DL, çok katmanlı veri temsillerini öğrenmesi üzerine odaklanmıştır. Derin öğrenmenin derinliği, yaklaşımın başardığı hiçbir derinlemesine anlayışa referans değildir, bunun yerine ardışık temsil katmanları fikrini ifade eder. Bir veri modelinde kaç katmanın olduğu, modelin derinliği olarak adlandırılır. Modern DL genellikle onlarca, hatta yüzlerce ardışık katmanı içerir ve hepsi eğitim gerçekleştiğinde öğrenir. DL'de her ardışık katman, kendinden önceki katmandaki çıktıyı giriş verisi olarak alır. Bu arada, makine öğrenmesine yönelik diğer yaklaşımlar, bir veya iki katmandan oluşan veri temsillerini öğrenir.

Derin öğrenmede katmanlı temsiller, NN ile birbiri üzerine yığılmış katman yapılarıyla öğrenmeyi gerçekleştirir. Sinir ağı terimi, nörobiyolojiye gönderme yapmaktadır, ancak DL'deki kavramların bir kısmı, insan beyninin işleyişinden ilham alarak kısmen geliştirilse de, DL modelleri beyin modelleri değildir. İnsan beyninin, DL modellerinde kullanılan öğrenme mekanizmaları gibi bir şey uyguladığına dair hiçbir kanıt yoktur. Biyolojik sinir sistemindeki nöronların birbirleri arasında ilişki kurması gibi YSA

sistemlerinde de nöron olarak tanımlanan yapılar birbirleri ile bağlantılı şekilde modellenmiştir. YSA algoritmalarının bu şekilde öğrenmeyi gerçekleştireceği ve veriler arasındaki ilişkiyi ortaya çıkarma kapasitesine sahip olacağı düşünülmüştür. DL karmaşık veri temsillerini öğrenmek için matematiksel bir çerçevedir (Chollet, 2018). Bir görüntü için temsil denildiğinde, görüntünün bir pikseli başına yoğunluk değerleri, kenar kümeleri gibi özellikler düşünülebilir. Bu özelliklerin içinden bazıları veriyi daha iyi temsil etmektedir.

DL'de katmanlar bütün olarak algıladığımız fotoğrafın en küçük anlamlı parçasından, tam olarak fotoğrafa dönüşene kadar her aşamasını içeren temsili varlıklardır. DL algoritmaları, giriş olarak verilen çok sayıda verinin ayırt edici özelliklerini kendisi öğrenir. Sistem, giriş olarak verilen özellikleri başarılı bir şekilde öğrenebilmesi için yeteri kadar eğitilmelidir. Giriş olarak verilen verilerin özelliklerinin öğrenme aşaması katman temsillerinden oluşmaktadır. Alt seviye katmanlardaki özelliklerin ayıt ediciliği daha azdır. Alt seviyedeki katman temsillerinin bir araya gelmesiyle oluşan bir üst seviyedeki katmanlarda bulunan özellikler daha fazla ayırt ediciliğe sahiptir. Alt seviye katmanlardaki özellikler daha anlamlı sonuçlar üretilebilmesi için, üst seviye katmanlara temel oluşturur. Bu şekilde oluşan yöntem, geleneksel ML algoritmalarının öğrenme yöntemlerinden farklıdır. Geleneksel ML algoritmalarında belirlenen özelliklerin eğitim aşaması gerçekleşmeden önce, bir insan tarafından hesaplanması gerekir. Hesaplanan bu özelliklere göre öğrenme işlemi gerçekleştirilir. Geleneksel ML algoritmaları insan bağımlı özellikler ile çalışabilirken, DL insan bağımsız özellikler ile çalışmaktadır. DL algoritmaları verinin ayırt edici özelliklerini kendi kendine öğrenmektedir. DL algoritmalarının bu yeteneği, algoritmanın başarısı açısından çok önemli etkendir.

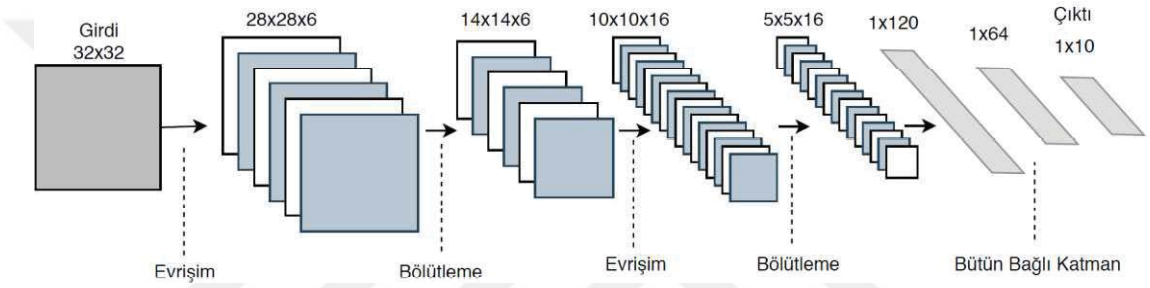
Sürücüsüz araç, bir insan etkileşimi olmadan kendisini yönlendirebilen bir araçtır. Kamera, radar ve LiDAR gibi pek çok bileşen kullanarak sürücüsüz araç oluşturabiliriz. Hesaplama yeteneklerindeki artışla, çevresini algılayabilen, yolun durumuna göre karar verebilen ve çok önemli ayrıntıları öğrenebilen karmaşık yapıları eğitebiliriz. DL, otonom sürüşü mümkün kılmanın yollarından biridir. Sürücüsüz araçların tamamen otonom bir şekilde yönlendirilmesi, yani insan etkileşimi olamadan hareket etmesi zaman alabilir,

fakat önümüzdeki günlerde sürücüsüz araçlar tamamen hayatımıza girmeye başlaması kaçınılmaz olacaktır.

2.3. Evrişimsel Sinir Ağları (Convolutional Neural Networks)

Çok fazla parametreyi hesaplamak için veriden farklı bilgiler çıkarmak gerekmektedir. Görüntü sınıflandırma, nesne tespiti, nesne takibi, doğal dil işleme, stil transferi gibi problemlerin çözümü için yapay sinir ağı tabanlı sistemlere ihtiyaç duyulmaktadır. Artan veri miktarı ve bu verilerden daha anlamlı bilgilere ulaşmak için, öznitelik kestirimleriyle ilgili optimizasyon yapmak gerekmektedir. Klasik YSA modeliyle nöronlarla katmanlar arasındaki bağlantılar ve öğrenilen parametreler çok büyük hesaplama zorlukları ortaya çıkarmaktadır. Evrişimsel Sinir Ağları (Convolutional Neural Networks, CNN), YSA'da bulunan kayıp fonksiyonu, güncellenebilen ağırlık matrisleri ve türevlenebilir skor fonksiyonu gibi parametrelere sahiptir. Görüntü sınıflandırma işlemlerinde CNN, YSA'ya göre daha başarılıdır. Sıradan sinir ağında girdi verisi olarak bir vektör verilir. Bu giriş vektörü ağırlıklar çarpılarak ve aktivasyon fonksiyonundan geçirilerek gizli katmana aktarılır. Gizli katmanlarda da ağırlık güncelleme işlemleri uygulandıktan sonra çıkış katmanında sonuç üretilir. Yapay sinir ağlarında her katman birbirine bağlıdır. YSA'da bir girişin görüntü olduğunu düşünürsek, bütün düğümler birbirlerine bağlı olduğu için, küçük boyutlu bir görüntüde bile çok sayıda parametre çarpımıyla karşı karşıya kalınmış olur ve işlem maliyeti artar. Bu durum aşırı uyuma neden olabilir. CNN bu tarz durumlarda büyük avantaj sağlamaktadır. CNN'de her bir katman üç boyutlu olarak tanımlanmıştır ve kendisinden önceki katmanın sadece küçük bir kısmına bağlanmıştır. Böylelikle hesaplaması gereken parametre sayısı azalmıştır. CNN'de her bir katman üç boyutlu giriş ve çıkış üretmektedir. CNN sınıflandırma problemleri için, içerisinde sınıflandırma değeri olan çıkış üretmektedir. Basit bir CNN modeli evrişim (Convolution), Ortaklama (Pooling) ve Bütün Bağlı (Fully Connected) katmanlardan oluşmaktadır. CNN'de her bir katman kendinden önceki katmanın çıkış değerini giriş değeri olarak alır. Bahsedilen üç katman CNN modelinin mimarisini oluşturmaktadır. $36 \times 36 \times 3$ boyutunda giriş, evrişim, ReLU, örnekleme ve bütün bağlı olmak üzere 5 katmandan oluşan bir CNN mimarisi düşünelim. Giriş resmimiz 3 kanaldan oluşmaktadır, eni ve boyu 36 olacaktır. Girişin yerel bölgesine bağlı düğümleri evrişim katmanı hesaplar. Giriş görüntüsünün küçük bir bölgesindeki değerlerin ve düğümlerin

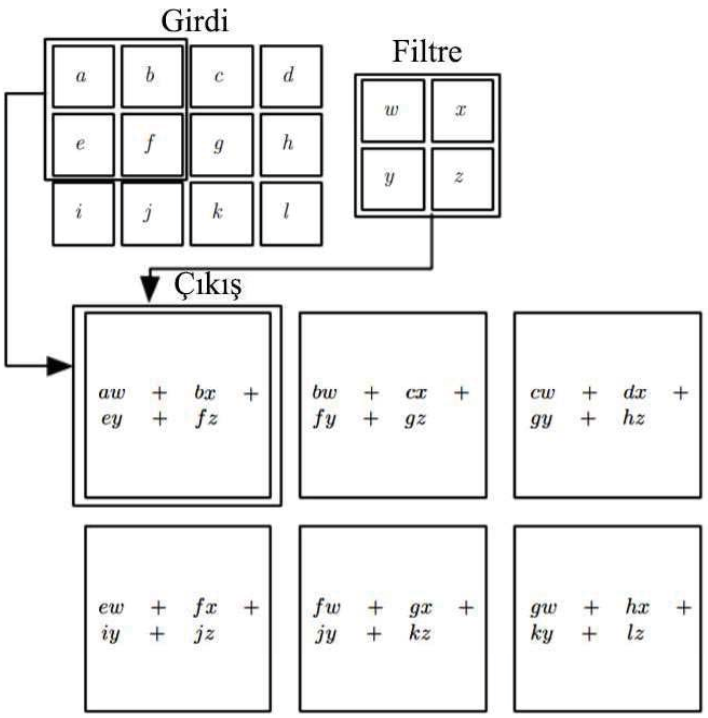
ağırlıklarının noktasal çarpımıyla her bir işlem gerçekleşir. 10 adet filtre kullanıldığı düşünüldüğünde evrişim katmanından sonra çıkışın boyutu $36 \times 36 \times 10$ olur. Aktivasyon fonksiyonu, ReLU katmanında uygulanır. Boyut bu katman sonrasında değişmez. Boyut düşürme işlemi örnekleme katmanında yapılır. Daha da az hacme sahip $18 \times 18 \times 10$ gibi yarıya inmiş bir çıktı elde edilir. Sınıf sayısı kadar hacme sahip bir vektör bütün bağlı katmanda üretilir. Sınıf sayısının 5 olduğunu düşünüldüğünde, her bir sınıf değerinin olası skor değerlerini $1 \times 1 \times 5$ boyutunda olan tek bir vektör barındırır. Parametreler gradyan iniş yöntemine göre güncellenir. CNN mimarisinin sonunda tüm görüntü tek bir sınıf skor vektörüne indirgenir.



Şekil 2.7. LeNet mimarisi (LeCun ve ark. 1998)

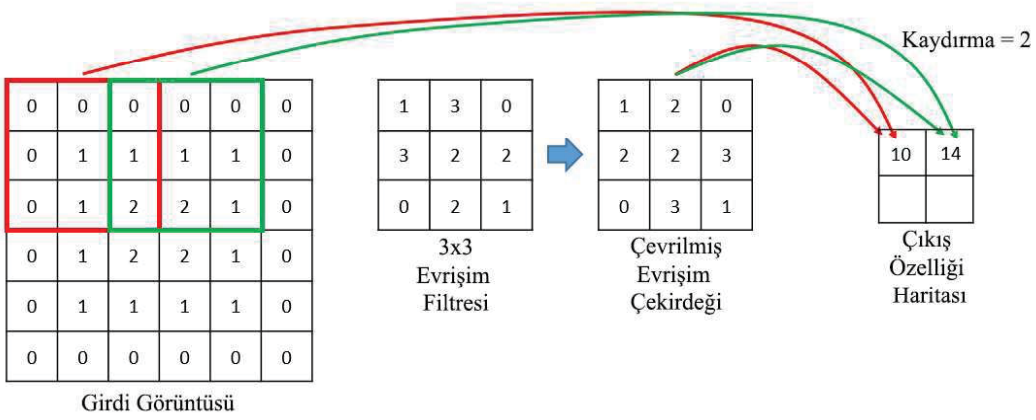
Şekil 2.7'de 32×32 boyutunda tek kanallı bir giriş görüntüsü verilmiştir. CNN mimarisi her katmanı farklılaştırılabilir bir işlevle, bir aktivasyon hacmini diğerine dönüştürür. Verilen LeNet mimarisinde 6 adet filtre kullanılmış ve giriş katmanından sonraki katmanın boyutu $28 \times 28 \times 6$ olmuştur. Bir sonraki katmanda boyut yarıya indirilmiş ve $14 \times 14 \times 6$ olmuştur. En son katmanda 1×10'luk bir çıkış elde edilmiştir.

Evrişim (Convolution): Evrişim katmanında bir takım parametreler öğrenilebilir filtreler bulunmaktadır. Her filtre giriş vektörüyle aynı en, boya ve derinliğe sahiptir. Üç kanallı bir görüntüye 3×3'lük bir filtre uygulanmak istendiğinde evrişim katmanı $3 \times 3 \times 3$ olabilir. Filtre boyutu değerleri değişebilir, fakat derinlik değeri aynı olmak zorundadır. Skor fonksiyonunun hesaplanırken filtre giriş görüntüsü üzerinde kaydırılarak ilerler ve filtre değeri ile giriş görüntüsüne karşılık gelen o anki pozisyonundaki değerler noktasal olarak çarpılır. Bu işlem tüm görüntü matrisi için gerçekleştiğinde 2 boyutlu bir özellik haritası elde edilmiş olur. Bu işleme çapraz korelasyon işlemi denmektedir. Şekil 2.8'de 2×2 filtre kullanılarak gerçekleştirilen bir çapraz korelasyon işlem örneği verilmiştir.



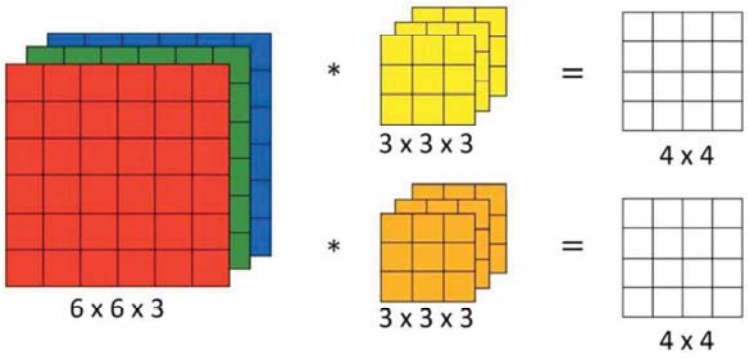
Şekil 2.8. Çapraz Korelasyon İşlemi

CNN'de evrişim işleminde iki boyutlu veriye uygulanacak olan filtrenin x ve y eksenine göre simetrisi alınır. Bütün değerler matriste birbirleriyle çarpılır ve bütün değerlerin toplamı çıkış matrisinin ilgili elemanı olarak kaydedilir. Simetrisi alınmadan yapılan işlem Şekil 2.8'de gösterildiği gibi gerçekleşir. Örnek bir görüntü için, giriş verisi tek kanallı iken bu işlem basitçe yapılabilmektedir. Ancak görüntü farklı formatlarda ve farklı kanal sayısına sahip olabilir. Şekil 2.9'da 2 boyutlu kaydırma uygulanarak gerçekleştirilen evrişim işlemi örneği gösterilmiştir. Gerçekleştirilen hesaplama işlemini sinir ağındaki bir katman olarak düşünelim. Giriş görüntüsü ve filtre sürekli geri yayılımla güncellenen ağırlıklar matrisidir. Aktivasyon fonksiyonu uygulanan çıkış matrisine bir $bias$ değeri eklenir. CNN mimarileri, girdilerin mimaride belirli özellikleri kodlamamıza izin veren görüntüler olduğu konusunda açık bir varsayımda bulunur. Bunlar daha sonra ileri yayılım fonksiyonunu, ağdaki parametre miktarını uygulamak ve büyük ölçüde azaltmak için daha verimli kılar.



Şekil 2.9. Evrişim İşlemi Gösterimi

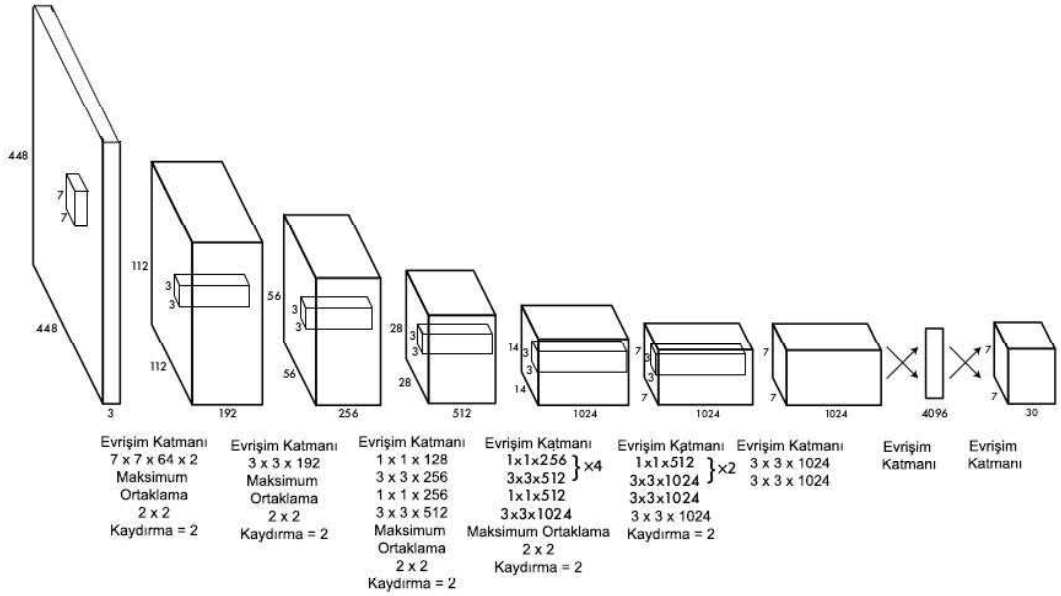
Renkli görüntüler, kırmızı, yeşil ve mavi olmak üzere üç kanaldan meydana gelmektedir. Bu şartlarda evrişim işlemi üç kanal için yapılır. Çıkış işaretinin kanal sayısı da uygulanan filtre sayısı ile eşit olarak hesaplanır. Giriş görüntüsünün yüksek frekanslı bölgeleri kenarları göstermektedir. Kenarlar en çok ihtiyaç duyulan özniteliklerdir. Şekil 2.10'da kenar bilgisini elde etmek için üç kanaldan oluşan bir görüntüdeki yatay ve dikey filtreleme işlemi gösterilmektedir.



Şekil 2.10. Yatay ve Dikey Filtreleme İşlemi

Giriş ve çıkış boyutu arasında, evrişimsel sinir ağlarında yapılan hesaplamalar sonucunda farklılık meydana gelmektedir. Şekil 2.10'da 6×6 boyutunda üç kanallı giriş görüntüsüne 3×3'lük bir filtre uygulanmaktadır. Giriş vektörü $a \times a$, filtre $b \times b$olarak kabul edildiğinde $(a - b + 1) * (a - b + 1)$ formülüne göre gerçekleşen evrişim işlemi sonunda elde edilen çıkış görüntüsünün boyutları 4×4 olur.

Filtreler CNN mimarisinde farklı özellikleri öğrenmektedir. Filtreler istenen sınıflandırma için görüntünün özelliklerini çıkarmaktadırlar. Bunun için CNN mimarisinde birden çok filtre kullanılmaktadır. Böylelikle CNN mimarisi, farklı özellikleri elde eden filtreleri öğrenebilmektedirler.



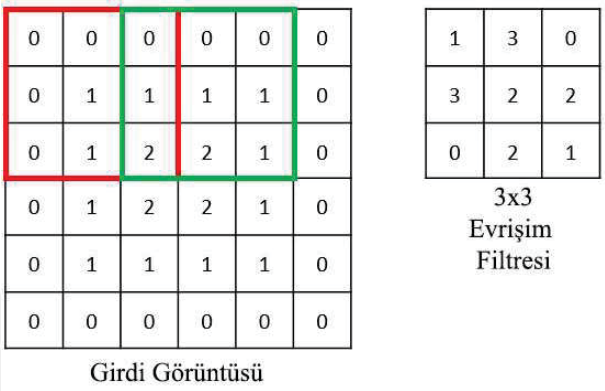
Şekil 2.11. Örnek CNN mimarisi

Örneğin Şekil 2.11'de $7 \times 7 \times 64 \times 2$'nin anlamı; 7×7 boyutunda 64 tane filtre uygulanıyor ve bu filtreler her uygulama sonrası 2 piksel kaydırılıyor. Bu mimaride kullanılan her bir filtre farklı özellikler öğrenmektedir. CNN mimarisinde bölgesel olarak bağlılık bulunmaktadır. Giriş görüntümüzün boyutu $448 \times 448 \times 3$ ve kullanılan filtre 7×7 olduğunda her bir düğüm için $7 * 7 * 3 = 147$ adet ağırlık bulunur.

CNN mimarisinde filtre sayısını çok fazla arttırmak bazen işlem maliyetini arttırır ve bazı filtrelerin aynı özellikleri öğrenmesine neden olur. Bu yüzden filtre sayısının kullanılacak Girdi vektörüne göre ayarlanması gerekmektedir. CNN mimarisinde kullanılacak derinlik parametresi de kullanılacak filtre sayısıyla ilgilidir. CNN mimarisinde çıktı boyutunun belirlenmesinde derinlik, kaydırma ve sıfır ekleme işlemleri uygulanır.

Kaydırma (Stride): Evrişim işleminde filtrelerin özellikleri öğrenmeleri için, filtreler girdi vektörü üzerinde kaydırılmaktadır. CNN'de kaydırma, evrişim işlemi için ağırlık

matrisi olan filtreyi giriş vektörü üzerinde birer piksellik adımlarla ya da daha büyük adımlarla kaydıracağının bilgisini verir. Kaydırma işlemi yapılırken noktasal çarpım gerçekleşmektedir. Kaydırma miktarı 2 olduğunda, giriş vektörü üzerinde filtrenin 2 piksel kaydırılması demektir. Kaydırma işlemi hassas bir şekilde belirlenmelidir. Şekil 2.12'de örnek bir kaydırma işlemi verilmiştir. Kaydırma işlemi doğrudan çıkış boyutunu etkileyen diğer bir parametredir.



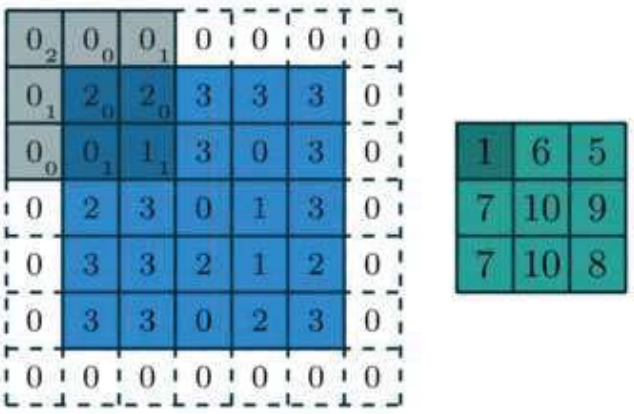
Şekil 2.12. Giriş Görüntüsünde Kaydırma İşlemi

Sıfır Ekleme (Zero Padding): CNN'de girdi vektörü ile çıkış vektörünün aynı boyutta olması istendiğinde özellik haritasının kenarlarına sıfır ekleme işlemi yapılır. Evrişim işleminden sonra giriş değeri ile çıkış değeri arasındaki boyut farkını ayarlamak elimizde olan bir hesaplamadır. Bu işlem giriş matrisine eklenecek ekstra pikseller ile sağlanır. Genellikle girdi ile çıktının aynı boyutta olması istendiğinde üretilen özellik haritasının kenar bölgelerine girdi ile aynı boyutta olacak şekilde sıfır değeri eklenir. Çıktı boyutu Denklem 2.6'daki gibi belirlenir

$$y = (x - f + 2s)/k + 1 \tag{2.6}$$

Denklem 2.6'de y çıktı boyutunu, x girdi boyutunu, f filtre boyutunu, s sıfır ekleme işlemini ve k ise kaydırma işlemini belirtmektedir. Şekil 2.13'de 5×5'lik bir giriş görüntüsüne 3×3'lük bir filtre uygulanmaktadır. 1x1 sıfır ekleme ve 2×2 kaydırma olduğunda, üretilen çıktı 3×3 olmaktadır. Kaydırma miktarı, sıfır ekleme işlemi ve filtre boyutu düzgün belirlenmezse üretilen çıktı tam sayı olmayacaktır ve hatalı bir işlem

gerçekleşecektir. Bu yüzden bu işlemlerin Denklem 2.6'da verilen formüle göre hesaplanması gerekmektedir.



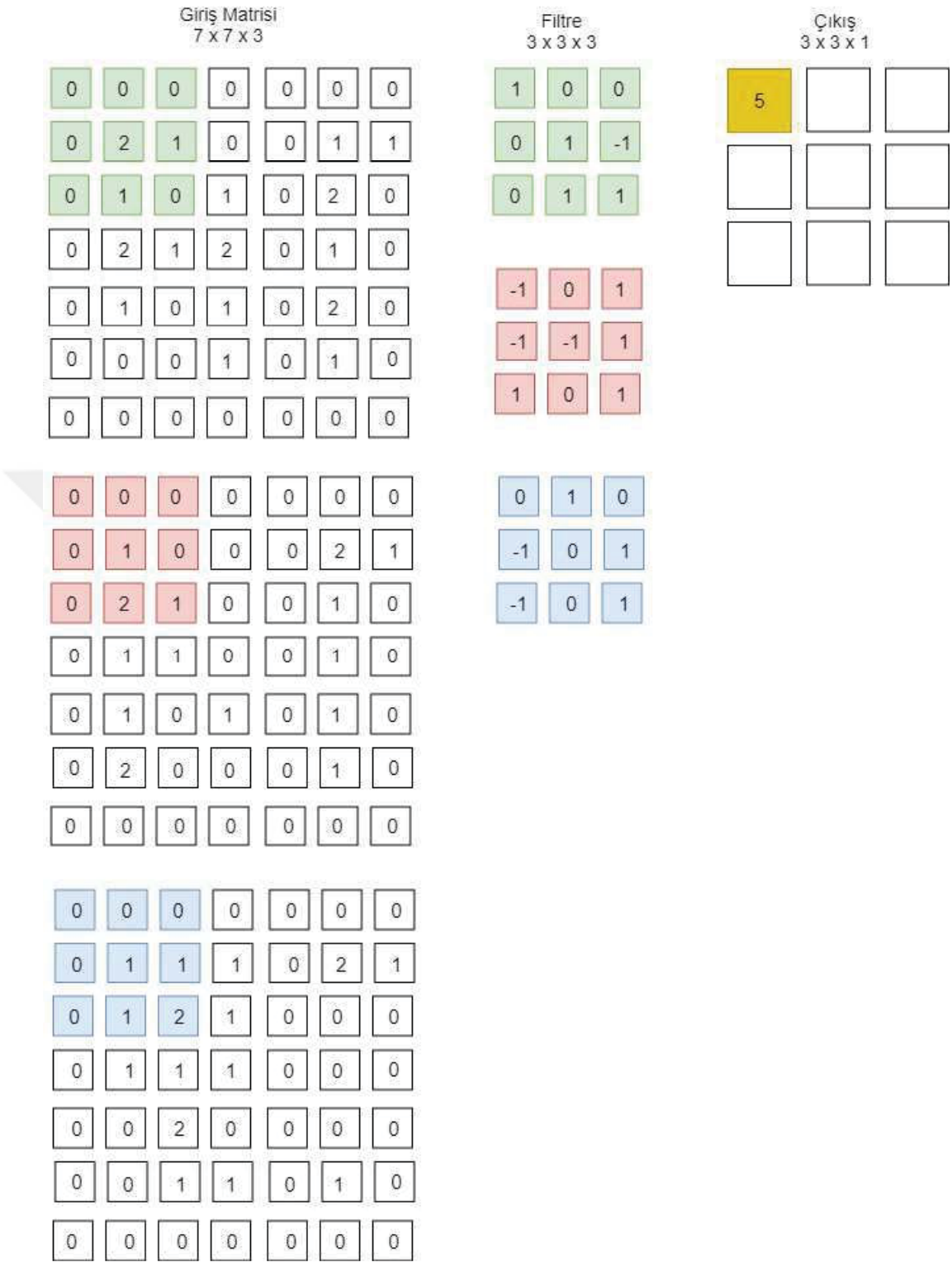
Şekil 2.13. Sıfır Ekleme İşlemi

Evrişim Örneği: Her bir boyut ayrı renklendirilerek $5 \times 5 \times 3$'lük bir giriş görüntüsüne $3 \times 3 \times 3$'lük bir filtre 2 adım kaydırmayla uygulanmaktadır ve 1×1'lik sıfır ekleme yapılmıştır. Çıktı boyutu Denklem 2.6'ya göre (5 - 3 + 2)/2 + 1 = 3 olarak hesaplanır. Şekil 2.14'te evrişim işleminin birinci adımı, Şekil 2.15'te evrişim işleminin 2. Adımı gösterilmiştir. Şekil 2.14'te aynı renkteki matrisler için birbirleriyle evrişim uygulanmıştır. Evrişim işleminde giriş görüntüsü üzerinde filtre kaydırılarak noktasal çarpma işlemi yapıldıktan sonra, çıkan sonuçlar bir birleriyle toplanarak sonuç hesaplanmıştır.

$$C_{1,1} = (0*1) + (0*0) + (0*0) + (0*0) + (2*1) + (1*1) + (0*0) + (1*-1) + (0*1) \qquad C_{1,1} = 2$$

$$C_{1,2} = (0*-1) + (0*-1) + (0*1) + (0*0) + (1*-1) + (2*0) + (0*1) + (0*1) + (1*1) \qquad C_{1,2} = 0$$

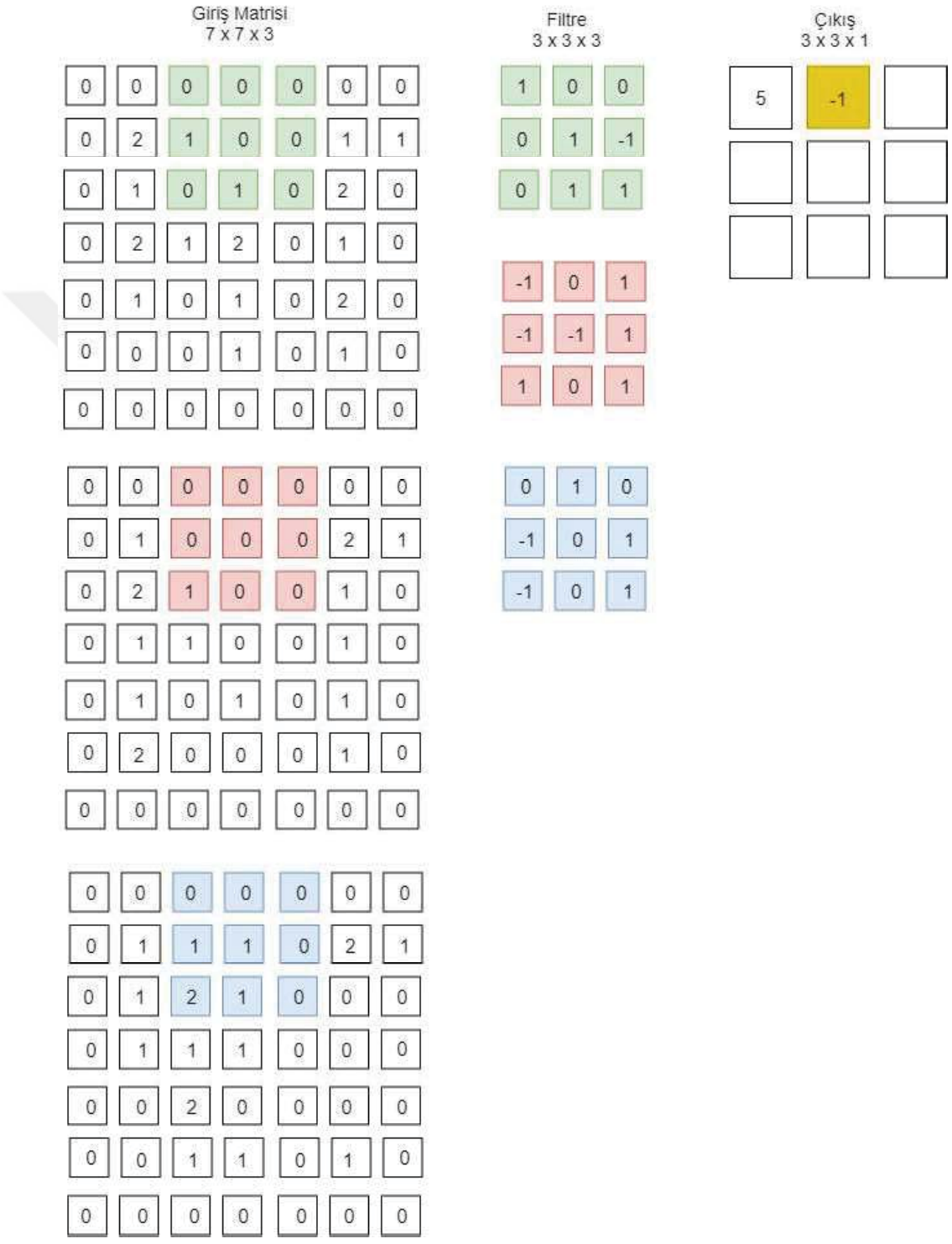
$$C_{1,3} = (0*0) + (0*-1) + (0*-1) + (0*1) + (1*0) + (1*0) + (0*0) + (1*1) + (2*1) \qquad C_{1,3} = 3$$



Şekil 2.14. Evrişim İşlemi Örneği – Adım 1

$C_{11,}$ yeşil renkteki vektörlerin çıkış değerini temsil etmektedir. $C_{1,2}$ kırmızı renkteki vektörlerin çıkış değerini temsil etmektedir. $C_{1,3}$ mavi renkteki vektörlerin çıkış değerini

temsil etmektedir. Çıkış vektörü sonuçları üretildikten sonra, bu değerler toplanıp yerel evrişim hesaplanır. $C_1 + C_2 + C_3 = 2 + 0 + 3 = 5$ sonucu Şekil 2.15'de hardal rengi ile gösterilen bölgeye yazılır.



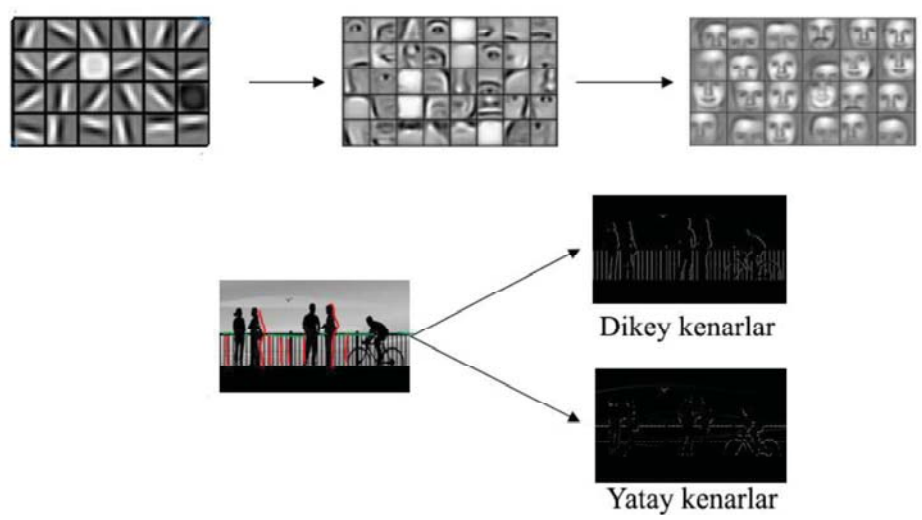
Şekil 2.15. Evrişim İşlemi Örneği – Adım 2

$$C_{2,1} = (0 * 1) + (1 * 0) + (0 * 0) + (0 * 0) + (0 * 1) + (1 * 1) + (0 * 0) + (0 * -1) + (0 * 1) \qquad C_{2,1} = 1$$

$$C_{2,2} = (0 * -1) + (0 * -1) + (1 * 1) + (0 * 0) + (0 * -1) + (0 * 0) + (0 * 1) + (0 * 1) + (0 * 1) \qquad C_{2,2} = 1$$

$$C_{2,3} = (0 * 0) + (1 * -1) + (2 * -1) + (0 * 1) + (1 * 0) + (1 * 0) + (0 * 0) + (0 * 1) + (0 * 1) \qquad C_{2,3} = -3$$

Şekil 2.15'de bir sonraki yerel bölge için evrişim işlemi aynı şekilde yapılır. $C_{2,1}$ yeşil, $C_{2,2}$ kırmızı, $C_{2,3}$ ise mavi renkle verilen vektörlerin çıkış değerini temsil etmektedir. Görüntü matrisi üzerinde kaydırma işlemini gerçekleştirirken dikkat etmek gerekmektedir. Verilen miktar kadar piksel kaydırması yapılmalıdır. 2 piksel kaydırılarak çarpılacak olan bölgenin belirlenmesi gerekmektedir.

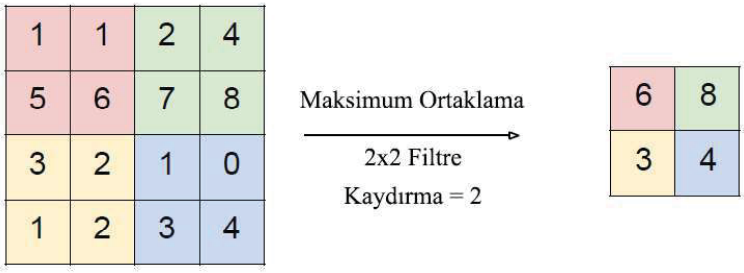


Şekil 2.16. Kenar Bulma İşlemi

Kenar Bulma: Görüntüden elde edilen öznitelikler içinde en çok kenar bilgilerine ihtiyaç duyulmaktadır. Bir görüntüde kenarlar yüksek frekanslı bölgeleri temsil etmektedir. Bir görüntüdeki kenar bilgilerini elde etmek için yatay ve dikey olmak üzere iki filtre ayrı

ayrı kullanılır. Şekil 2.16'da geleneksel kenar bulma yöntemleri kullanılarak evrişim işlemi gerçekleştirildiğinde, üretilen çıkış değeri görüntünün kenar bilgilerini gösterir.

Ortaklama (Pooling): Genellikle bu katmanda maksimum ortaklama yöntemi kullanılır. Öğrenilen parametre ağın bu katmanında yoktur. Giriş matrisinde kanal sayısı sabit tutularak, yükseklik ve genişlik bilgisi azaltılır. Hesaplama karmaşıklığını azaltmak için kullanılır. Bu yöntem verideki önemli bazı bilgilerin de kaybolmasına sebep olduğu için başarımdan ödün vermektedir. Lokasyon bilgisinin çok önemli olmadığı işlemlerde bile oldukça güzel sonuçlar vermektedir. Giriş vektörü üzerinde kaydırma işlemi gerçekleşirken, seçilen ortaklama boyutu içindeki piksellerin en büyüğünü çıkışa aktarır. Şekil 2.17'te 2×2 filtre ile maksimum ortaklama işlemi 2 adım kaydırılarak uygulanmıştır. İki piksellik kaydırma işlemi uygulanırken ilgili dört elemanın olduğu alandaki en büyük değer çıkışa aktarılır. Çıkışta 4×1 boyutlu bir veri elde edilmiş olur.



Şekil 2.17. Ortaklama İşlemi Gösterimi

Düzleştirme Katmanı (Flattening Layer): Bu katman bütün bağlı katmanın girişindeki değerleri hazırlar. Sinir ağı giriş verilerini genel olarak tek boyutlu bir diziden alır. Bu sinir ağındaki veriler ise evrişim ve ortaklama katmanından gelen matrislerin tek boyutlu diziye çevrilmiş halidir.

Bütün Bağlı Katman (Fully Connected Layer): CNN'in son ve en önemli katmanıdır. Verileri düzleştirme katmanından alır ve sinir ağı yoluyla öğrenme işlemini geçekleştirir.

Erken Durdurma (Early Stopping): Eğitim esnasında güncelleme işlemi gerçekleştirilirken, tekrarlanan adımlar bazen ağın başarımını iyileştirmek yerine kötüleştirebilmektedir. Sinir ağını eğitirken doğrulama kümesindeki hatayı en aza

indirerek eğitim seti üzerinde değişiklikler yapılmalıdır. Gerçekleştirilen değişiklikler sayesinde eğitim verisinin hatası düşmeye devam eder, fakat ara-test setinin hatası bir noktadan sonra artmaya başlayabilir. Bu aşamada yapılması gereken işlem eğitimi erken durdurmaktır. Böylelikle sinir ağı, olabilecek en iyi durumdayken güncelleme işlemi bitirilmelidir.

Seyreltme (Dropout): Bütün bağlı katmanlarda belli eşik değerin altında kalan düğümlerin seyreltilmesi eğitim başarısını arttırabilir. Yani zayıf bilgilerin unutulması eğitim başarını arttırmaktadır. Genelde seyreltme değeri 0,5 olarak kullanılmaktadır, fakat bu sayı veri setine ve probleme göre değişebilir. Seyreltme işlemi rasgele eleme yöntemiyle yapılmalıdır. Her katmanda aynı seyreltme değerinin kullanılması zorunlu değildir.

Yığın Boyutu: Veri setinde bulunan tüm verileri aynı anda işleyerek öğrenme, derin öğrenmede zaman ve bellek açısından maliyetli bir iştir. Çünkü öğrenmenin her adımın geriye yayılım işlemi ile ağ üzerinde geriye dönük olarak gradyan inişi yapılmaktadır ve böylece ağırlık değerleri güncellenmektedir. Veri sayısı ne kadar fazla ise, hesaplama da o oranda fazla sürmektedir. Bu tarz problemleri çözmek için, veri seti küçük gruplara ayrılabilir ve öğrenme işlemi seçilen bu küçük gruplar üzerinde yapılabilir. Böylece birden fazla girdinin parçalar halinde işlenmesi yığın boyutu olarak adlandırılmaktadır. Model tasarlanırken yığın (batch) parametresi olarak belirlenen değer, modelin aynı anda kaç veriyi işleyeceği anlamına gelmektedir. Verilerin gruplar halinde işlenmesi durumunda öğrenmede kayıp değeri artabilir, fakat zamandan maliyetini azaltabiliriz.

Yığın Normalizasyonu (Batch Normalization): Yığın normalizasyonu sinir ağının hızını, performansını ve dengesini arttırmak için kullanılan bir tekniktir. Aktivasyon fonksiyonunu ayarlayarak ve ölçekleyerek giriş katmanını normalleştirmek için kullanılır. Normalize edilmemiş modele kıyasla öğrenme oranı arttırılabilir.

2.4. Denetimli Öğrenme (Supervised Learning)

Denetimli öğrenme yöntemleri açıkça sürücüsüz araçları eğitmenin basit yoludur. Modelleri basit ve doğrudandır, giriş görüntülerini sığ bir ağ kullanarak eylem tahminlerine eşleştirir. Makine öğrenmesinin en yaygın şekli denetimli öğrenmedir. Bir evi, bir otomobili, bir insanı veya evcil bir hayvanı içeren görüntüleri sınıflandırabilecek bir sistem kurmak istediğimizde, ilk önce her biri kategorisiyle etiketlenmiş ev, araba, insan ve evcil hayvan resimlerinden oluşan geniş bir veri kümesi oluşturulmalıdır. Eğitim sırasında, sisteme bir resim gösterilir ve sistem her kategori için bir tane skor vektörü şeklinde bir çıktı üretir. İstediğiniz kategorinin tüm kategorilerde en yüksek skora sahip olmasını istiyoruz, ancak bunun eğitimden önce gerçekleşmesi olası değildir. İstenen skor ile çıkış skoru arasındaki hatayı (veya uzaklığı) ölçen nesnel bir fonksiyon oluşturulmalıdır. Sistem daha sonra bu hatayı azaltmak için ayarlanabilir parametrelerini değiştirir. Genellikle ağırlık (weights) olarak adlandırılan bu ayarlanabilir parametreler, sistemin giriş-çıkış fonksiyonunu tanımlayan gerçek rakamlardır. Tipik bir derin öğrenme sisteminde, makineyi eğitmek için yüz milyonlarca ayarlanabilir ağırlık ve yüz milyonlarca etiketli örnek olabilir. Ağırlık vektörünü uygun şekilde ayarlamak için, öğrenme algoritması, her ağırlık için, ağırlık küçük miktar arttırıldıysa hatanın ne kadar artacağını veya azaltacağını gösteren bir gradyan vektörü hesaplar. Ağırlık vektörü daha sonra gradyan vektörüne zıt yönde ayarlanır.

Tüm eğitim örnekleri üzerinden ortalama alınan amaç fonksiyon, yüksek boyutlu ağırlık değerlerinde bir tür tepelik olarak görülebilir. Negatif gradyan vektörü, bu tepelik olarak görünen şekildeki en dik inişin yönünü belirtir ve çıktı hatasının ortalama olarak düşük olduğu en düşük seviyeye yaklaşır. Uygulamada çoğu uygulayıcı stokastik gradyan inişi (stochastic gradient descent) adı verilen bir yöntem kullanır. Giriş vektörünü birkaç örnek için göstermek, çıktıları ve hataları hesaplamak, bu örnekler için ortalama gradyanı hesaplamak ve ağırlıkları buna göre ayarlamaktan ibarettir. Bu işlem, eğitim seti amaç fonksiyonunun ortalamasının düşmesi durana kadar birçok küçük örnek grubu için tekrarlanır. Buna stokastik denir, çünkü her küçük örnek kümesi, tüm örnekler üzerindeki ortalama gradyanın gürültülü bir tahminini verir. Bu basit prosedür genellikle çok daha ayrıntılı optimizasyon tekniklerine kıyasla hızlı bir şekilde iyi bir ağırlık seti bulur. Eğitimden sonra, sistemin performansı test seti adı verilen farklı bir dizi örnek üzerinde

ölçülür. Sisteme eğitim sırasında hiç görmediği yeni veriler verilir ve sistemin bu veriler hakkında cevaplar üretme yeteneğini test edilir.

Makine öğrenmesinin güncel pratik uygulamalarının birçoğu doğrusal sınıflandırıcılar kullanır. İki sınıflı bir doğrusal sınıflandırıcı, özellik vektör bileşenlerinin ağırlıklı toplamını hesaplar. Ağırlıklı toplam bir eşiğin üstünde ise, giriş belirli bir kategoriye ait olarak sınıflandırılır. Görevin önceden tanımlanmış kesin talimatlar dizisi olarak resmileştirildiği klasik bilgisayar programlarının aksine, ML algoritmaları kararlarını verilerden elde edilen bilgilere dayandırmaktadır. Bu, özellikle görevi açıkça belirtmenin mümkün olmadığı veya statik özelliklerin yeterince sağlam olmadığı durumlarda önemlidir. Bir resim sınıflandırma işleminde; her resim üzerinde bir kedi veya köpek görüntüsü bulunan bir veri kümesi verilmiştir. Bazı resimlerde bir kedi ya da köpek olup olmadığına dair etiketler verilmiş ve başka bir görüntü grubu için etiketler verilmemiştir. Eğitim seti bilinen etiketler ile eşleştirilmiş görüntü grubundan oluşur. Bilinmeyen etiketlere sahip görüntü grubuna genellikle test seti denir. Sistemin performansı eğitim setinden çıkarılan bilgilerin, test setindeki görüntü etiketleri hakkında öngörülerde bulunmak için ne kadar genelleştirildiğini ölçer (Krasheninnikov, 2017).

Sürücüsüz aracın yolda ilerlerken şerit takibinde, yol şeritlerinin ya da yol şeritlerinin bulunduğu sınıfı öğrenmek istediğimizi düşünelim. Elimizdeki farklı çizgi örneklerini sisteme gösterip hangisinin yol şeridi, hangisinin farklı çizgi olduğunu belirlemesini istiyoruz. Yol şeritlerini artı, farklı çizgileri ise eksi olarak etiketliyoruz. Tüm artı örneklerinin olduğu sınıfı tanımladığımızda öngörü yapabiliriz. Sisteme gösterilmemiş farklı özellikte olan bir çizgiyi sınıftaki örneklerle karşılaştırıp bunun yol şeridi olup olmadığı belirlenebilir. Bir sınıflandırıcı oluşturmak için; yol şeritlerini diğer çizgilerden ayıran renk (x_1) ve uzunluk (x_2) öznitelikleri sınıflandırıcının girdisi olarak kabul edildiğinde, bir şeridi sayısal değerle belirleyebiliriz ve bu şekilde etiket sınıf değerini belirtebiliriz.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.7}$$

$$r = \begin{cases} 1, & eğer\ x\ artı\ bir\ örnekse \\ 0, & eğer\ x\ eksi\ bir\ örnekse \end{cases} \tag{2.8}$$

Her bir çizgi kümesi (x, r) çiftiyle gösterilir ve öğrenme kümesi n örnekten oluşur.

$$x = \{x^t, r^t\}^n t = 1 \tag{2.9}$$

Burada t, kümedeki farklı örnekleri gösterir.
$\{y^{(1)},.....,y^{(m)}\}$ çıkış kümesi ile ilişkili olan $\{x^{(1)},.....,x^{(m)}\}$ veri noktalarının kümesi dikkate alındığında, y 'den x 'i nasıl tahmin edebileceğimizi öğrenen bir sınıflandırıcı tasarlamak istiyoruz. Çizelge 2.8'de farklı model tahminleri gösterilmiştir ve Çizelge 2.9'da ise farklı modellerin gösterimi verilmiştir.

Çizelge 2.8. Farklı tahmin modellerinin gösterimi (Amidi ve Amidi, 2018)

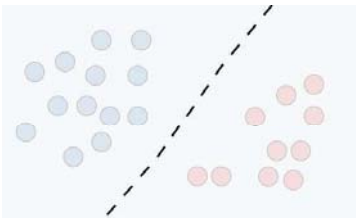
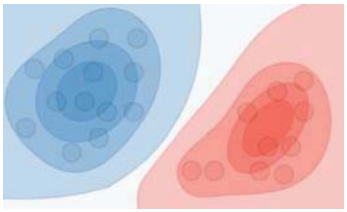
	Regresyon	**Sınıflandırıcı**
Çıkış	Sürekli	Sınıf
Örnekler	Lineer Regresyon	Lineer Regresyon, Destek Vektör Makineleri (Support Vector Machine SVM), Naive Bayes

Gösterimler ve Genel Kavramlar

Hipotez: Çizdiğimiz modelde hipotez h_θ olarak belirtilmiştir. Verilen $x^{(i)}$ verisi için modelin tahminlediği çıkış $h_\theta(x^{(i)})$ olarak verilmiştir. Eğitim kümesi kullanılarak hipotez fonksiyonuna en uygun parametreler seçildiğinde, uygun model oluşturulabilir. Elimizde bulunan önceki verilerle hipotezdeki uygun parametreler seçilebilir ve farklı veri girdilere karşılık tahmin yapılabilir.

Kayıp Fonksiyonu: $L : (z, y) \in R \times Y \mapsto L(z, y) \in R$ gösterimiyle tanımlanan kayıp fonksiyonu y gerçek değerine karşılık geleceği tahmin edilen z değerini giriş olarak alan ve farklı olduklarını gösteren bir fonksiyondur. Genel olarak kullanılan kayıp fonksiyonları Çizelge 2.10'da özetlenmiştir.

Çizelge 2.9. Farklı modellerin gösterimi (Amidi ve Amidi, 2018)

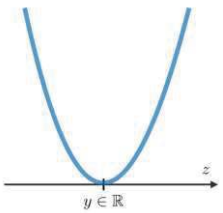
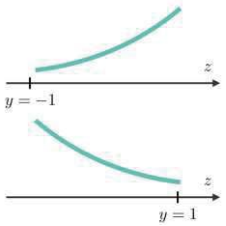
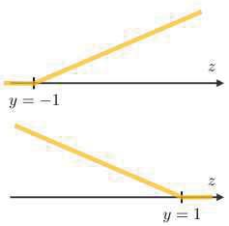
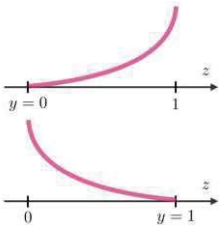
	Regresyon	Sınıflandırıcı
Amaç	Doğrudan tahmin $P(y \mid x)$	$P(y \mid x)$'i tahmin etmek
Öğrenilenler	Karar sınırı	Verilen olasılık dağılımı
Örnekleme		
Örnekler	Regresyon, SVM	Gauss Diskriminant Analiz (Gaussian Discriminant Analysis, GDA), Naive Bayes

Maliyet Fonksiyonu: Bir modelin performansını ölçmek için J maliyet fonksiyonu kullanılır.

$$J(\theta) = \sum_{i}^{m} L(h_0(x^{(i)}), y^{(i)}) \tag{2.10}$$

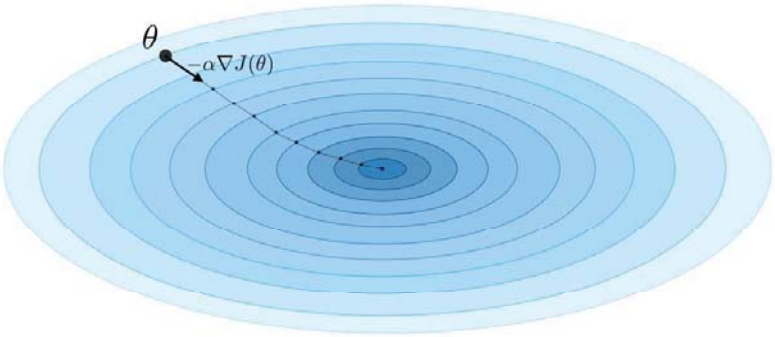
Denklem 2.10'da L kayıp fonksiyonu, h_θ modelin hipotezini, $x^{(i)}$ giriş kümesi ve $y^{(i)}$ çıkış kümesi olarak belirtilmiştir.

Çizelge 2.10. Yaygın olarak kullanılan kayıp fonksiyonları (Amidi ve 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

Gradyan İnişi: $\alpha \in R$ gradyan inişi için güncelleme kuralı olarak ifade edilen öğrenme oranı ve J maliyet fonksiyonu Denklem 2.11'deki gibi ifade edilir. Maliyeti minimize etmek için en uygun değerler bulunmaya çalışılır. En yüksek doğruluk oranı için hata

oranı en aza indirilmelidir. Gradyan inişi maliyet fonksiyonunu en aza indirerek minimum hatayı bulmak için kullanılmaktadır (Şekil 2.18).



Şekil 2.18. Gradyan inişi

Denklem 2.11'de α öğrenme katsayısını ifade etmektedir. Yığın gradyan inişi bir dizi eğitim örneği üzerindedir ve stokastik gradyan inişi her eğitim örneğine bağlı olarak parametreyi günceller.

$$\theta \leftarrow \theta - \alpha \nabla J\,(\theta) \tag{2.11}$$

Olabilirlik: θ parametrelerini verilen bir $L(\theta)$ modelinin olabilirliği maksimize ederek en uygun θ parametrelerini bulmak için kullanılır. Uygulamada optimize edilmesi daha kolay olan log-olabilirlik $l(\theta) = \arg\max_{\theta} L(\theta)$ kullanılır.

$$\theta^{opt} = \arg\max_{\theta} L(\theta) \tag{2.12}$$

Newton Algoritması: $l'(\theta) = 0$ olacak şekilde bir θ bulan sayısal bir yöntemdir.

$$\theta \leftarrow \theta - \frac{l'(\theta)}{l''(\theta)} \tag{2.13}$$

Denklem 2.14'de verilen güncelleme kuralına sahip olan formül Newton-Raphson yöntemi olarak da bilinir ve çok boyutlu genelleme yapar.

$$\theta \leftarrow \theta\,(\nabla_\theta^2\, l(\theta))^{-1}\nabla_\theta l(\theta) \tag{2.14}$$

Doğrusal (Lineer) Regresyon: $y|x;\theta \sim N(\mu,\sigma^2)$ olduğunu varsayıyoruz. X matris tasarımı olmak üzere, maliyet fonksiyonunu en aza indiren θ değeri X'in matris tasarımını kaydederek maliyet fonksiyonunu en aza indiren θ değeri kapalı formu bir çözümdür. Doğrusal regresyon birbirinden bağımlı veya bağımsız olan parametreler arasındaki ilişki fonksiyonunu elde eder.

$$\theta = (X^T X)^{-1} X^T y \tag{2.15}$$

En Küçük Ortalama Kareler Algoritması: α öğrenme oranı olmak üzere, m veri noktasını içeren eğitim kümesi için Widrow-Hoff öğrenme oranı olarak bilinen En Küçük Ortalama Kareler Algoritmasının (Least Mean Squares) güncelleme kuralı Denklem 2.16'da verilmiştir. Denklem 2.16'daki güncelleme kuralı gradyan yükselişin özel bir gösterimidir.

$$\forall j, \qquad \theta \leftarrow \theta + \alpha \sum_{i=1}^{m} \left[y^{(i)} - h_\theta(x^{(i)})\right] x_j^{(i)} \tag{2.16}$$

Yerel Ağırlıklı Regresyon: Yerel Ağırlıklı Regresyon (Locally Weighted Regression, LWR) ağırlıkları her eğitim örneğini maliyet fonksiyonunda $w^i(x)$ ile ölçen doğrusal regresyonun bir çeşididir.

$$w^i(x) = \exp\left(\frac{(x^i - x)^2}{2\tau^2}\right) \tag{2.17}$$

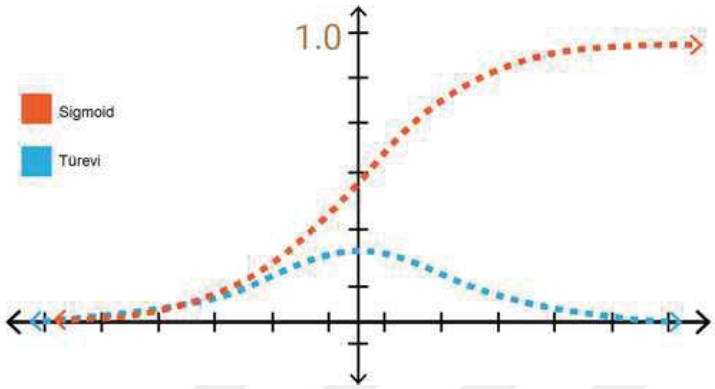
Sınıflandırma ve Lojistik Regresyon

Sigmoid fonksiyonu: Lojistik fonksiyonu olarak da bilinen sigmoid fonksiyonu g, denklem 2.18'deki gibi tanımlanır.

$$\forall z \in R, \quad g(z) = \frac{1}{1+e^{-z}} \in]0,1[\tag{2.18}$$

Şekil 2.19'da görüldüğü gibi sigmoid fonksiyonu 0 ile 1 arasında değer üretir. Çıkış değerleri, giriş değerlerine çok fazla tepki vermediğinde türev değerleri çok küçük olur

ve sıfıra yakınsar ve öğrenme minimum düzeyde gerçekleşir. Bu durumda hatayı minimize eden optimizasyon algoritması yerel minimum değerlere takılabilir ve oluşturduğumuz modelden yüksek performans alamayız.



Şekil 2.19. Sigmoid Fonksiyonu (Kızrak 2019b)

Çizelge 2.11. Yaygın olarak kullanılan üstel dağılımlar (Amidi ve Amidi, 2018)

Dağılım	$\boldsymbol{\eta}$	$\boldsymbol{T(y)}$	$\boldsymbol{a(\eta)}$	$\boldsymbol{b(\eta)}$
Bernoulli	$\log\left(\frac{\phi}{1-\phi}\right)$	y	$\log(1 + \exp(\eta))$	1
Gauss	μ	y	$\frac{n^2}{2}$	$\frac{1}{\sqrt{2\pi}} exp\left(-\frac{y^2}{2}\right)$
Poisson	$\log(\lambda)$	y	e^n	$\frac{1}{y!}$
Geometrik	$\log(1-\phi)$	y	$\log\left(\frac{e^n}{1-e^n}\right)$	1

Lojistik Regresyon: $y|x;\theta \sim Bernoulli(\phi)$ olduğunu varsayıyoruz. Denklem 2.19'da lojistik regresyon formülü gösterilmiştir. Lojistik regresyon durumunda kapalı form çözümü yoktur.

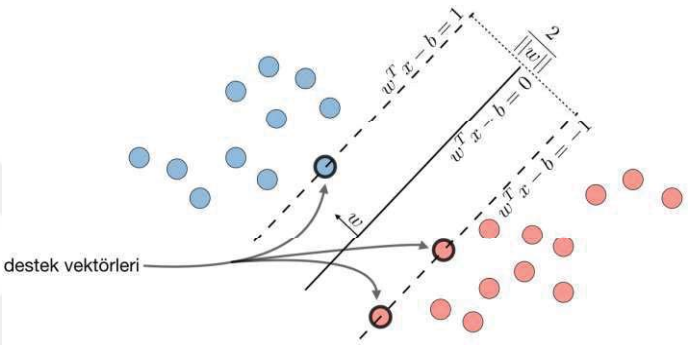
$$\phi = p(y = 1|x;\theta) = \frac{1}{1 + \exp(-\theta^T x)} = g(\theta^T x) \tag{2.19}$$

Softmax Regresyonu: Fazla sayıda sınıf olduğunda lojistik regresyonu genelleştirmek için, çok sınıflı lojistik regresyon olarak da adlandırılan, Softmax regresyonu kullanılır. Genel kabul olarak, her i sınıfı için Bernoulli parametresi ϕ_i'nin eşit olmasını sağlaması için $\theta_K = 0$ olarak ayarlanır. Çizelge 2.11'de yaygın olarak kullanılan üstel diyagramlar gösterilmiştir.

$$\phi_i = \frac{\exp(\theta_i^T x)}{\sum_{j=1}^{K} \exp(\theta^T x)} \tag{2.20}$$

Destek Vektör Makineleri (Support Vector Machine)

Destek Vektör Makinelerinin amacı minimum mesafeyi maksimuma çıkaran doğruyu bulmaktır. Şekil 2.20'de Destek Vektör Makinelerinin gösterimi verilmiştir. Gösterimde verilen doğru $w^T x - b = 0$ şeklinde tanımlanır.



Şekil 2.20. Destek Vektör Makinesi

Optimal marj sınıflandırıcısı: h optimal marj sınıflandırıcısı (Optimal margin classifier) Denklem 2.21'de verilmiştir.

$$\mathrm{h(x)} = sign(w^T x - b) \tag{2.21}$$

Denklem 2.21'de $(w, b) \in R^n \times R$, aşağıda verilmiş olan Denklem 2.22'nin çözümüdür.

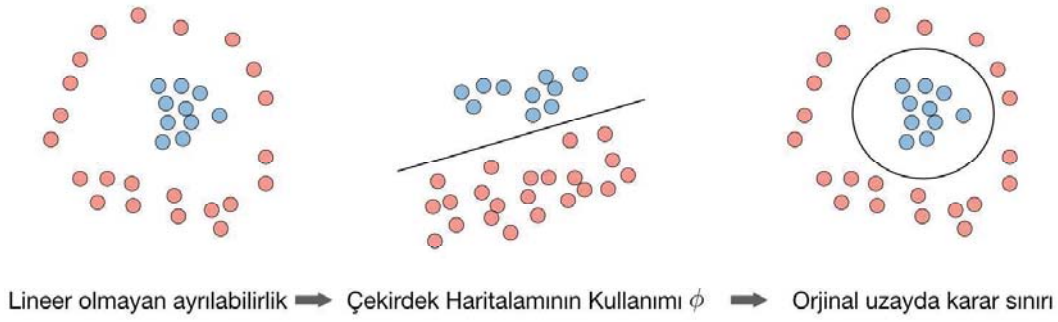
$$min \frac{1}{2} ||w||^2 \quad öyle\ ki \quad y^{(i)}\left(w^T x^{(i)} - b\right) \geq 1 \tag{2.22}$$

Menteşe Kaybı: Menteşe yitimi (Hinge loss), Destek Vektör Makinelerinin ayarlarında kullanılır ve aşağıda verilen Denklem 2.23'deki gibi tanımlanır.

$$L(z, y) = [1 - yz]_+ = \max(0, 1 - yz \tag{2.23}$$

Çekirdek: Çekirdek (Kernel), ϕ gibi bir özellik haritası verildiğinde, K olarak verilen çekirdeği tanımlarız. Çekirdeği kullanarak maliyet fonksiyonunu hesaplamak için çekirdek numarasının kullanıldığı belirtilmelidir, çünkü genellikle çok karmaşık olan ϕ

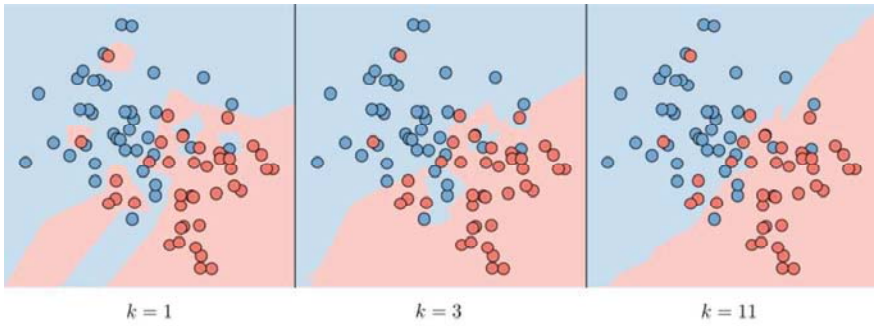
açık haritalamasını bilmeye gerek yoktur. Bunun yerine, yalnızca $K(x, z)$ değerlerine ihtiyaç duyulmaktadır (Şekil 2.21).



Şekil 2.21. Çekirdek Haritalaması

Üretici Öğrenme: Üretici Öğrenme'de (Generative Learning) üretken bir model, önce Bayes kuralını kullanarak $P(y|x)$ değerini tahmin etmek için kullanabileceğimiz $P(x|y)$ değerini tahmin ederek, verilerin nasıl üretildiğini öğrenmeye çalışır.

Denetimli öğrenmede bazı parametrik olmayan yaklaşımlar kullanılmaktadır. k-en yakın komşular algoritması k-NN olarak adlandırılmaktadır. k-NN veri noktasının tepkisi eğitim kümesindeki kendi k komşularının doğası ile belirlenen parametrik olmayan bir yaklaşımdır. Hem sınıflandırma, hem de regresyon yöntemleri için kullanılabilir. k parametresi ne kadar yüksekse, yanlılık o kadar yüksek ve k parametresi ne kadar düşükse, varyans o kadar yüksek olur (Şekil 2.22).



Şekil 2.22. k-NN Sınıflandırması

Denetimli öğrenme en yaygın kullanılan öğrenme yöntemidir. Gerçekleşecek durumun çıktıları eğitim verisinde bulunur. Veri örnekleri kullanılarak en iyi karar modeli oluşturulmaya çalışılır ve tüm uzayı temsil etmektedir. Denetimli öğrenme girişleri ve

çıkışları belli olan, önceki olay örnekleri kullanılarak gelecek durumlar için sonucu tahmin etmeye çalışır.

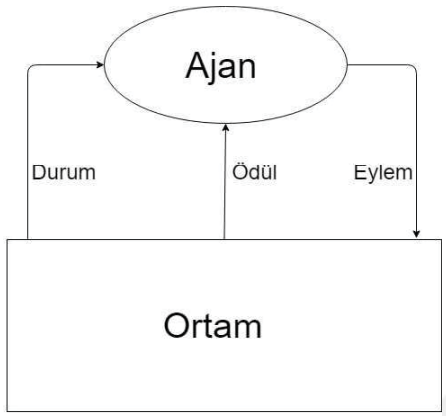
2.5. Pekiştirmeli Öğrenme (Reinforcement Learning)

Pekiştirmeli öğrenme diğer öğrenme tekniklerinden farklılık göstermektedir. Pekiştirmeli öğrenme, Markov karar sürecinin algılama (sensation), eylem (action) ve hedef (goal) özelliklerini kullanmaktadır. Bu yaklaşımda geçmiş verilerden öğrenme aşaması yoktur. Öğrenme deneme yanılma yöntemiyle olur. Gelecekte ne yapılması gerektiği bilinmez, fakat geçmişte yapılan eylemlerin iyi olup olmadığı bilinir. Geri besleme seyrektir. Birçok aksiyon gerçekleştirip ödül aldıktan sonra, gerçekleştirilen eylemler arasından hangilerinin ödül kazanmakta işe yaradığı bilinir. Sonraki gerçekleşecek eylemlerde bunlar güncellenir. Pekiştirmeli öğrenme algoritmaları, ara durumların amaca ne kadar yakın olduğunu gösteren iç bir değer öğrenmektedir. Öğrenme gerçekleştiğinde en yüksek ödül elde edilen aksiyonlar seçilir. Şekil 2.23'te ajanın ortam ile ilişkisi gösterilmektedir. Ortamın belli durumunda ajan eylem gerçekleştirir. Bu eylem sonucunda ortam değişebilir ve ödül elde edilebilir. Pekiştirmeli öğrenmede ajanın, ortama göre geliştirdiği aksiyona politika (policy) denir. Politika, π Markov karar sürecinde ajan davranışını karakterize eder.

$$\pi(a|s) = P(A_t = a|S_t = s) \tag{2.24}$$

Denklem 2.3'te S, olası tüm durumların kümesini göstermektedir, t ise zamanı belirtmektedir. S_t ajanın t anındaki durumunu gösterir. A_t ajanın t anındaki aksiyonudur. π ile gösterilen politika ajanın durumunu tanımlar. $A(s_t)$, s_t durumunda olası tüm aksiyonların kümesidir. Ajan, s_t durumunda a_t aksiyonunu yaptığı zaman bir birim ilerler, $r_{t+1} \in R$ ödülünü alır ve ajan bir sonraki s_{t+1} durumuna geçer. Gerçekleşen işlem Markov karar süreci kullanılarak modellenir. Sonraki durum ve ödül değerleri $p(r_{t+1}|s_t, a_t)$ ve $P(s_{t+1}|s_t, a_t)$ olasılık dağılımlarından örneklenir. Gerçekleşen aksiyona bağlı olarak, sonraki aksiyon ve ödül gerçekleşir. Bu süreç Markov süreci olarak adlandırılır. Ajan geliştirdiği aksiyona karşı çevreden bir tepki bekler. Önceden belirlenmiş ödül sistemi vardır. Ajanın eğitimi kazanılan ödül (reward) doğrultusunda

gerçekleşir ve gerçekleştirilen eğitimin ne kadar doğru ya da yanlış olduğunu anlar. Ajanın çevrede oluşturduğu olumlu ya da olumsuz tepkiler ödülü belirler. Yeni gerçekleştirmiş olduğu eyleme karşılık çevreden aldığı puandır.



Şekil 2.23. Pekiştirmeli Öğrenme Ajan Ortam İlişkisi

Pekiştirmeli öğrenmede amaç, uzun vadede alınacak ödülleri en yüksek seviyeye çıkarmaktır. Eylemin ne kadar iyi ya da kötü olduğunu ödül belirler. Ajan, zaman içerisinde izlemiş olduğu politikayı değiştirir. İleride kazanılacak ödül, durum (state) değeri olarak düşünülebilir. Ödülü düşük olan bir seçeneğin daha sonrasında büyük ödüllere götürebilme olasılığıdır. Bir sonraki adımda ne olacağını belirlemek için ödül değeriyle birlikte durum değeri ajana verilir. Ajan verilen ödülü en yüksek seviyede tutmak isterken yeni durumlar keşfetmelidir. Bir aksiyon düşük bir ödüle ama yüksek bir değere sahip olabilir. Bunun nedeni düşük ödül veren aksiyonun devamında gelecek yüksek ödüllü diğer aksiyonlardır. Sürekli olarak yüksek ödül veren bir aksiyondan sonra düşük ödüller veren aksiyonlar da olabilir. Burada oluşan durum "ileri görüşlülük" olarak düşünülebilir.

Pekiştirmeli öğrenmede deneme yanılma yöntemiyle eğitim gerçekleştiği için modelin öğrenmesinde hatalar oluşabilir. Gerçekleşen hatalar ajanın yeni durumlar keşfetmesini ve öğrenmesini sağlarken daha iyi ödüllere de götürebilir. Ajanın her aşama sonunda verdiği kararlar deneyimlerle rasgele olmaktan çıkıp deneyimlere dönüşebilir. Pekiştirilmeli öğrenmede keşif (exploration) ve sömürü (exploitation) gibi kavramların uygulamaya geçirilmesi gibi zorluklar vardır. Ajan pozitif ödül aldığı ve geçmişte gerçekleştirdiği eylemleri gerçekleştirerek daha fazla ödül alabilir, fakat karşısına daha

fazla ödül alacağı eylemler çıkarsa bunları da keşfedebilir. Ajan en iyi ödül aldığı eylemleri aşamalı olarak destekler. Ajanlar çevredeki özellikleri algılayabilir.

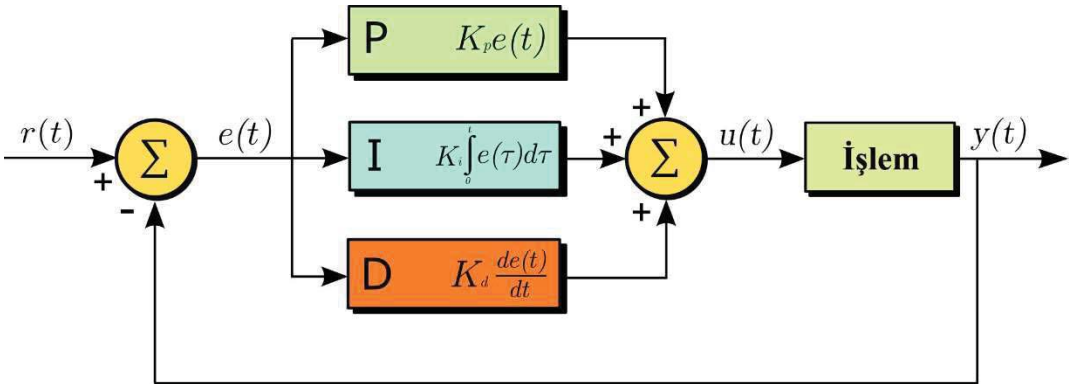
Model kavramı isteğe bağlı olarak sisteme dahil edilir. Model, ajanın eylemi gerçekleştirmeden önce, eylemin sonunda elde edeceği ödülü ve sonucu tahmin etmesini sağlar. Bunun sonucunda planlama yapılarak ajanın davranışında değişiklik meydana gelir. Pekiştirmeli Öğrenme, en fazla durum unsuruna dayanmaktadır. Politika unsuru ve değer unsuru giriş olarak kullanılır. Model unsuru hem giriş hem de çıkış olarak kullanılır. Pekiştirmeli öğrenmede ajan, aracı bir sinir ağı olarak tanımlayan bir dizi parametreye dayanarak bazı görevleri yerine getirir. Gerçekleştirilen görevlerde ajan kayıp fonksiyonunu en aza indirmek için gradyan iniş fonksiyonunu kullanır.

2.6. PID Kontrol

PID, Oransal (Proportional), İntegral (Integral), Türev (Derivative) için kullanılan bir kısaltmadır. Bu terimler hataya uygulanan üç temel matematiksel fonksiyonu açıklamaktır. PID kontrolörün ana görevi hatayı en aza indirmektir. PID, bir girdiyi alır, amaçlanan davranış sapmasını hesaplar ve amaçlanan davranışı sapmanın minimum ve daha yüksek doğruluk elde etmek için çıkışını ayarlar.

Sürücüsüz araç yoldaki şeritleri takip ederken çizgiyi tam olarak yakalayabilir. PID kontrolü kullanıldığında, araç yolun tam orta noktasından ilerlerken istikrarlı bir şekilde ve ivedilikle yolun orta noktasına kendisini sabitleyebilmesi gerekmektedir. Sürücüsüz araç yolda hareket halindeyken kontrolünü kaybetmemesi gerekmektedir. Yolda çeşitli ışık seviyelerine sahip ortamlardan geçerken, bunlardan etkilenmemesi gerekmektedir. Bu gibi problemlerin üstesinden gelmek için PID kontrolü kullanmak gerekmektedir.

Şekil 2.24'de hedef konum $r(t)$ ile gösterilmiş, ölçülen konum işe $y(t)$ ile gösterilmiştir. Zamana bağlı değişim oranı $r(t)$ ile $y(t)$'nin farkı alınarak hesaplanır. Sisteme kullanıcı tarafından oluşturulan hata değeri eklenir. Her bir kontrolör için hata (error) değeri ile değişim oranı çarpılır.



Şekil 2.24. PID Kontrolör Yapısı (Anonim 2019d)

Her bir kontrolörden çıkan sonuç toplanarak kontrolör çıkışı hesaplanır. Denklem 2.24'de PID kontrolörün çıkış değerinin hesaplanması gösterilmiştir. Burada $e(t)$ zamana bağlı hata değerini, K_P oransal kontrolörün hata değerini, K_D türev kontrolörün hata değerini K_i integral kontrolörün hata değerini, göstermektedir.

$$u(t) = K_P e(t) + K_i \int_0^t e(t')dt' + K_D \frac{de(t)}{dt} \tag{2.25}$$

Oransal (Proportional): Adından da anlaşılacağı gibi tamamen oransal çalışan bir sistemdir. Doğru orantı ya da ters orantı şeklinde olabilir. Doğru orantı olarak düşünüldüğünde, bir değişkenin değerinin artması ile artan, değişkenin değerinin azalmasıyla azalan değerdir. Ters orantı olarak düşünüldüğünde bir değişkenin değerinin azalmasıyla artan, değişkenin değerinin artmasıyla azalan değer şeklinde çalışan bir mekanizmadır. Örnek olarak bir tahterevalliyi düşündüğümüzde, bir tarafın aşağı inmesi durumunda diğer taraf yukarı çıkmaktadır. Burada dikkat edilmesi gereken husus, bir taraf bir birim aşağı indiğinde, diğer taraf bir birim yukarı çıkmaktadır. Bu da oransal katsayısını 1 olduğu anlamına gelmektedir.

Başka bir örnek olarak bir su deposundaki şamandıra sistemini düşünülebilir. Su deposunda suyun seviyesi yükseldikçe, deposunu içindeki şamandıra su ile birlikte yükselecek ve şamandıra yükseldikçe de su girişini sağlayan musluk kısılacaktır. Su deposu içerisindeki su belli bir seviyeye ulaştığında musluk zamanla kapanacaktır. Musluğun kapanıp depoya su girişinin engellenmesi şamandıranın da daha fazla yükselmesi engellenmiş olacaktır. Su deposunun içerisindeki şamandıra kolunun uzun

olması, su seviyesindeki yükselme oranı, depoya su aktarımında bulunan musluğun kapanma oranından büyük olacaktır. Bu durumda şamandıradaki birkaç birimlik yükselme, musluğa bağlı olan kola daha düşük oranda yansıyacaktır. Tam tersi durum düşünüldüğünde, su seviyesi azaldığı zaman, şamandıra aşağı doğru hareket edecektir. Bu durumda musluğa bağlı olan kol yukarı doğru kalkacak ve musluğun önündeki engeli kaldırarak su girişini hızlanmasına neden olacaktır. Büyük oranda olan su seviyesinde ki azalma, daha küçük bir oranda musluk tarafına yansıyacaktır.

Gerçek hayattan bir sistem örneği verecek olursak; x sıcaklık değerini ölçen elektronik termometre ve y soğutucu fanın şiddetini ayarlayan bir elektronik devre olarak düşünüldüğünde, sıcaklık değeri yükseldikçe fanı kontrol eden elektronik devre akım şiddetinin artmasıyla ortam ısısını azaltmaya çalışacaktır. Ortam ısısı azaldıkça, termometreden ölçülen sıcaklık değeri de düşecektir. Böylelikle fanı kontrol eden akım şiddetinin değeri de oransal olarak azalacaktır. Ortam ısısında gerçekleşecek olan değişim, giriş sisteminden gelen bilgiyle beslenip ona göre hareket edecektir. Bu tür sistemler oransal kontrollü sistemlerdir. Sürücüsüz araçlarda ise, gelen konum bilgisine göre çizginin yolun sağına kaydığı oranda sol motorun devir sayısı artacak, sağ motorun da devir sayısı azalarak çizgi istikametinde ilerlemesi sağlanacaktır. Burada x giriş değişkeni sensörlerden gelen pozisyon bilgisi olsun, motorun hızı yani çıkış y olsun. x giriş bilgisine göre y çıkış bilgisi değişecektir. Sürücüsüz aracın hareketini sağlamak için ortama K_p hata değeri eklenir.

$$\begin{aligned} e(t) &= r(t) - y(t) \\ P_{\text{çıkış}} &= K_P * e(t) \end{aligned} \tag{2.26}$$

Denklem 2.26'da oransal kontrol çıkışının hesaplanması gösterilmektedir. Burada SP hedef değeri, PV şimdiki değeri ve $e(t)$ bu iki değer arasındaki farkı, yani hatayı göstermektedir. Bu yaklaşım ile hata bulunur. Dolayısıyla çıkış değeri değiştirilir ama salınımlı gerçekleşebilir. Bunu kontrol etmek için, türev kontrolünü de kullanmak gerekir.

Türev (Derivative): PID kontrol için, Türev kontrolü bazı durumlarda çok önem arz etmektedir. Türev, eski durum ile mevcut durum arasında ki farkın ne kadar hızlı ya da yavaş değiştiğiyle ilgilenir. Türev kontrol, eski durum ile yeni durumun arasındaki farkın zamana göre yorumlanması durumudur. Sürücüsüz araç düz bir yolda ilerlediğinde, eski durum ile yeni durum arasındaki fark 0 olduğundan, türev etkisinden bahsedilmez. Araç virajlı yollara girdiğinde, yolum eğimi artacaktır ve türev kontrol aracın salınım oranlarını ayarlayacaktır. Viraj boyunca eğimin değişmediği durumlarda araç hızı sabit kalarak, aracın virajda salınım ve savrulma gibi olumsuz durumlarını türev kontrolü ortadan kaldırmaktadır.

Denklem 2.27'de türev kontrolör çıkışının hesaplanması gösterilmektedir. İşlem hatasının türevi, zaman hata eğrisinin eğimi ve K_D türev kazancındaki değişim oranı ile çarpılarak hesaplanır. Türev kontrolü sürücüsüz araçlarda, oransal kontrolü baskılamak için kullanılır. Sürücüsüz araçta hız kontrolü için ve direksiyon açı değerinin hesaplanması için türev kontrolü kullanılabilir. Kontrol sistemlerinde türev kontrolün tek başına kullanılması tercih edilen bir durum değildir. Genellikle oransal kontrol ile birlikte kullanılmaktadır.

$$D_{çıkış} = K_D \frac{de(t)}{dt} \tag{2.27}$$

Kontrol sistemlerinde türev kontrol tercih edildiğinde kazanç değerinin hassas bir şekilde ayarlanması gerekmektedir. Düşük kazanç değeri oransal kontrolü baskılayamayacaktır ve çok fazla salınıma neden olacaktır. Yüksek kazanç değeri salınımın çok yumuşak olmasını sağlayacak ve bu seferde çok fazla baskılama yapmış olacaktır.

İntegral (Integral): İntegral değeri belirli aralıktaki değerlerin toplamını ifade eder ve zamana bağlı bir fonksiyondur. Sabit bir değere sahip giriş fonksiyonlarında, artan bir grafik eğrisine sebep olabilir. İntegral kontrolü sürekli hata durumunda artan bir eğri çizecektir. Bu durum oransal kontrol uyguladığı durumlarda sonuca daha hızlı ulaşılmasını sağlayacaktır. Sürücüsüz aracın şeridi takip etmesi için integral kontrolüne oransal kontrol de dahil edilir. Bu durumda sürücüsüz araç, gelen konum bilgisi ile orantılı olarak hareket ederken, integral kontrolü ile şeridin dışına her çıktığında daha

hızlı bir şekilde orta noktaya ulaşmasını sağlayacaktır. İntegral kontrol değeri düzgün bir şekilde ayarlanmadığı takdirde salınım gerçekleşecektir.

$$I_{\text{çıkış}} = K_i \int_0^t e(t)dt \tag{2.28}$$

Denklem 2.28'de integral kontrolörün çıkışını hesaplayan formül gösterilmektedir. İntegral kontrolörün katkısı, hem hatanın büyüklüğü, hem de hatanın süresi ile orantılıdır. İntegral bir PID kontrolörün zaman içinde anlık hatalarının toplamını verir. Farklı sistemler, P kontrol, PI kontrol, PD kontrol ve PID kontrol olarak ihtiyaca göre tercih edilebilir.

3. MATERYAL VE YÖNTEM

Bu bölümde sürücüsüz araçların yolda hareket ederken direksiyon açılarının belirlenmesi için simülasyon ortamında gerçekleştirilen DL tabanlı ve PID kontrolü olan yöntemlerden bahsedilecektir. Sürücüsüz araçlar genel kamu kullanımı için uygun bir seçenek haline gelmeden önce çözülmesi gereken problemleri vardır. Eğer sürücüsüz araçlar güvenli bir şekilde yolda hareket edemezse kamuya açık yollara çıkması güvenlik açısından endişe verici olacaktır ve kamuya açık alanlarda kullanılmasına izin verilmeyecektir. Milyonlarca görüntüde hiçbir zaman %100 algılama başarısı elde edilemez. Ancak amacımız bu tespit oranını mümkün olduğunca arttırmaktır. Sürücüsüz araçların başlıca problemleri aşağıda sıralanmıştır.

- Sürücüsüz araç şeridi takip ederken şerit çizgilerini tam olarak yakalayamaması ya da şerit olmayan yollarda nasıl davranacağına karar vermesi,
- Sürücüsüz araç hareket halindeyken istikrarlı bir şekilde ivedilikle şeride kendisini (orta noktaya ya da istenen değerine) sabitleyememesi,
- Sürücüsüz aracın hızı arttığında kontrolünü kaybetmesi,
- Farklı yol koşullarında çevresel faktörlerden (farklı ışık seviyeleri, kar yağmur vb. ortamlar) etkilenmesi,

Yukarıda belirtilen sorunları çözmek çok karmaşık görünse de, belirtilen problemlerin çözümü için DL tabanlı sistemler ve PID kontrolü sistemler kullanılmaktadır. Fiziksel hasarı önlemek için sürücüsüz araç simülasyon ortamında eğitilmiş ve test edilmiştir. Sürücüsüz araç eğitimini gerçekleştirmek için Udacity Inc. tarafından hazırlanan simülatör kullanılmıştır (Udacity 2018). Veriler toplandıktan sonra öğrenme başarımının arttırılması için veri arttırma işlemi ve bazı görüntü işleme yöntemleri kullanılmıştır. Model seçimi ve hiper parametreler ayarlandıktan sonra toplanan verilerle eğitim işlemi gerçekleştirilmiştir.

3.1. Sürüş Simülatörü

Sürücüsüz araçlar herhangi bir insan etkileşimi olmadan kendi kendine hareket edebilmektedir. Sürücüsüz araçlar navigasyon, GPS, LiDAR ve kamera gibi çeşitli

teknolojiler kullanırlar. Derin Öğrenme ve PID kontrolü özerk sürüşü mümkün kılmanın yollarından biridir. Udacity tarafından hazırlanan sürücüsüz araç simülatörü, Unity kullanılarak geliştirilmiştir. Udacity sürücüsüz araç simülatörünü açık kaynak olarak geliştirmiştir (Udacity 2018). Sürücüsüz araçlara ilgi duyan araştırmacılar kendi oluşturacakları modelleri eğitmek için bu simülatörü kullanabilmektedir. Simülatör için herhangi bir sistem gereksinimi verilmemiş, fakat simülatör çalıştığında bilgisayarı oldukça yormaktadır. Bunun için öncelikle bilgisayarın işlemci, ekran kartı ve ram boyutu gibi özellikleri çok büyük bir öneme sahiptir. Daha etkili sonuçlar alabilmek için bilgisayar özelliklerinin yüksek olması gerekmektedir. Sürüş simülatörü çalıştırılabilir dosya olarak gelmektedir. Sürüş simülatörü bilgisayarda çalıştırıldıktan sonra, eğitim verilerini oluşturmak için bir araç manuel olarak sürülebilir veya makine öğrenme modeli bağımsız bir şekilde test için sürülebilir. Simülatör başlatıldığında ilk önce ekran çözünürlüğü, grafik kalitesi ve ekran seçilmelidir. Bu çalışmada 800×600 çözünürlük ve *Fantastic* grafik kalitesi seçilmiştir.

Şekil 3.1. Sürüş Simülatörünün Kuş Bakışı Görünümü

Ekran seçimi yapıldıktan sonra Şekil 3.2'de gösterilen ekran karşınıza çıkmaktadır. Sürücüsüz araç simülatörünün ana ekranından bir sahne ve bir mod seçebilmelidir. İlk önce, sahne resimlerinden birini tıklatarak bir sahne seçilmelidir. Bu tez çalışmasında Şekil 3.2'de verilen simülatör ana ekranından sol taraftaki göl kenarı sahnesi seçilmiştir. Sürüş simülatöründe seçilen sahnenin kuş bakışı görüntüsü Şekil 3.1'de verilmiştir. Aynı

parkur, direksiyon açısı ile ilgili kararın araca monte edilen kameradan gerçek zamanlı bir görüntü ile alındığı özerk sürüş modu için de kullanılmıştır. Bu işlemden sonra, eğitim modu ya da özerk moddan herhangi bir tanesi seçilmelidir. Mod düğmelerinden birine tıklandığında, başlangıç konumunda bir araç belirecektir. Şekil 3.3'te eğitim modu ve sahne seçildikten sonra karşımıza çıkacak ekran görüntüsü verilmiştir. Bu ekran seçildiğinde kullanıcılar kendilerini geliştirmek için test sürüşü yapabilmektedir.

Şekil 3.2. Simülatör Ana Ekranı

Daha önce herhangi bir araç simülatörü kullanmamış ya da araba oyunu oynamamış kullanıcılar, bir süre simülatör ortamında araç kullandıktan sonra, sürücüsüz aracın eğitimi için veri toplama işlemini burada gerçekleştirebilmektedir. Eğitim modunda, sürüş davranışını kaydetmek için araç manuel olarak sürülmelidir. Makine öğrenme modelinizi geliştirmek için kaydedilen görüntüleri kullanılabilir. Aracı ileri yönde hareket ettirmek için klavyeden "W" karakteri, sola doğru yönlendirmek için "A", sağa doğru yönlendirmek için "D" ve aracın hızını yavaşlatmak ya da geri doğru hareket ettirmek için "S" karakteri kullanılır. Sürüş davranışının kaydedilmesi için ekranın sağ üst köşesindeki kırmızı renkte olan "Record" düğmesi tıklanır ve bilgisayarda hangi klasöre

kaydedileceği seçilir. Kaydı durdurmak için tekrardan aynı yere tıklanmalıdır. Sürüş simülatöründeki araçta sağ, sol ve ortada olmak üzere üç tane kamera vardır.

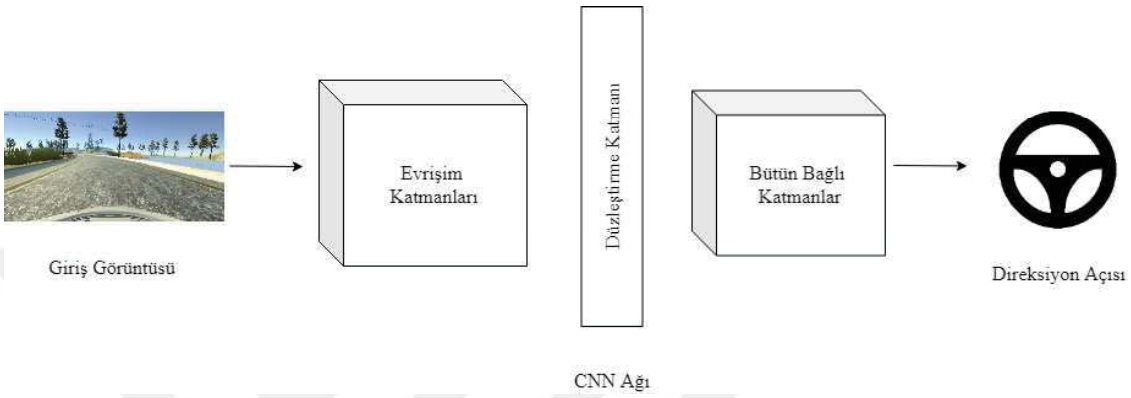
Şekil 3.3. Simülatör Ana Ekranı

Eğitim verilerini toplamak için sürüş gerçekleştiğinde her bir kameradan görüntü karesi kaydedilir. Eğitim modunda, simülatör üç kamera, hız değeri, gaz kelebeği değeri, fren değeri ve direksiyon açılarını da görüntü kareleriyle birlikte kaydeder. Bu veriler gerçekleştirilen model ile eğitildikten sonra özerk sürüş gerçekleştirilebilir. Özerk modda, sürücüsüz aracın, yol şeritlerinin dışına çıkmadan ve göle düşmeden ne kadar iyi hareket ettiğini görmek için DL modeli test edilir.

3.2. Derin Öğrenme Tabanlı Sürüş

Son yıllarda sürücüsüz araçların geliştirilmesine yönelik yarış muazzam bir şekilde hızlanmıştır. Pek çok bileşen sürücüsüz bir araç oluşturur ve en önemlilerinden bazıları ona güç veren sensörler ve DL yazılımıdır. Hesaplama yeteneklerindeki artışla, artık önemli detayları öğrenebilen ve aracın beyni haline gelebilecek, araca bir sonraki kararlar hakkında talimat verebilecek karmaşık ve derin sinir ağları kullanılmaktadır. Bu tez

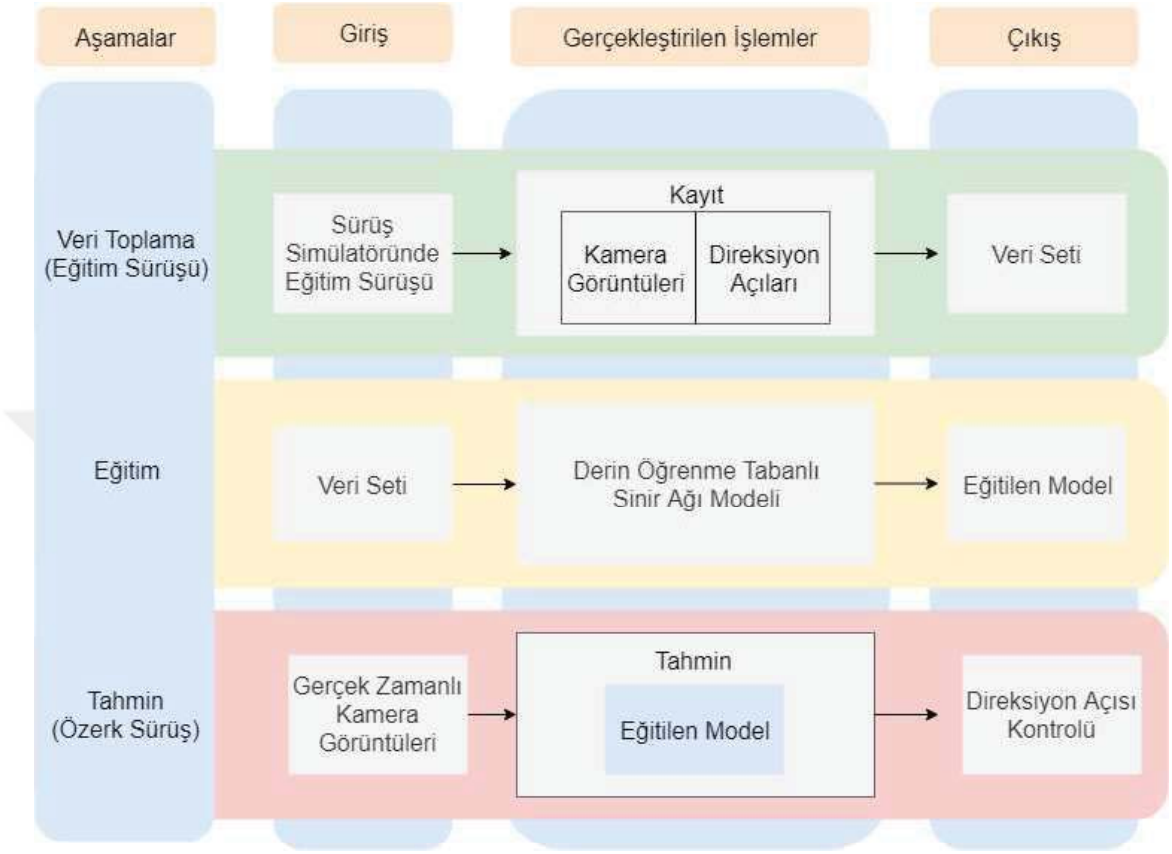
çalışmasında, direksiyon açısını tahmin etmek ve sürücüsüz aracın simülatör ortamında kendi kendine hareket etmesi DL modeli oluşturularak sağlanmıştır. Oluşturulan modelde Tensorflow tabanlı Keras kütüphanesi kullanılmıştır. Oluşturulan modele kameralardan elde edilen görüntüler verilerek aracın direksiyon açısı tahmin edilmiştir. Şekil 3.4'te kameradan gelen görüntülerin sinir ağında eğitiminin şeması gösterilmiştir.



Şekil 3.4. Derin Öğrenme Eğitim Modeli

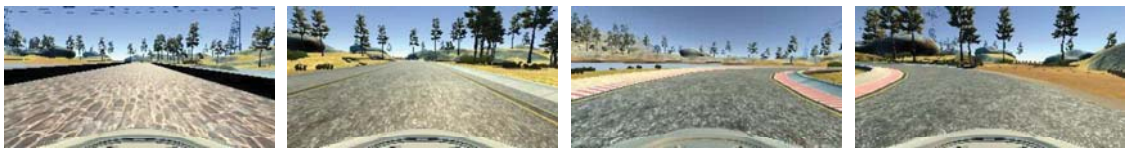
Sürücüsüz aracın özerk hareketi için süreç; veri toplama, toplanan verilerin oluşturulan model kullanılarak eğitimi ve sürücüsüz aracın test edilmesi olmak üzere üç bölümde ele alınmıştır. İlk olarak, DL modelinin eğitimi için kullanılacak verilerin toplanması işlemi gerçekleşmektedir. Simülatör sürüş ortamında bir kullanıcı tarafından aracın sürülmesi gerekmektedir. Sürüş sırasında sürüş simülatörü mevcut kameralarla görüntüleri ve direksiyon açılarını kaydetmektedir. Eğitim sürüşü modunda elde edilen veriler, kamera görüntüleri ve her karedeki direksiyon açısı değerleridir. Görüntüler özellik seti ve direksiyon açıları etiket seti olarak kullanıldı. Sadelik nedeniyle aracın hızı PID kontrolör kullanılarak sabitlendi. Bu yaklaşım kullanılarak toplanan veriler, daha fazla insan etkileşimi olmadan, yalnızca girdi verilerine dayanarak sürüş yapmayı öğrenecek sinir ağını eğitmek için kullanılmıştır. Bu teknik aynı zamanda davranış klonlaması olarak da adlandırılmaktadır. Eğitim verileri toplandıktan sonra, otonom sürüş için sinir ağı, direksiyon açısını tahmin etmek için bu veri seti üzerinde eğitildi. Son olarak, eğitilmiş model, aynı simülatör ortamında gerçek zamanlı bir özerk sürüş olan çıkarım için kullanıldı. Sürücüsüz aracın insan etkileşimi olmadan, kendi kendine hareketi sırasındaki başarı ölçütü, sürüş simülatöründe her zaman yolda kalması ve yolun ortasından hareket etmesidir. DL tabanlı sürücüsüz araç için kullanılan genel çerçeve Şekil 3.5'te verilmiştir.

Sürüş simülatöründeki parkurun farklı özellikleri, sinir ağı modelinin eğitimi için zorluklar ortaya koymaktadır.



Şekil 3.5. DL Tabanlı Sistem Çerçevesi (Kocić, 2019)

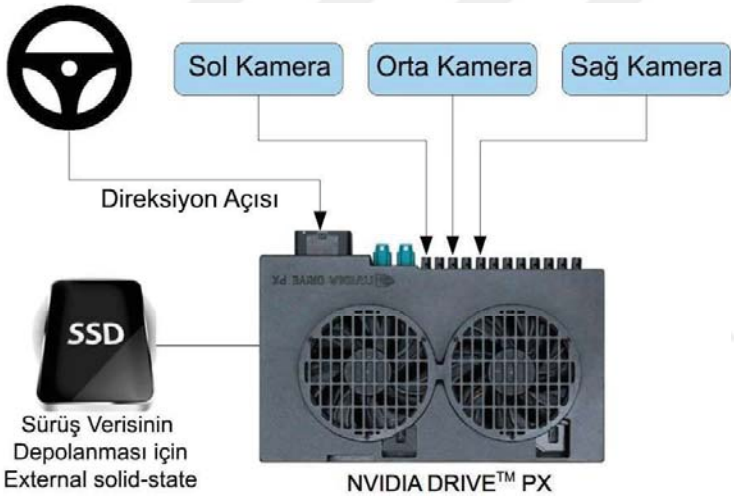
Örneğin, modelin keskin dönüşlerde, farklı dokularda ve yolun farklı sınırlarında aracı nasıl kullanacağını öğrenmesi gerekiyor. Özerk sürüş için sürüş simülatöründeki parkurun en zorlu kısımları, Şekil 3.1'de görüldüğü gibi, köprüden hemen sonra gelen üç keskin dönüşlü virajlardır. Sürüş pisti, genellikle virajlarda kırmızı ve beyaz şeritlerin arasında ve pist düz bir şekilde ilerlerken banketler arasında kalan yol olarak belirtilmiştir.



(a) (b) (c) (d)

Şekil 3.6. Merkezi Kamera ile Farklı Karelerde Yakalanan Giriş Verileri

Orta kamerayla kaydedilmiş farklı yol özelliklerini gösteren görüntü karelerinin örnekleri Şekil 3.2.3'de verilmiştir. Göl üzerindeki köprüyü gösteren yolun farklı dokusu 3.6.a'da gösterilmiştir. Köprüde yol tuğla ile kaplıyken, pistin diğer kısımlarında çoğunlukla asfalt yol bulunur. Ayrıca, pistin bu kısmının kenarlarında alçak bir duvar vardır. Şekil 3.6.b'de pist düz olarak ilerlerken banketler arasında ve farklı parlaklık değerine sahip görüntü karesi gösterilmiştir. Sürüş pistinin kırmızı ve beyaz şeritler arasında kalan kısmı Şekil 3.6.c'de gösterilmiştir. Şekil 3.6.d'de gösterildiği gibi pistin belirli bir kısmında tek taraflı şerit ve diğer tarafında toprak bir yol bulunmaktadır. Sürüş simülatöründeki yolun bu farklı özellikleri, farklı senaryolarda bağımsız özerk sürüşe yol açan modelin daha iyi genelleştirilmesine yardımcı olur.



Şekil 3.7. Veri Toplama Sistemi (Bojarski ve ark. 2016)

Veri Toplama: Veri toplama işlemi, araç sürüş simülatör pistinde eğitim modundayken yapılır. Sürüş simülatöründe, veri toplama aracının ön camına üç kamera monte edilmiştir. Kameralardan biri aracın ortasına, diğerleri ise aracın sağ ve sol taraflarına yerleştirilmiştir. Şekil 3.9'da kameralardan aynı anda bir görüntü karesi için alınan görüntüler verilmiştir. Şekil 3.7'de gösterilen, NVIDIA tarafından geliştirilmiş sürücüsüz araca yerleştirilen veri toplama sisteminden esinlenerek geliştirilmiş sürüş araç simülatöründe, eğitim sürüşünün sonunda, her bir karedeki görüntü başlıkları ve direksiyon açısı değerleri hakkında bilgi içeren tabloyla birlikte görüntüler de kaydedilmektedir. Direksiyon açı değeri -1 ile 1 arasında oluşmaktadır. Direksiyon açısının değeri negatif ise sola doğru ve değer pozitif ise sağa doğru bir yönelim

gerçekleşmiştir. Veri toplama sürecinde araç kameralarından alınan görüntüler bir klasöre kaydedilmektedir. Her bir görüntü karesi için kaydedilen kamera görüntüleri arasındaki ufak farklar modelin daha iyi bir genellemesini sağlar. Kaydedilen görüntü kareleriyle eşleştirilmiş direksiyon açısı, aracın hız değeri, gaz ve fren kelebeğinin açıları log.csv dosyasında tutulmaktadır. Şekil 3.8'de csv dosyasının görseli verilmiştir.

	A	B	C	D	E	F	G
1	merkez	sol	sag	direksiyon	gaz	geri	hız
2	IMG\center_2019_08_27_17_24_41_330.jpg	IMG\left_2019_08_27_17_24_41_330.jpg	IMG\right_2019_08_27_17_24_41_330.jpg	-0.2	0	0	6.944.267
3	IMG\center_2019_08_27_17_24_41_405.jpg	IMG\left_2019_08_27_17_24_41_405.jpg	IMG\right_2019_08_27_17_24_41_405.jpg	-0.35	0	0	6.882.987
4	IMG\center_2019_08_27_17_24_41_482.jpg	IMG\left_2019_08_27_17_24_41_482.jpg	IMG\right_2019_08_27_17_24_41_482.jpg	0	0	0	6.817.351
5	IMG\center_2019_08_27_17_24_41_550.jpg	IMG\left_2019_08_27_17_24_41_550.jpg	IMG\right_2019_08_27_17_24_41_550.jpg	0	0	0	6.762.021
6	IMG\center_2019_08_27_17_24_41_627.jpg	IMG\left_2019_08_27_17_24_41_627.jpg	IMG\right_2019_08_27_17_24_41_627.jpg	0	0	0	6.707.639
7	IMG\center_2019_08_27_17_24_41_694.jpg	IMG\left_2019_08_27_17_24_41_694.jpg	IMG\right_2019_08_27_17_24_41_694.jpg	0	0	0	6.667.165
8	IMG\center_2019_08_27_17_24_41_761.jpg	IMG\left_2019_08_27_17_24_41_761.jpg	IMG\right_2019_08_27_17_24_41_761.jpg	0	0	0	6.626.937
9	IMG\center_2019_08_27_17_24_41_831.jpg	IMG\left_2019_08_27_17_24_41_831.jpg	IMG\right_2019_08_27_17_24_41_831.jpg	0	0.20	0	6.687.139
10	IMG\center_2019_08_27_17_24_41_909.jpg	IMG\left_2019_08_27_17_24_41_909.jpg	IMG\right_2019_08_27_17_24_41_909.jpg	0	0.44	0	6.974.247
11	IMG\center_2019_08_27_17_24_41_976.jpg	IMG\left_2019_08_27_17_24_41_976.jpg	IMG\right_2019_08_27_17_24_41_976.jpg	0	0.64	0	7.340.194
12	IMG\center_2019_08_27_17_24_42_045.jpg	IMG\left_2019_08_27_17_24_42_045.jpg	IMG\right_2019_08_27_17_24_42_045.jpg	0	0.67	0	7.803.725
13	IMG\center_2019_08_27_17_24_42_114.jpg	IMG\left_2019_08_27_17_24_42_114.jpg	IMG\right_2019_08_27_17_24_42_114.jpg	-0.05	0.43	0	8.272.465
14	IMG\center_2019_08_27_17_24_42_183.jpg	IMG\left_2019_08_27_17_24_42_183.jpg	IMG\right_2019_08_27_17_24_42_183.jpg	-0.2	0.26	0	845.912
15	IMG\center_2019_08_27_17_24_42_253.jpg	IMG\left_2019_08_27_17_24_42_253.jpg	IMG\right_2019_08_27_17_24_42_253.jpg	-0.4	0.01	0	8.486.005
16	IMG\center_2019_08_27_17_24_42_324.jpg	IMG\left_2019_08_27_17_24_42_324.jpg	IMG\right_2019_08_27_17_24_42_324.jpg	0	0	0	8.421.688
17	IMG\center_2019_08_27_17_24_42_396.jpg	IMG\left_2019_08_27_17_24_42_396.jpg	IMG\right_2019_08_27_17_24_42_396.jpg	0	0	0	8.351.815
18	IMG\center_2019_08_27_17_24_42_466.jpg	IMG\left_2019_08_27_17_24_42_466.jpg	IMG\right_2019_08_27_17_24_42_466.jpg	0	0	0	8.301.245
19	IMG\center_2019_08_27_17_24_42_538.jpg	IMG\left_2019_08_27_17_24_42_538.jpg	IMG\right_2019_08_27_17_24_42_538.jpg	0	0	0	8.234.492
20	IMG\center_2019_08_27_17_24_42_611.jpg	IMG\left_2019_08_27_17_24_42_611.jpg	IMG\right_2019_08_27_17_24_42_611.jpg	0	0	0	8.168.305
21	IMG\center_2019_08_27_17_24_42_679.jpg	IMG\left_2019_08_27_17_24_42_679.jpg	IMG\right_2019_08_27_17_24_42_679.jpg	-0.15	0	0	8.113.885
22	IMG\center_2019_08_27_17_24_42_752.jpg	IMG\left_2019_08_27_17_24_42_752.jpg	IMG\right_2019_08_27_17_24_42_752.jpg	-0.35	0	0	8.021.576
23	IMG\center_2019_08_27_17_24_42_820.jpg	IMG\left_2019_08_27_17_24_42_820.jpg	IMG\right_2019_08_27_17_24_42_820.jpg	0	0	0	7.940.827
24	IMG\center_2019_08_27_17_24_42_889.jpg	IMG\left_2019_08_27_17_24_42_889.jpg	IMG\right_2019_08_27_17_24_42_889.jpg	0	0	0	7.866.215
25	IMG\center_2019_08_27_17_24_42_958.jpg	IMG\left_2019_08_27_17_24_42_958.jpg	IMG\right_2019_08_27_17_24_42_958.jpg	0	0	0	7.818.098
26	IMG\center_2019_08_27_17_24_43_027.jpg	IMG\left_2019_08_27_17_24_43_027.jpg	IMG\right_2019_08_27_17_24_43_027.jpg	0	0	0	7.755.153
27	IMG\center_2019_08_27_17_24_43_095.jpg	IMG\left_2019_08_27_17_24_43_095.jpg	IMG\right_2019_08_27_17_24_43_095.jpg	-0.1	0	0	7.706.478
28	IMG\center_2019_08_27_17_24_43_163.jpg	IMG\left_2019_08_27_17_24_43_163.jpg	IMG\right_2019_08_27_17_24_43_163.jpg	-0.25	0	0	7.646.973
29	IMG\center_2019_08_27_17_24_43_230.jpg	IMG\left_2019_08_27_17_24_43_230.jpg	IMG\right_2019_08_27_17_24_43_230.jpg	-0.45	0	0	7.549.898

Şekil 3.8. CSV Dosyası Görseli

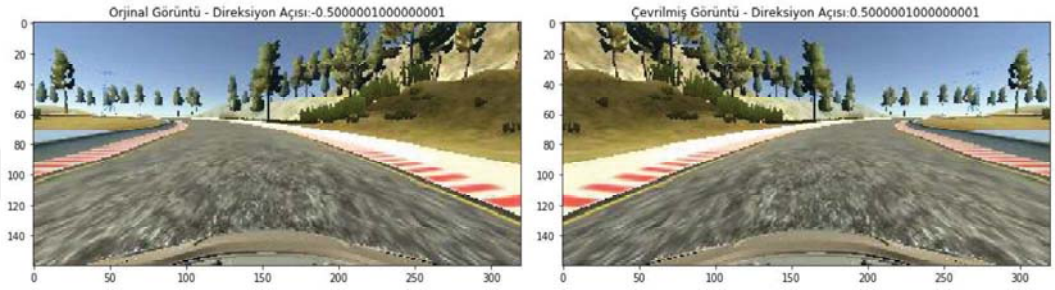
Eğitim modunda, modelin doğru özellikleri öğrenmesini sağlayacak kaliteli örneklere sahip olmak için, özerk sürüş sırasında aracın otonom hareket etmesini beklediğimiz şekilde sürülmelidir. Veri toplama sırasında amaç aracın yolun ortasında hareket ettirilmesidir. Eğitim sürüşü gerçekleştirilirken, yüksek eğime sahip virajlarda özel dikkat gösterilmiştir. Çünkü oluşturulan modelin virajlarda nasıl davranacağını öğrenmesi çok önemlidir.

(a) (b) ©

Şekil 3.9. Kamera Görüntüleri **a.** Sol Kamera **b.** Orta Kamera **c.** Sağ Kamera

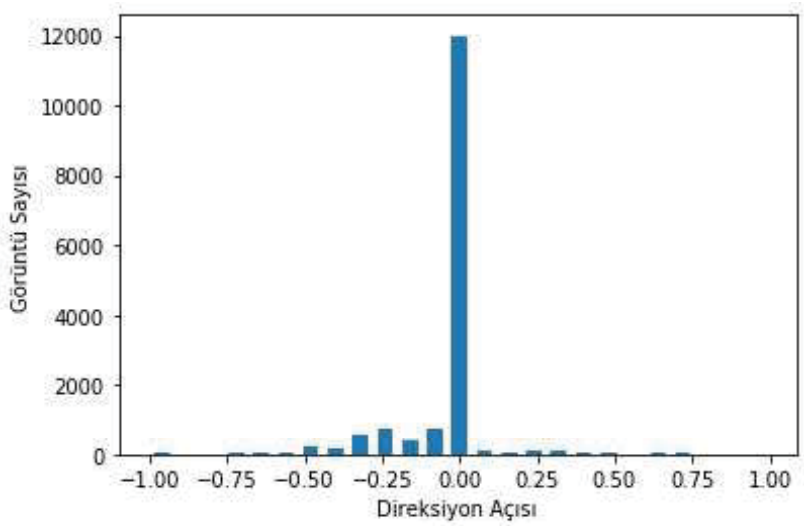
Veri Çoğaltma (Data Augmentation): Veri toplama için, üç kameradan alınan verilerin toplandığı 20 dakikalık eğitim modu sürüşü sırasında veriler kaydedildi. 20 dakikadan

sonra bile, toplanan veri kümesi nispeten küçük olmasından dolayı, veri çoğaltma teknikleri uygulanmıştır. İlk olarak verilerin direksiyon açılarına bakıldığında, direksiyon açısının 0 olduğu durumlar, yani aracın düz olarak ilerlediği durumlar yoğunluktadır. Şekil 3.11'de veri kümesinin histogram grafiği verilmiştir. Histogramdaki dağılıma bakılarak, dengeli bir veri kümesi oluşturmak ve öğrenmenin başarısını arttırmak için, bir eşik değeri (threshold) seçilerek, herhangi bir direksiyon açısı dağılımı 400'ün üzerinde olan değerler kırpılarak eğitim verisinden çıkarılmıştır.



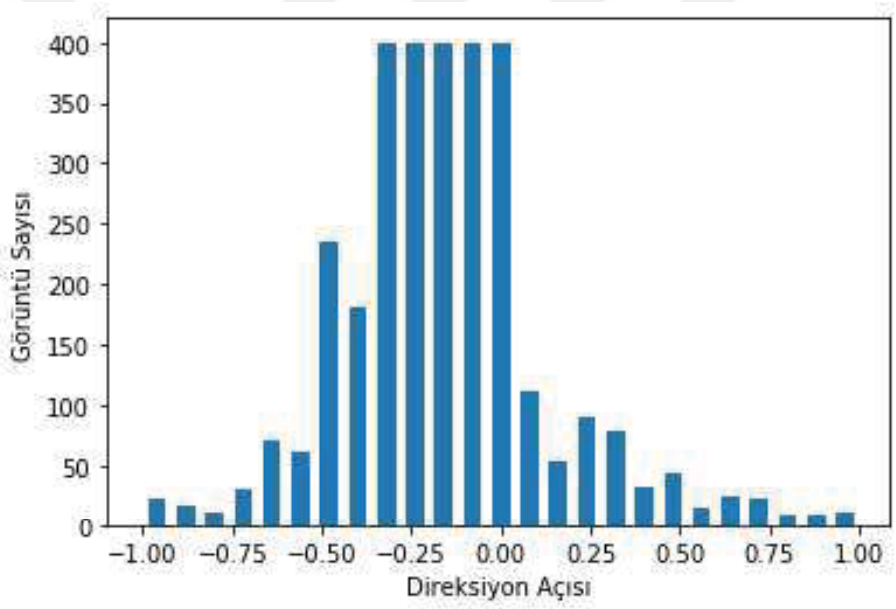
Şekil 3.10. Orijinal ve Çevrilmiş Görüntü Karesi

Eşik değeri uygulandıktan sonra oluşan histogram dağılımı Şekil 3.12'de gösterilmiştir. Sinir ağının öğrenmesi gereken en önemli özelliklerden biri virajlardır. Veri kümesi boyutunu iki katına çıkarmak ve özellikle virajlardaki başarım oranını arttırmak için, Şekil 3.10'da gösterildiği gibi görüntüler dikey olarak çevrilerek ve direksiyon açısı ters çevrilerek veri arttırma işlemi gerçekleştirilmiştir.



Şekil 3.11. Bütün Verinin Direksiyon Açılarının Histogramı

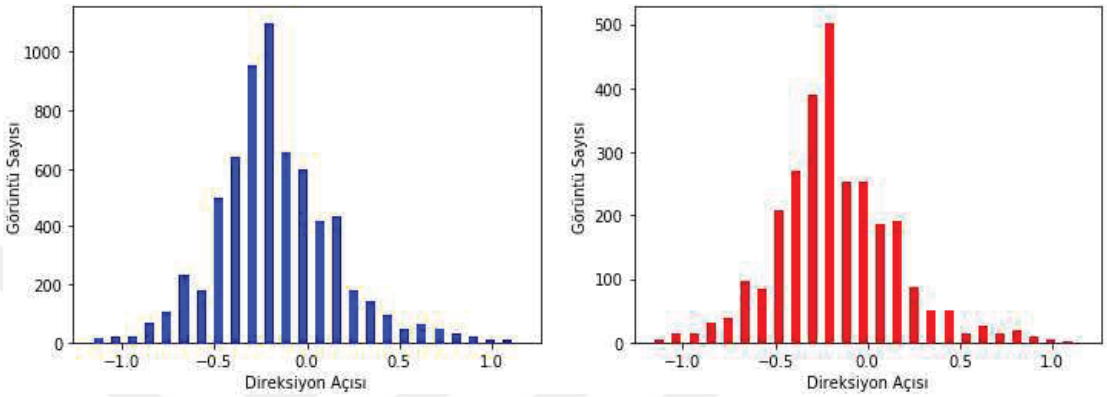
Eğitim verisinin sayısını iki katına çıkarmanın yanı sıra, görüntüleri çevirerek veri artırma, modelin başarısını arttırmasını sağlamıştır. Pist bir halka gibi olduğundan, bir yönde diğerinden daha fazla viraj vardır. Eğitim sırasında toplanan veriler arasında, solda daha fazla viraj bulunmaktadır. Şekil 3.12'de görüldüğü gibi sola doğru yönelim daha fazla oluştur. Toplanan verilerde yalnızca görüntülerde çevirme işlemi uygulanarak veri arttırma işlemi olmadan kullanırsak, o çerçevenin temel gerçeği düz kalsa bile model sola yönlendirmeyi öğrenecektir. Yatay bir çevirme kullanarak görüntüleri yansıtarak ve direksiyon açılarını ters çevirerek veri artırma, modelin hem saatin tersi yönünde hem de saat yönünün tersine yöneldiği düşünülen dengeli veri setleri sağlamıştır.



Şekil 3.12. Eşik Değeri Uygulandıktan Sonra Direksiyon Açılarının Histogramı

Veri kümesi: 20 dakikalık eğitim sürüşünde, $320 \times 160 \times 3$ çözünürlükte toplam örnek sayısı 46.809 görüntüden oluşmaktadır. Kaydedilen her görüntü 320 piksel yüksekliğinde, 160 piksel genişliğinde ve üç kanal derinliğinde (Red Green Blue, RGB) görüntüden oluşmaktadır. Kaydedilen bir görüntünün ortalama bellek boyutu yaklaşık 20 KB'dir. Her görüntüye karşılık gelen, -1 ile 1 aralığında normalize edilmiş direksiyon açısı değeriyle eşleştirilmiştir. Veri çoğaltma işlemleri uygulandıktan sonra, veri setimizdeki toplam örnek sayısı 93.618 olmuştur. Direksiyon açılarının dengelenmesi için belli bir eşik değeri uygulanmış ve direksiyon açısının 0 olduğu çoğu görüntü silinmiştir.

Kırpma işleminden sonra veri setinde 28.206 tane görüntü kalmıştır. Veriler eğitim ve doğrulama kategorilerine ayrılmıştır. Çizelge 3.1'de gösterildiği gibi, eğitim için verilerin %70'i, 19.743 görüntü, örnek ve doğrulama için verilerin %30'u, 8.463 görüntü eğitim için ayrılmıştır. Eğitim ve doğrulama verilerindeki direksiyon açı değerlerinin histogram grafiği Şekil 3.13'de verilmiştir.



(a) (b)

Şekil 3.13. Eğitim ve Doğrulama Histogramı **a.** Eğitim Verisi **b.** Doğrulama Verisi

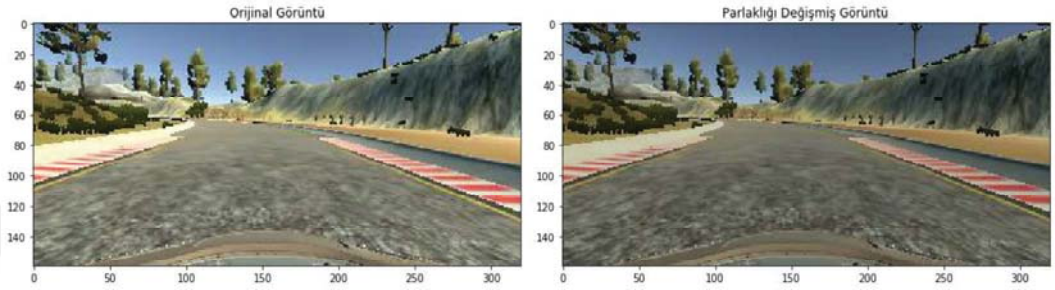
Test sürüşü model oluşturulduktan sonra gerçek zamanlı olarak yapılmıştır. Orta kameradan elde edilen gerçek zamanlı görüntüler, araç direksiyon açısının kontrolü için kullanılan modele sürekli olarak gönderilmiştir. Modelin başarım değerlendirilmesi, aracın parkur boyunca özerk sürüş yapıp yapmadığı gözlemlenerek yapılmıştır. Eğer araç yoldan çıkmışsa, başarısız özerk sürüş olarak kabul edilmiştir.

Çizelge 3.1. Veri Kümesi

	Eğitim	**Doğrulama**	**Toplam**
Örnek Sayısı	19.743	8.463	28.206
Toplam Veri Kümesi Yüzdesi	%70	%30	%100

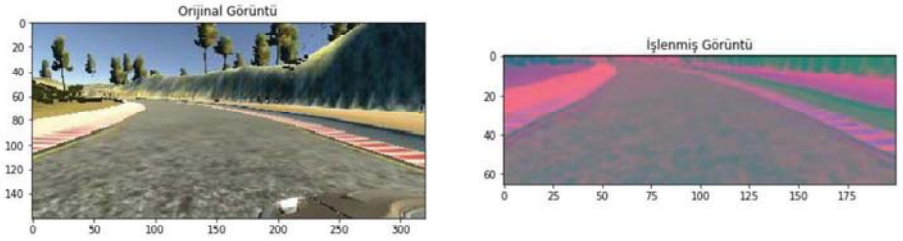
Veri Ön İşleme: Derin sinir ağını eğitmek için üç kameradan da elde edilen görüntüler kullanılmıştır. Bu görüntülerin tümü farklı konumlardan benzer bir görsel yakalamıştır. Bir kamera yerine üç kamera kullanmanın avantajı, üç kat daha fazla veri elde etmek ve araç yana doğru kaymaya başladığında yolun ortasına geri dönmek için daha iyi

performans sağlar. Direksiyon açısı, orta kameraya karşılık gelen belirli bir çerçeveden üç görüntü ile eşleştirilmiştir. Sol ve sağ kameralardan gelen görüntüler, sırasıyla yolun sol veya sağ tarafında kaydırılan bir görüş alanına sahiptir. Kameralardan elde edilen görüntüler üzerinde kırpma işlemi uygulanmıştır. Özerk sürüş için Şekil 3.15'de görüldüğü gibi görüntünün önemli bilgiye sahip olmayan kısımlarını görüntü karelerinden kırpılmıştır.



Şekil 3.14. Orijinal ve Parlaklığı Değiştirilmiş Görüntü Karesi

Şekil 3.11'de gösterilen veri dengesizliği nedeniyle, ham veriler üzerinde geliştirilen her öğrenme algoritmasının sadece direksiyon değerini 0 tahmin etmeyi öğrenecektir. Ayrıca, test sürüşün, eğitim sürüşünden tamamen farklı olduğu görülebilir. Çeşitli veri büyütme biçimleri yanlış öğrenme probleminin üstesinden gelebilmektedir.

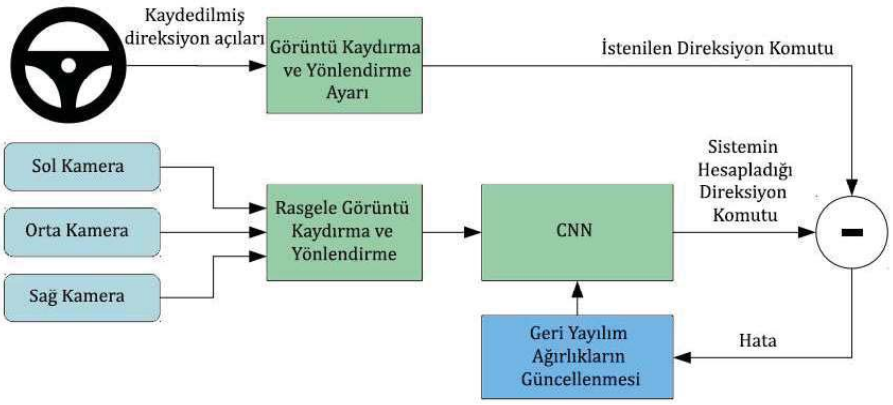


Şekil 3.15. Orijinal ve Önişlemden Geçirilmiş Görüntü Karesi

Eğitim setini güçlendirmek için sağ ve sol kameralardan elde edilen görüntülerde veri artırma işlemi uygulanabilir. Şekil 3.14'de gösterildiği gibi görüntü parlaklığını değiştirmek için, belirli aralıktaki rastgele bir değer ile eleman bazında çarpılır. Aralık ne kadar genişse, orijinal olan görüntü karelerinden ortalama olarak artırılmış görüntü kareleri o kadar farklı olacaktır. Sinir ağına görüntüler verilmeden önce, Şekil 3.15'de gösterildiği gibi RGB renk uzayından oluşan her görüntü karesi YUV renk uzayına

dönüştürülür. Görüntü karesi eğitime girmeden önce görüntü çözünürlüğü $66 \times 200 \times 3$ olarak tekrardan boyutlandırılmıştır.

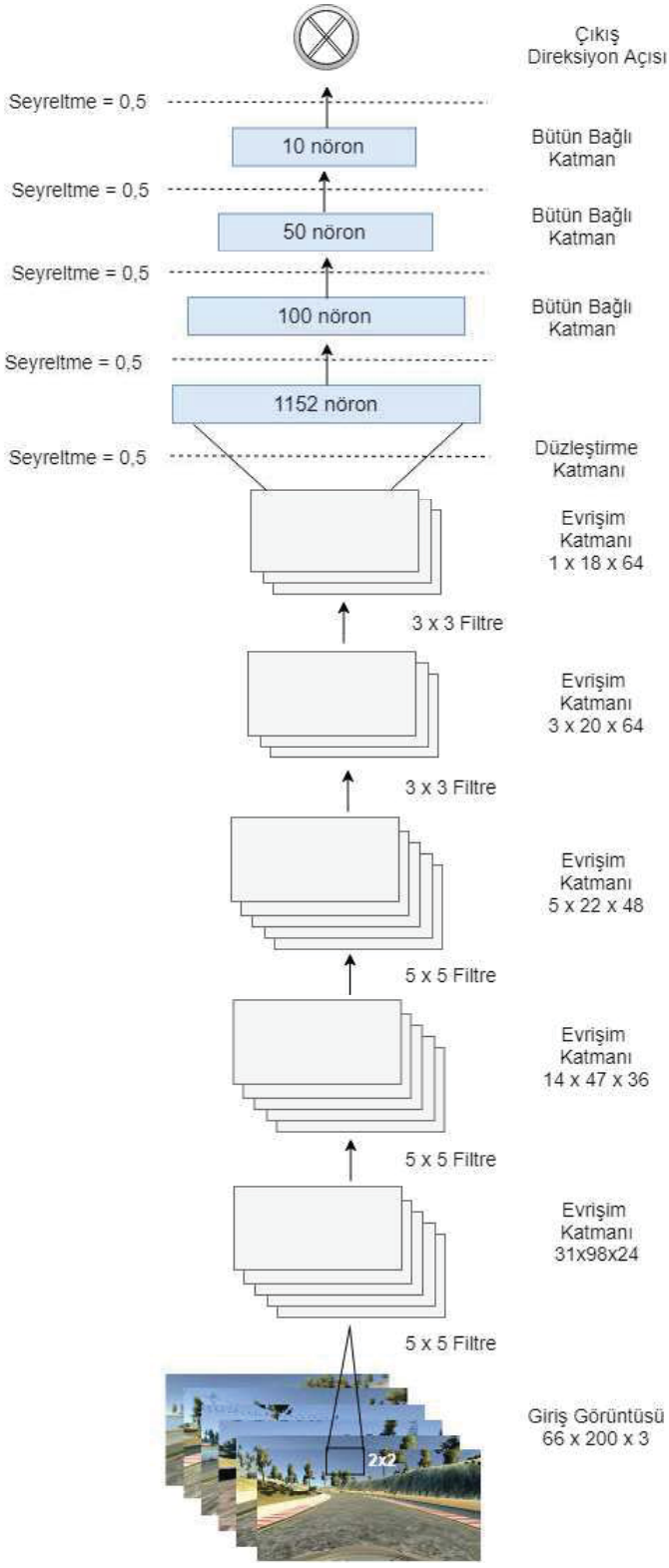
Önerilen Yöntem: Eğitim sisteminin bir blok şeması Şekil 3.16'da gösterilmektedir. Sürücüsüz araç hareket halindeyken aracın üç kamerasından toplanan görüntüler rastgele kaydırılır ve döndürülür ve ardından sinir ağı beslenir. Bu girdilere dayanarak, sinir ağı tek bir değer, direksiyon açısını üretir. Temel olarak, giriş görüntülerine dayanarak, sinir ağı aracın hangi açıdan yönlendirilmesi gerektiğine karar verir. Bu çıktı değeri, sinir ağının kararındaki hatayı hesaplamak için insan sürüşünden toplanan direksiyon verileriyle karşılaştırılır. Model, bu hatayı azaltmak için ağırlıkları geriye yayılım algoritmasını kullanarak günceller.



Şekil 3.16. Sürücüsüz Araç için CNN Eğitimi (Bojarski ve ark. 2016)

Bu tez çalışmasında farklı model ve farklı veri kümeleri kullanılarak DL tabanlı sürücüsüz araç eğitimi gerçekleştirilmiştir. Modelin eğitimi, Intel(R) Core(TM) i7 2600@3.40 GHz işlemci, 16 GB ram ve NVDIA GeForge GTX 1080 grafik kartı olan bilgisayarda gerçekleştirilmiştir. Bulgular bölümünde DL modellerinin performans değerleri detaylı olarak verilecektir. Kullanılan modellerde en başarılı sonuç NVDIA modelinden esinlenerek oluşturulan model CNN mimarisinde gerçekleşmiştir. Bu bölümde en başarılı sonuç alınan model anlatılacaktır. Oluşturulan modelin mimari yapısı Şekil 3.17'de verilmiştir. CNN modelinde elu aktivasyon fonksiyonu kullanılmıştır. Giriş görüntüleri üzerinde ilk önce normalize işlemleri uygulanmıştır ve sonra normalize görüntüler CNN'e giriş olarak verilmiştir. CNN'in ilk evrişim katmanında 5×5'lik 24

tane filtre bulunmaktadır. Evrişim katmanında görüntülerin farklı özellikleri çıkartılmıştır. Oluşturulan CNN modeli mimarisinde beş tane evrişim katmanı ile bir tane düzleştirme



Şekil 3.17. CNN Modeli Mimarisi (Bojarski ve ark. 2016)

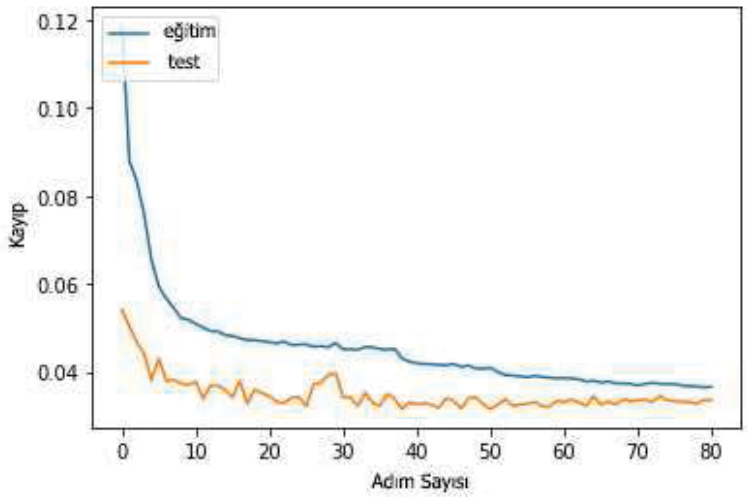
katmanı ve dört tane de tam bağlı katman bulunmaktadır. İkinci erişim katmanında 5×5 boyutunda 36 tane filtre, üçüncü evrişim katmanında 5×5 boyutunda 48 tane fitre, dördüncü evrişim katmanında 3×3 boyutunda 64 tane filtre ve en son evrişim katmanında 3×3 boyutunda 64 tane filtre kullanılmıştır. Evrişim katmanlarından sonra aşırı uyumu engellemek için 0,5 seyreltme uygulanmış ve sonrasında matris düzleştirme katmanında tek boyutlu bir vektöre dönüştürülmüştür. Düzleştirme katmandaki tüm düğümlerin tek bir çıktıyla birleştirildiği, direksiyon açısı değerlerini öngören, bütün bağlantılı katman uygulanmıştır. Düzleştirme katmanından sonra 0,5 seyreltme uygulanarak bütün bağlı katmanlar kullanılmıştır. Çizelge 3.2'de CNN katman modelleri ve kullanılan parametreler gösterilmiştir. Modelde toplam parametre sayısı 252.219'dir.

Çizelge 3.2. CNN Katman Modelleri ve Parametreler

Katman	Çıkış Modeli	Parametre
Evrişim 1	$31 \times 98 \times 24$	1824
Evrişim 2	$14 \times 47 \times 36$	21.636
Evrişim 3	$5 \times 22 \times 48$	43.248
Evrişim 4	$3 \times 20 \times 64$	27.712
Evrişim 5	$1 \times 18 \times 64$	36.928
Seyreltme 1	$1 \times 18 \times 64$	0
Düzleştirme	1 152	0
Bütün Bağlı 1	100	115.300
Seyreltme 2	100	0
Bütün Bağlı 2	50	5.050
Seyreltme 3	50	0
Bütün Bağlı 3	10	0
Seyreltme 4	10	0
Bütün Bağlı 4	1	11

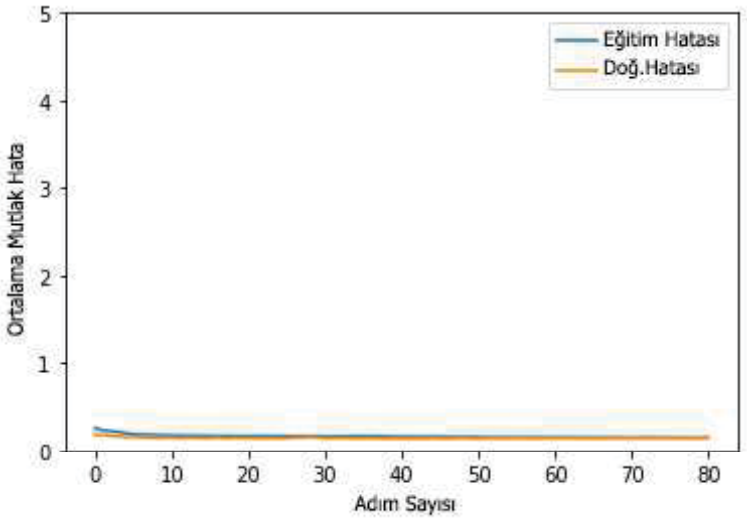
Oluşturulan modelde 27 milyona civarında bağlantı bulunmaktadır. Oluşturulan modelde en iyileme yöntemi olarak $Adam\ En\ İyileme$ kullanılmıştır. Yığın parametresi 100 olarak belirlenmiştir. İlk denemelerde ReLU aktivasyon fonksiyonu kullanılmış, fakat modelin başarısının düşük çıkması sonucu elu aktivasyon fonksiyonu kullanılmıştır.

Önerilen model, 100 adım için eğitilmiş, fakat erken durdurma kullanılarak 549 dakika 9 saniyede eğitim 81 adımda eğitim tamamlanmıştır.



Şekil 3.18. Kayıp Grafiği

Model eğitiminden sonra, modelin başarısı test edilmiştir. Şekil 3.18'de başarım oranı en yüksek çıkan modelin kayıp grafiği gösterilmiş ve oluşturulan modelin kaybı 0,0334 olarak ölçülmüştür.



Şekil 3.19. Ortalama Mutlak Hata Grafiği

Şekil 3.19 ortalama mutlak hata grafiğinde de belirtildiği gibi geliştirilen CNN modelinin ortalama mutlak hatası 0,1431 olarak ölçülmüştür. Modelin hata oranının düşük olması modelin gerçek zamanlı olarak sürüş simülatöründe de başarım oranının yüksek olmasını

sağlayacaktır. DL tabanlı oluşturulan modellerin her biri daha önce toplanan aynı veri seti ve farklı veri seti kullanılarak eğitilmiştir. DL tabanlı oluşturulan modellerin her biri daha önce toplanan aynı veri seti ve farklı veri seti kullanılarak eğitilmiştir.

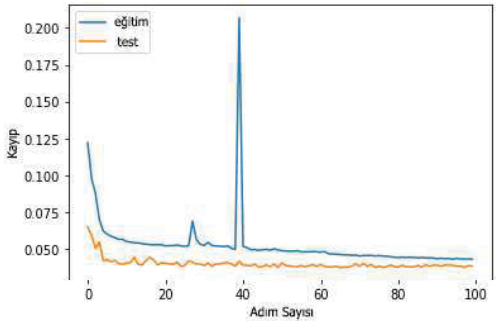
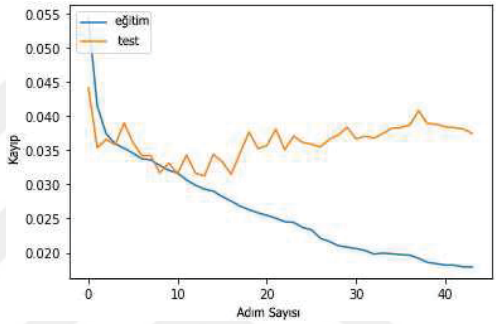
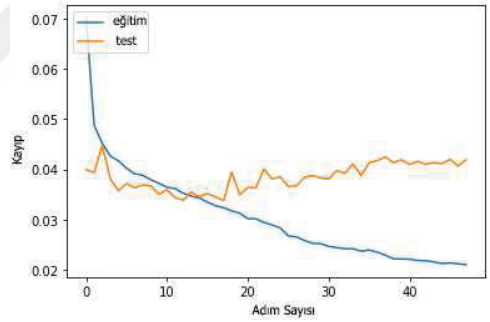
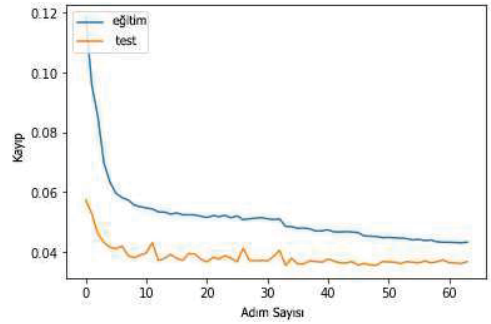
Erken durdurma yöntemi kullanılarak eğitim gerçekleştirildiği için, bütün modeller farklı adım değerlerinde eğitimlerini tamamlamıştır. Bu tez çalışmasında oluşturulan diğer modellerde kullanılan veri setindeki örnek sayısı ve kullanılan katmanlar Çizelge 3.3'de gösterilmiştir.

Çizelge 3.3. DL Tabanlı Oluşturulan Modellerin Özellikleri

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**
Evrişim Katmanı Sayısı	5	5	5	5	5
Bütün Bağlı Katman Sayısı	4	5	5	4	4
Adım Sayısı	100	44	48	64	81
Aktivasyon Fonksiyonu	ReLU	ReLU	eLU	eLU	eLU
Yığın	100	100	100	100	100
Yığın Normalizasyonu	-	Uygulandı	Uygulandı	-	-
Örnek Sayısı	74.484	46.809	71.031	71.031	46.809
Eğitim Süresi	860dk 49sn	301dk 39sn	394dk 34sn	517dk 41sn	549dk 9sn

Kayıp fonksiyonu için, ortalama hata karesi, gerçek direksiyon açısı ile modelin tahmin ettiği direksiyon açısı arasındaki hatayı minimuma indirmek için kullanılmıştır. Ortalama hata karesi, regresyon ağları için uygun bir kayıp fonksiyonu olduğu için seçilmiştir. Ortalama hata karesi, orijinal değerler ile öngörülen değerler arasındaki fark karesinin ortalamasını alır. Ortalama hata karesinin avantajı kolay bir şekilde gradyan hesaplamasıdır.

Çizelge 3.4. Oluşturulan Diğer Modellerin Kayıp Fonksiyonu Grafikleri

Modeller	Kayıp Fonksiyonları
Model 1	
Model 2	
Model 3	
Model 4	

Hata karesinin hesaplanması, daha büyük ve daha küçük hataların daha belirgin hale gelmesine neden olur, bu nedenle model daha büyük hatalara odaklanır. Çizelge 3.4.'de tez çalışmasında oluşturulan diğer modellerin eğitim sonunda oluşan kayıp fonksiyonu grafikleri verilmiştir. Bu tez çalışmasında oluşturulan DL tabanlı modellerin eğitim

sonunda oluşan ortalama kare hata değerleri ve ortalama mutlak hata değerleri Çizelge 3.5'de verilmiştir.

Çizelge 3.5. Oluşturulan Diğer Modellerin Hata Ölçüm Değerleri

	Model 1	**Model 2**	**Model 3**	**Model 4**
Ortalama Hata Karesi	0,0383	0,0374	0,0419	0,0368
Ortalama Mutlak Hata	0,1539	0,1513	0,1614	0,1545

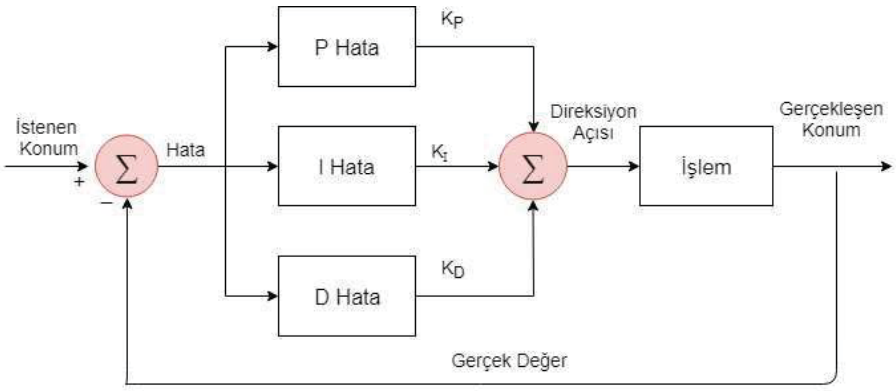
Oluşturulan modellerin örnek sayıları ve eğitim sürüşü sırasında kullanıcının pistteki aracı ne kadar iyi yönlendirdiği modelin başarısını etkilemektedir. Model 2 ve Model 3'te diğer modellerden farklı olarak yığın normalizasyonu (batch normalization) kullanılmıştır. Çizelge 3.2.3'de özellikleri belirtilen modellerden, Model 5 en yüksek başarım oranına sahiptir ve tez çalışmasında bu model ele alınmıştır. Model 5 ile ilgili detaylı bilgiler önceki bölümlerde verildiği için Çizelge 3.2.4'te grafikler gösterilmemiş ve Çizelge 3.2.5'te de değerler verilmemiştir.

DL tabanlı oluşturulan modellerin hepsinde en iyileme yöntemi olarak $Adam$ en iyileme kullanılmıştır. $Adam$ optimizasyonu, CNN eğitimi için en etkili optimizasyon algoritmalarından biridir. Deneysel sonuçlar $Adam$'ın pratikte iyi çalıştığını ve diğer rasgele optimizasyon yöntemleriyle olumlu şekilde karşılaştırdığını göstermektedir.

3.3. PID Tabanlı Sürüş

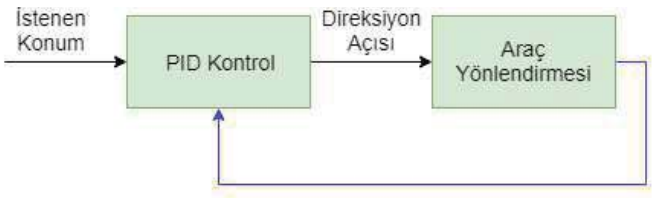
Bu tez çalışmasında, Udacity tarafından tasarlanmış olan bir sürüş simülatörde keskin sola ve sağa dönüşleri olan dairesel pistte özerk sürüş için bir PID kontrol cihazı uygulanmıştır. PID kontrolü robot biliminde yaygın olarak kullanılmaktadır. Bunlar, manuel olarak ayarlanana sistem parametrelerini otomatik ayarlamak için bir yol sağlar. Manuel ayarlama çok uzun ve sıkıcı bir işlemdir ve yine de en uygun parametre setini vermeyebilir. Bu nedenle, PID kontrolörleri bize zaman kazandırmakla kalmaz, aynı zamanda daha fazla permütasyon test edilebildiğinden daha iyi sonuçlar verir. Bir PID kontrol cihazı, istenen değer (set point) ile gerçek değer arasındaki farkı, hata değeri olarak sürekli hesaplar ve orantılı, integral ve türev kontroller ile bir düzeltme

uygulamaya çalışır. PID parametreleri, çapraz yol hatasını en aza indirecek şekilde ayarlamaya çalışır ve sürücüsüz aracın yolun orta kısmında kalması ve yol şeridinin kenarlarına değmeden veya yol şeridinin dışına çıkmadan yumuşak sol ve sağ dönüşler yapmasını sağlar. Sürüş simülatöründe gerçekleştirilen otonom sürüş sırasında, aracın yanal konumu ile şeridin merkezi arasındaki çapraz yol hatası (Cross Track Error, CTE) ölçmüştür.



Şekil 3.20. PID Sistem Mimarisi (Anonim 2019c)

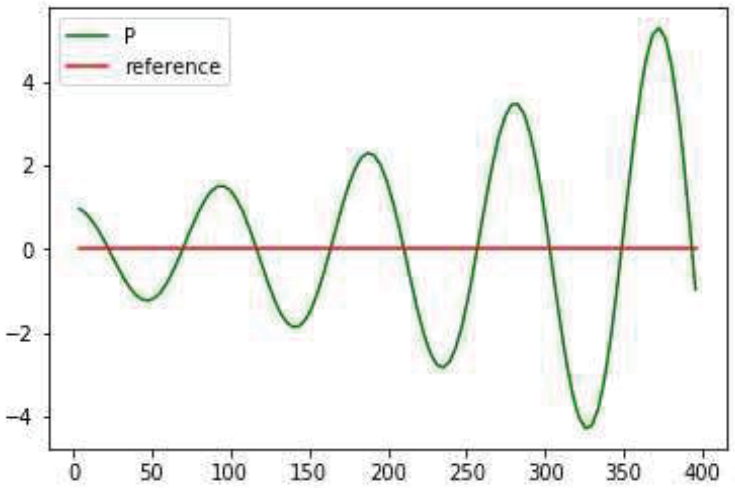
PID kontrolün amacı sistemdeki toplam hatanın en aza indirilmesi veya sistemdeki toplam kazancın maksimize edilmesidir. Simülatörde, CTE değeri simülatör tarafından gönderilen veri mesajından okunur ve PID kontrol hata değerlerini günceller ve toplam hataya dayanarak direksiyon açısını tahmin eder. Şekil 3.20'de PID sistem mimarisi verilmiştir. CTE aracın direksiyon açısını düzenleyen kontrolörün P (proportional), integral (I) ve derivative D bileşenlerini hesaplamak için kullanılmıştır. Direksiyon açısı, PID terimlerine dayanarak istenen değere ulaşmaya çalışmaktadır. Şekil 3.21'de PID kontrol işlem akışı verilmiştir.



Şekil 3.21. PID Kontrol İşlem Akışı

P veya "orantılı" bileşen, sürücüsüz araç davranışı üzerinde doğrudan gözlemlenebilir etkiye sahiptir. Sürücüsüz aracın, şerit merkezinden uzaklığı ile orantılı bir şekilde

yönlendirmesini sağlar. Eğer araç çok fazla sağa doğru giderse, hafifçe sola, tam tersi durumda da aracı hafifçe sağa doğru yönlendirir. Oransal kontrol tek başına kullanıldığında sürekli bir salınım gerçekleşir. Şekil 3.22'de oransal kontrol tek başına kullanıldığında gerçekleşen salınım grafiği verilmiştir. Oransal kontrol, hedef değere ulaşmada yardımcı olur, ancak hedef değer etrafındaki hata salınımı (küçük de olsa) hedef değere ulaşmak için azaltılmalıdır. İntegral kontrolün devreye girdiği yer burasıdır. İntegral kontrolü, zaman içindeki hataların toplamıdır. Böylece hata küçük olduğunda, integral bileşen hedef değere nispeten kısa bir sürede ulaşmaya yardımcı olur çünkü integral zamanın değişim hızının üstündedir.

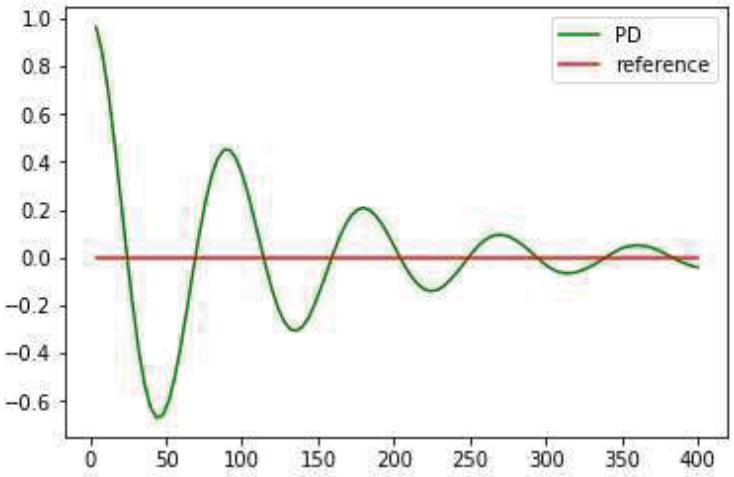


Şekil 3.22. Oransal Kontrol ile Salınım

PI kontrol ile bir değişkenin kontrol edilmesi sağlanabilir, ancak hatayı tahmin etmek ve bunu geribildirim sistemine dahil etmek sabit bir denetleyiciye sahip olmanın çok önemli bileşenidir. Türev kontrolü tam da bunu yapar, önceki hatayı temel alarak gelecekteki hatayı hesaplar ve sisteme dahil eder. Denetleyicinin çıktısında salınım olasılığı varsa, türev kontrolü salınım kontrolüne yardımcı olacaktır. Şekil 3.23'de oransal ve türev kontrol birlikte kullanıldığında gerçekleşen salınım grafiği verilmiştir.

Bütün kontrolörler birlikte kullanıldığında zamana bağlı hata oranı düşecektir ve salınım azalacaktır. Şekil 3.24'de PID kontrolün salınım grafiği verilmiştir. PID, direksiyon açısını hesapladıktan sonra, bir gaz değeri elde edilir ve simülatöre geri gönderilir. Yeni bir mesaj alındığında, güncelleme ve tahmin işlemini tekrar başlatmak için yeni CTE

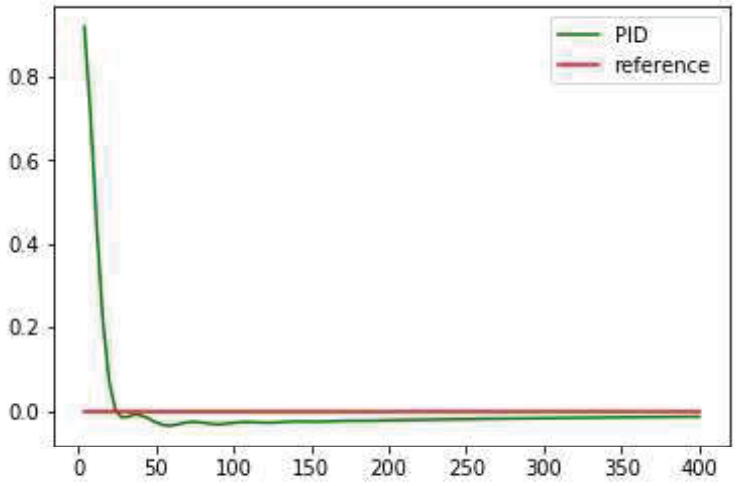
değeri kullanılır. Veri mesajlarının programa gönderilme hızı, simülatörün açılış ekranında seçilen çözünürlük ve grafik kalitesinden oldukça etkilenmektedir. Ayrıca, başka programların CPU / GPU kullanımı, simülatörü çalıştıran bilgisayarın hızı ve işletim sistemi de mesajların gönderilme hızını etkilemektedir. Programa mesajlar geç geldiğinde güncelleme yavaş olmaktadır ve aracın yol üzerindeki salınımı artmaktadır. Bu salınımlar aracın yoldan çıkmasına neden olabilmektedir.



Şekil 3.23. Oransal Kontrol ve Türev Kontrol ile Sapmalar

Salınımları gidermek için PID parametrelerinin çok hassas bir şekilde ayarlanması gerekmektedir. Bu tez çalışmasında PID parametreleri manuel olarak ayarlanmıştır. PID değerleri farklı koşullara göre değişebilmektedir, bu yüzden sürücüsüz aracı dengede tutacak değerlerin seçilmesi önemlidir. Parametre değerlerinden herhangi birinin hatalı olması durumunda, daha fazla salınım ve sürüş pistindeki yoldan sapmalar görülmektedir. PID kontrolü istenen konumda ilerlerken bazen araç salınımlar gerçekleştirmektedir. Sürücüsüz araç istenen değere (yolun ortasına) ulaşmak istemektedir. Kazanç ayarlarının manuel olarak yapılması, PID kontrollerini ayarlamanın en basit yöntemidir. PID kontrolörünü manuel olarak ayarlamak için, önce integral kazancı (K_I) ve türev kazancı (K_D) sıfır yapılmalıdır. Araç salınım gerçekleştirene kadar oransal kazanç değeri (K_P) artırılır. Genel olarak oransal kazanç bu değerin yarısına kadar ayarlanmalıdır. Aracın direksiyon açısının hesaplanmasında Denklem 3.1 kullanılır. K_P ayarlandıktan sonra, uygun bir zaman ölçeğinde salınım düzeltilinceye kadar K_I artırılır. K_I değeri çok fazla artırılırsa, araç istenen değere ulaşmada kararsızlık yaşamaktadır. K_I değeri ayarlandıktan

sonra, K_D değeri artırılır. K_D, salınımı azaltır ve sistemi hızlı bir şekilde yolun orta noktasına götürür. Eğer K_D değeri çok fazla artırılırsa, sürücüsüz araç kontrolü cevap veremeyecek kadar yavaş olur.



Şekil 3.24. PID Kontrol ile Gerçekleşen Salınımlar

$$\propto = -K_P * P_{Hata} - K_I * I_{Hata} - K_D * D_{Hata} \quad (3.1)$$

Kazanç ayarları ile oynayarak, PID kontrolün performansını en üst düzeye çıkararak, sistemdeki değişikliklere hızlı bir şekilde yanıt veren ve istenen konum değerinin salınımını etkili bir şekilde azaltan bir devre ile sonuçlanabilir. PID kontrolörünün kazanç değerlerinin bu şekilde ayarlanması Ziegler-Nichols yöntemi olarak bilinmektedir (Anonim 2019c). 10mph hızında etkili sonuç alabilmek için parametreler hassas bir şekilde ayarlanmıştır (Thorlabs 2019). Gerçekleştirilen tez çalışmasında İlgili yöntem kullanılarak kazanç değerleri hesaplanmıştır ve salınım mümkün olduğunca düşük olmuştur.

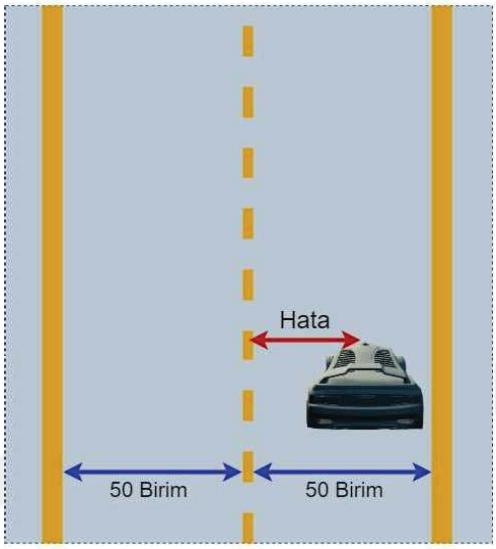
4. BULGULAR

Sürüş simülatöründe oluşturulan DL tabanlı modellerin ve PID tabanlı modelin, sürüş pistindeki şeritlerin tam ortasından ilerlemesi istenmektedir. Araç sürüş pistinde otonom olarak ilerlerken şeritlerin merkezinden sapmalar yaşanmaktadır. Otonom sürüşün test edilebilmesi için araç sürüş pistinde ilerlerken, aracın şeritlerin orta noktasından sol ve sağ şeride olan mesafesinin ölçülmesi gerekmektedir. Aracın her şeritten ne kadar uzakta olduğunu hesaplamanın birkaç yolu vardır. Simülatör ortamında araca yerleştirilen kameralardan yararlanıp ya da sürüş kaydı yapılıp görüntü işleme yöntemleri kullanılabilir. Görüntü işleme yöntemleriyle sürüş pistindeki şeritler bulunup işaretlenebilir ya da şeritleri bulmak için derin öğrenme yöntemleri de kullanılabilir. DL yöntemleri kullanılacaksa, LiDAR gibi diğer sensörlerden gelen konumları kullanabilir ve sinir ağı sensörlerden gelen bilgiler kullanılarak eğitilebilir (Gurghian ve ark. 2016).

Şekil 4.1. Sürücüsüz Araç Pisti

Bu tez çalışmasında otonom sürüş sırasında, aracın hareketi kayıt altına alınmıştır. Bu video kaydı performans değerlendirilmesinde kullanılmıştır. Sürücüsüz araç test edilirken, sürüş pistindeki bir tam turdaki performansı değerlendirilmiştir. Sürücüsüz aracın hızı 10 mph olarak belirlenmiş ve aracın şeridin orta noktasından ne kadar sapma yaptığı ölçülmüştür. Sürüş pistindeki iki şeridin arasında kalan mesafe Şekil 4.1'de gösterildiği gibi 100 birim olarak değerlendirilmiştir. Hata değeri ölçülürken şeritlerin

tam ortası 0 olarak kabul edilmiştir. Şekil 4.2'de kesikli çizgilerle gösterilen çizgi aracın sürekli hareket etmesi istenen konumdur. İstenen notadan sola doğru sapmalar 0 ile -50 arasında ve sağa doğru sapmalar ise 0 ile 50 arasında değerlendirilmiştir.



Şekil 4.2. Sürücüsüz Aracın Orta Çizgiden Sapması

Araç Konum Tespiti ve Test İşlemi: Sürücüsüz aracın testleri Udacity tarafından geliştirilen sürüş simülatöründe gerçekleştirilmiştir. Sürüş simülatöründe bir tam tur için testler gerçekleştirilmiştir. Sürücüsüz aracın sürüş simülatöründe bir tam turu kayıt altına alındı ve aracın konumu ve şeritlerin konumu bulundu. Kayıt altına alınan video görüntüsü görüntü karelerine ayrılarak aracın konumu tespit edilmiştir. Bu işlem için ilk önce aracın farklı konumlardaki görüntüsü alınmıştır. Aracın farklı konumlardaki görüntüsünde parlaklık değerleri farklı olduğundan, tüm parlaklık değerlerinin ortalaması alınmıştır. Her bir görüntü karesine çapraz korelasyon işlemi uygulanarak aracın konumu tespit edilmiştir. Aracın konumu tespit edildikten sonra aracın orta noktası merkez noktası olarak kabul edilmiştir.

Sürüş simülatör pistinde şeritlerin tespiti için, yol harici bölgeler görüntüden kırpılmıştır. Sürüş simülatöründe şeritlerin tespit edilmesi için iki farklı yöntem uygulanmıştır. Sürüş pistinde köprü haricindeki yollarda sarı şeritler vardır, fakat köprüde şeritler bulunmamaktadır. Köprü haricinde olan yollardaki sarı şeritlerin bulunması için, görüntü karelerinde sarı pikseller segmentasyon işlemi uygulanarak tespit edilmiş ve hough

dönüşümü uygulanmıştır. Hough dönüşümüyle şeritler belirlenip, araca en yakın çizgi şerit çizgisi olarak kabul edilmiştir. Sürüş pistinde köprüde sadece hough dönüşümü uygulanmıştır. Çizgilerin sonsuz olduğu ve sıfır olduğu bölgelerde eğim (slope) alınmıştır. Doğru şerit çizgileri bulunduktan sonra Şekil 4.2'de gösterildiği gibi aracın görünge merkezinden sapması hesaplanmış ve yörünge merkezinden sapma grafiğiyle birlikte histogram grafiği oluşturulmuştur.

4.1. Sürüş Simülatöründe Otonom Sürüş

Sürüş simülatöründe modelimizin otonom hareketi için, iki yönlü istemci-sunucu iletişimi kurmak gerekmektedir. CNN tarafından oluşturulan modeli simülasyona bağlamak için Flask-SocketIO uygulamasını bilgisayara kurmak gerekir (Anonim, 2019e). Windows işletim sistemine sahip bilgisayarda sanal bir ortam oluşturup, ilgili uygulamanın 4567 portundan sürüş simülatörüyle haberleşmesinin sağlanması gerekmektedir. Sanal bir ortam oluşturmak ve sürüş simülatörüyle model arasındaki bağlantıyı kurmak için, komut isteminden ilgili Python kütüphanelerini kurmamız gerekmektedir. Bilgisayardan komut isteminde C:/ sürücüsüne girip; $conda\ create\ --name\ myenviron$ komutuyla sanal bir ortam oluşturulur. Bu ortamı aktif etmek için; $activate\ myenviron$ komutu yazılır. Aktif duruma getirilen ortamda aşağıdaki komutlar sırasıyla girilerek; *flask, socketio, eventlet, tensorflow, keras, pillow, numpy ve opencv* kütüphaneleri yüklenir.

Komut isteminde yüklenecek kütüphaneler:

$conda\ install\ -c\ anaconda\ flask$
$conda\ install\ -c\ conda-forge\ python-socketio$
$conda\ install\ -c\ conda-forge\ eventlet$
$conda\ install\ -c\ conda-forge\ tensorflow$
$conda\ install\ -c\ conda-forge\ keras$
$conda\ install\ -c\ anaconda\ pillow$
$conda\ install\ -c\ anaconda\ numpy$
$conda\ install\ -c\ conda-forge\ opencv$

Yukarıda verilen kütüphaneler komut isteminde oluşturulan sanal ortama yüklendikten sonra CNN modeli çalıştırabilir. CNN modelini çalıştırmak için $drive.py$ dosyası aşağıdaki gibi oluşturulmalıdır.

drive.py Dosyası:

```
import socketio
import eventlet
import numpy as np
import cv2
from flask import Flask
from keras.models import load_model
import base64
from io import BytesIO
from PIL import Image

sio = socketio.Server()
app = Flask(__name__)
speed_limit = 30

def img_preprocess(img):
    img = img[60:135, :, :]
    img = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img, (3, 3), 0)
    img = cv2.resize(img, (200, 66))
    img = img / 255
    return img

@sio.on('telemetry')
def telemetry(sid, data):
    speed = float(data['speed'])
    image = Image.open(BytesIO(base64.b64decode(data['image'])))
    image = np.asarray(image)
```

```
    image = img_preprocess(image)
    image = np.array([image])
    steering_angle = float(model.predict(image))
    throttle = 1.0 - speed/speed_limit
    print ('Direksiyon açısı = {} Gaz = {} Hız = {}'.format(steering_angle, throttle, speed))
    send_control(steering_angle, throttle)

@sio.on('connect')
def connect(sid, environ):
    print('Bağlandı')
    send_control(0, 0)

def send_control(steering_angle, throttle):
    sio.emit('steer', data = {
            'steering_angle': steering_angle.__str__(),
            'throttle': throttle.__str__()
    })

if __name__ == '__main__':
    model = load_model('model.h5')
    app = socketio.Middleware(sio, app)
    eventlet.wsgi.server(eventlet.listen(('', 4567)), app)
```

Sürüş simülatöründe aracın otonom hareketi için $drive.py$ dosyasının bulunduğu klasör belirlenip python $drive.py$ komutu verilerek çalıştırılmalıdır. PID tabanlı otonom sürüş yapılacaksa *cmake*, *make*, *MinGW* ve *uWebSockets* paketlerinin bilgisayara kurulması gerekmektedir. Paket kurulumları gerçekleştirildikten sonra PID sürüş simülatörü seçilip çalıştırılmalıdır (Anonim, 2019f). Derin öğrenme tabanlı oluşturulan model ve PID kontrolüyle oluşturulan sistem aynı pistlerde otonom sürüş gerçekleştirmektedir. Aynı sürüş pistlerinin bulunduğu Udacity tarafından hazırlanmış olan, PID kontrolü için farklı ve derin öğrenme model için farklı sürüş simülatörleri bulunmaktadır. Bu sürüş simülatörleri aynı portlardan haberleşmektedirler.

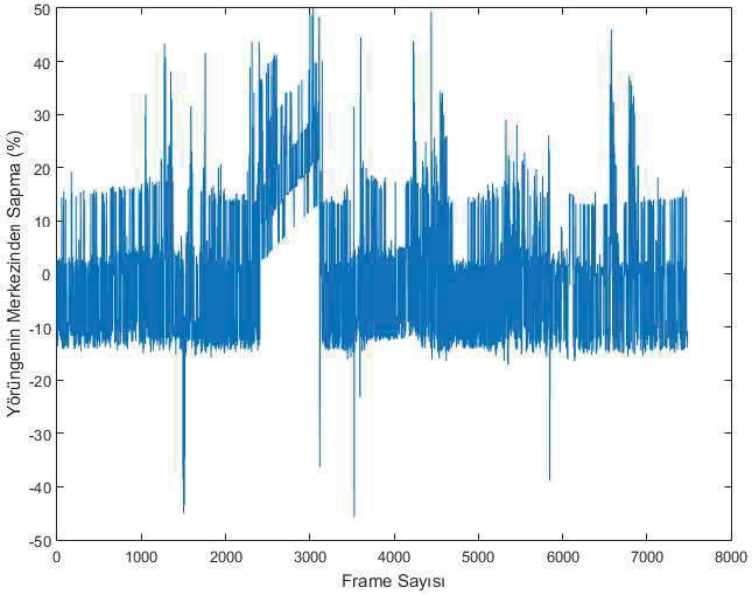
4.2. PID Kontrol Testleri

PID kontrol testleri sürüş simülatöründe gerçekleştirilmiştir. Otonom sürüş sırasında aracın hızı 10mph'ye sabitlenmiş ve sürüş pistinde aracın bir tam turda performans değerlendirilmesi yapılmıştır. Otomobilin nasıl tepki vereceğini ve farklı parametre değerlerinin sürüş performansını nasıl etkilediği görmek için PID parametreleri empirik olarak ayarlanmıştır.

Çizelge 4.1. PID Kazanç Değerleri

	Kazanç Değeri
K_P	0,75
K_I	0,0001
K_D	1

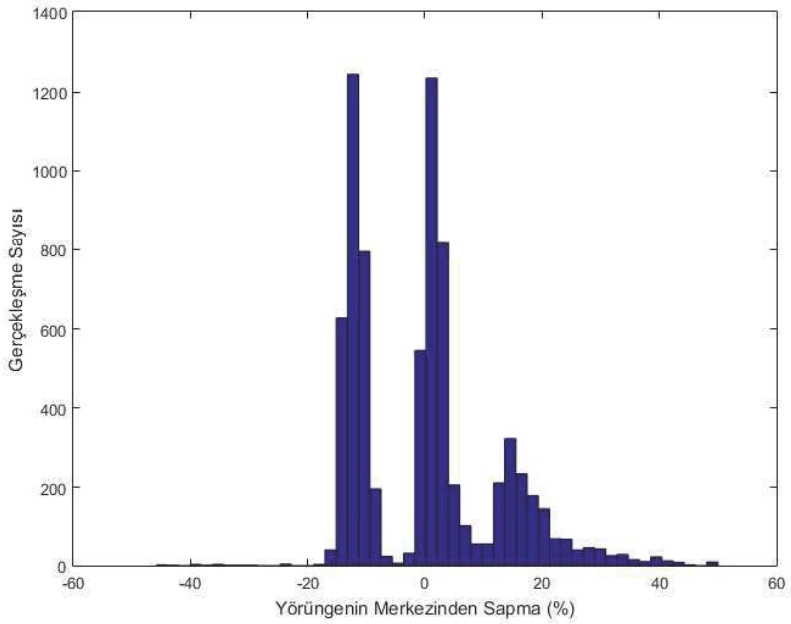
Oransal katsayısı aracın direksiyon girişi üzerinde doğrudan etkiye sahip olduğundan, ilk olarak K_P değeri ayarlanmıştır. K_P değeri ayarlandıktan sonra araç yine gitmesi istenen yolun ortasından sapmalara yapmıştır. K_P değeri çok yüksek olduğunda sürücüsüz aracın keskin dönüşler ve K_P değeri düşük olduğunda aracın yumuşak dönüşler yaptığı gözlenmiştir. Çok fazla salınım yapmaması için hız düşürülmüştür.



Şekil 4.3. Yörünge Merkezinden Sapma - PID

Sürücüsüz aracın sol şeritten sağ şeride, sağ şeritten sol şeride doğru tekrarlayan salınımını azaltmak için K_D değeri ayarlanmıştır. K_D değeri düşük olduğunda K_P değeri tarafından üretilen sapmayı engelleyememiştir ve yüksek olduğunda direksiyon çok seğirmeye başlamıştır. Daha sonra K_I değeri zamansal değişimi göstermesi için ayarlanmıştır. Manuel olarak ayarlanan PID kontrolör kazanç değerleri Çizelge 4.1'de gösterilmiştir.

Sürüş pistinin bir tam turunu PID tabanlı sürücüsüz araç 10mph hızla tamamlamıştır. Aracın bir tam turu kayıt altına alınmıştır (Aki 2019b). Araç hareket halindeyken kaydedilen videodaki görüntülerden, yoldaki şeritler tespit edilmiştir. İki şeridin tam ortası yörünge merkezi olarak kabul edilmiştir. Sürücüsüz aracın yoldaki konumu tespit edildikten sonra, her görüntü karesinde sürücüsüz aracın merkeze olan uzaklığı ölçülmüştür. Bu ölçüm hata olarak kabul edilmiştir. Sürücüsüz aracın bir tam turdaki salınımı grafiksel olarak Şekil 4.3'te gösterilmiştir. Sürücüsüz aracın yörünge merkezinden sola doğru sapması negatif olarak ve sağa sapması pozitif olarak değerlendirilmektedir.



Şekil 4.4. Yörünge Merkezinden Sapma Histogramı - PID

Kaydedilen video üzerinde hata ölçümü gerçekleştirildikten sonra, yörünge merkezinden sapma grafiği üzerinde normalizasyon işlemi gerçekleştirilmiş ve bu değerlerin histogram grafiği Şekil 4.4'deki gibi oluşturulmuştur.

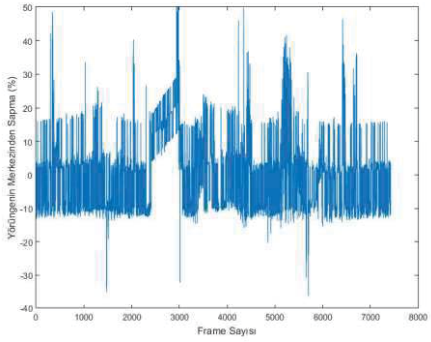
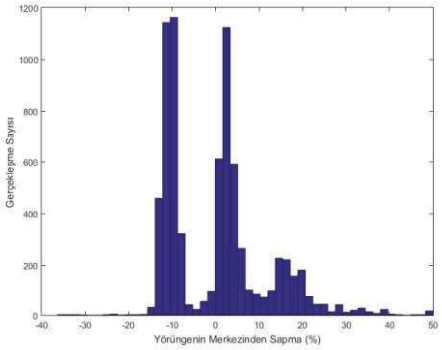
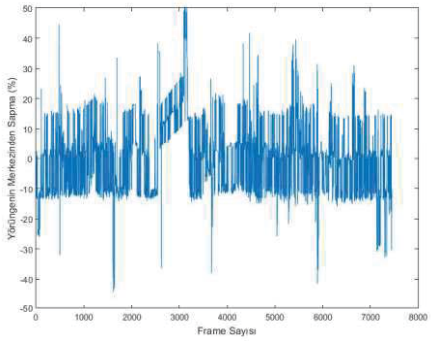
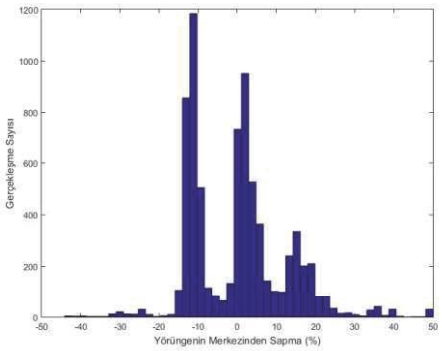
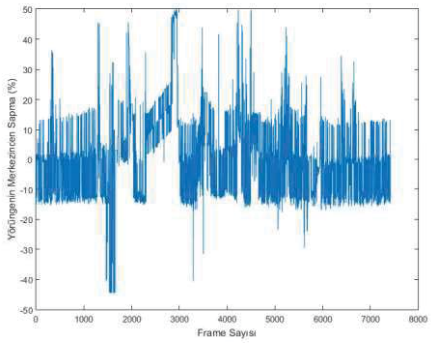
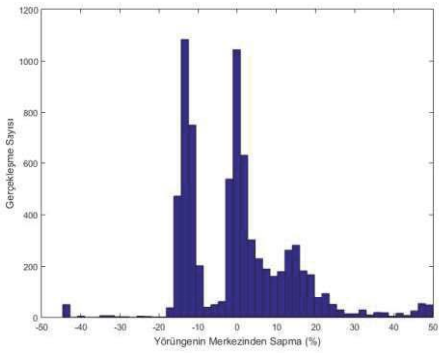
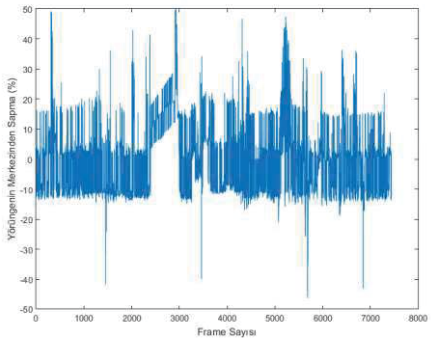
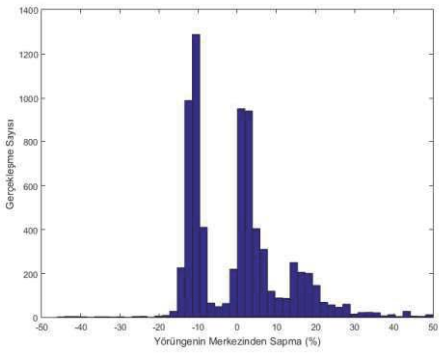
4.3. Derin Öğrenme Testleri

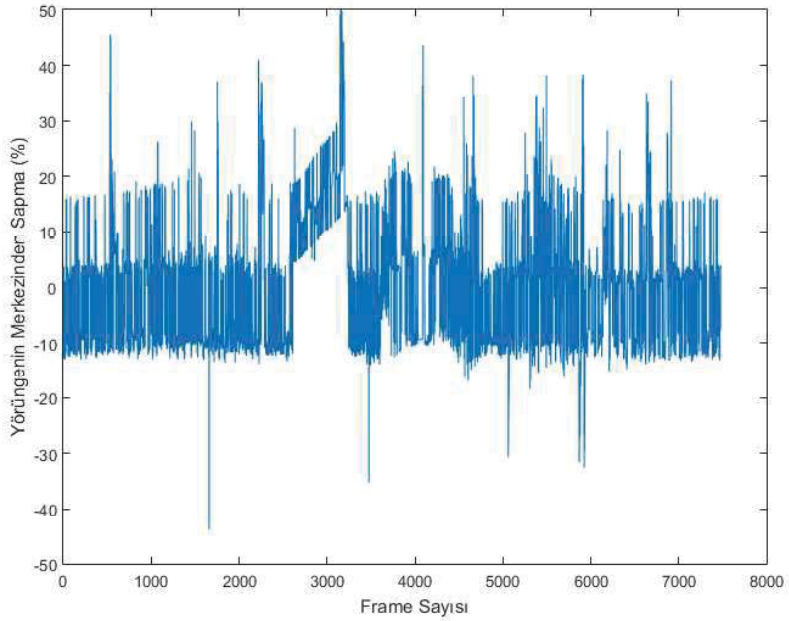
Modeller, Adam en iyileme yöntemi kullanılarak eğitilmiştir ve öğrenme oranı 0,001 olarak ayarlanmıştır. CNN modelleri farklı yığın değerleriyle eğitildiğinde örnek sayısının az olmasından kaynaklı başarım değerleri arasında büyük farklar oluşmuştur. En başarılı sonuçlarda yığın değeri 100 olarak ayarlanmıştır. DL tabanlı yöntemlerin performans değerlendirmesinin objektif olarak bir yapılması için tüm yöntemler birbirleriyle karşılaştırılmıştır. Başarılı otonom sürüşün testleri, sürüş simülatöründeki pistte gerçekleştirilmiştir. Özerk sürüş modu sırasında, araca monte edilmiş merkezi kameradan gelen sinyal sürekli olarak alınmış ve eğitilen modele girdi olarak gönderilmiştir. Model kendisine gelen sinyallere karşılık direksiyon açısı tahmin etmiş ve aracın yörüngede kalmasını sağlamıştır.

Oluşturulan modellerin hepsinin sürüş pistinde bir tam turu kaydedilmiştir. Başarım oranı en yüksek olan DL tabanlı modelin yörüngeden sapma grafiği ve gerçekleşen sapmanın histogram grafiği, diğer DL tabanlı modellerden ayrı değerlendirilmiştir. Sürüş pistindeki bir tam tur için diğer modellerin yörünge merkezinden sapma ve histogram grafikleri Çizelge 4.2'de verilmiştir.

81 adımda eğitimini tamamlayan DL tabanlı modelin sürüş pistinde, otonom sürüş sırasında 10 mph hızla bir tam turu kayıt altına alınmıştır (Aki 2019a). Sürücüsüz araç yol boyunca yörünge merkezinde kalmayı başarmıştır. Eğitim sürüşü sırasında toplanan verirlerden kaynaklı bazen yörünge merkezinden sapmalar gerçekleşmiştir. Kaydedilen görüntüye ait yörüngeden sapma grafiği Şekil 4.5'te verilmiştir. İki şerit arasında kalan nokta yörünge merkezi olarak kabul edilmiştir. Sürücüsüz araç otonom sürüş sırasında, yörünge merkezinden sola ve sağa doğru sapmalar gerçekleştirmiştir. Sağa ve sola doğru yörüngeden sapma değerleri normalize edilmiş ve sola sapma negatif (-), sağa sapma pozitif (+) olarak gösterilmiştir.

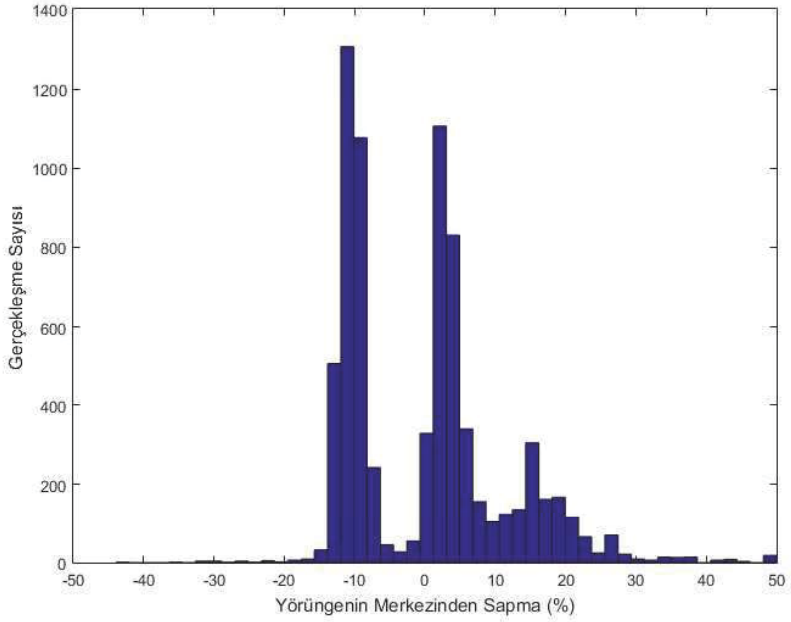
Çizelge 4.2. Oluşturulan Diğer Modellerin Yörünge Merkezinden Sapması

Model	Yörünge Merkezinden Sapma	Histogram
Model 1		
Model 2		
Model 3		
Model 4		



Şekil 4.5. Yörünge Merkezinden Sapma **-** Model 5

Sürücüsüz aracın sürüş pistindeki özerk sürüşünün istatistiksel analizi de histogram aracılığıyla sunulmaktadır. Bu analiz uzun vadeli testler için önemlidir. Salınımları incelemek için, sürüş pistinde sürücüsüz aracın bir tam tur başına yörüngenin ortasından bağıl sapma histogramı Şekil 4.6'da sunulmuştur.



Şekil 4.6. Yörünge Merkezinden Sapma Histogramı **-** Model 5

DL tabanlı tüm modeller iyi bir performans sergilemiştir ve yörüngenin merkezinden önemli bir sapma olmadan özerk sürüş turunu başarıyla tamamlamıştır. DL tabanlı modellerin yörünge merkezinden sapmalarına ait ortalama mutlak hata, standart sapma ve ortalama değerleri hesaplanmıştır. Bu değerler Çizelge 4.3'de gösterilmektedir.

Çizelge 4.3. DL Tabanlı Modellerin Performans Değerlerinin Karşılaştırılması

Model	Ortalama Mutlak Hata (%)	Standart Sapma (%)	Ortalama
Model 1	9,19753901	11,43511505	5,05e-14
Model 2	9,64676542	12,29487581	-1,12e-13
Model 3	10,2401501	13,7285633	-2,05e-15
Model 4	9,48165634	11,98705726	-1,15e-13
Model 5	9,19071511	11,28509483	6,25e-14

DL tabanlı modeller arasından en yüksek başarıma sahip olan model sürüş pistinde, yörünge merkezinden çok az sapma gerçekleştirerek özerk sürüşü gerçekleştirmiştir. Genel pist üzerindeki virajlarda sapmalar gerçekleşmiştir.

4.4. PID Tabanlı ve Derin Öğrenme Tabanlı Sürüş Testlerinin Karşılaştırılması

Sürüş simülatörü kullanılarak sürücüsüz araç testleri yapılırken, orta kamera görüntüleri kullanılmaktadır. Orta kameradan alınan görüntülerle aracın direksiyon açısı kontrol edilmeye çalışılmaktadır. DL tabanlı ve PID tabanlı sürücüsüz aracın otonom sürüşü için, bir tam turda yörünge merkezinden sapma değerlerinin hesaplaması yapılmıştır. PID tabanlı sürücüsüz aracın virajlarda çok fazla salınım yaptığı gözlenmiştir. Özellikle de köprüden sonraki keskin virajda çok fazla salınım gerçekleştirmiştir. Bazı virajlarda sürücüsüz aracı yörünge merkezinde tutmak için küçük salınımlar gerçekleştirmiştir. PID tabanlı sürücüsüz araç, yörünge merkezinde kalabilmek için çok fazla salınım gerçekleştirdiğinden gerçek hayat için uygun olmadığı görülmüştür. DL tabanlı sürücüsüz araç daha stabil bir sürüş sergilemiştir. DL modeli yolun farklı desen ve farklı parlaklık değerlerini öğrenmiştir. Köprüden sonraki ilk keskin dönüşte sol şeride doğru bir sapma gerçekleştirmesine rağmen, sonrasında yörünge merkezinde gitmeyi başarmıştır. Burada belirtilen keskin viraj hariç yolun geri kalan kısmında DL tabanlı sürücüsüz araç daha iyi

performans göstermiştir. PID tabanlı sürücüsüz araç performansı ile DL tabanlı sürücüsüz araç performansı Çizelge 4.4'de karşılaştırılmıştır.

Çizelge 4.4. Performans Değerlerinin Karşılaştırılması

Model	Ortalama Mutlak Hata (%)	Standart Sapma (%)	Ortalama
DL Tabanlı Sürüş	9,191	11,285	6,25e-14
PID Tabanlı Sürüş	9,520	12,217	3,09e-14

Sürüş pistindeki bir tam turda yörünge merkezinden sapma değerleri ölçüldükten sonra, anlaşılabilirliği artırmak için normalizasyon uygulanmıştır. Normalizasyon uygulandıktan sonra DL tabanlı sürücüsüz aracın ortalama mutlak hata değeri 9,19 ve PID tabanlı sürücüsüz aracın ortalama mutlak hata değeri 9,52 olarak ölçülmüştür. DL tabanlı sürücüsüz aracın sürüş pistinde bir tam turda standart sapması 11,28 olarak ölçülürken, PID tabanlı sürücüsüz aracın standart sapma değeri 12,21 olarak ölçülmüştür. Her iki sisteminde saniyedeki görüntü karesi sayısı 30fps olarak ölçülmüştür. Genel olarak performans değerleri karşılaştırıldığında DL tabanlı sistemin daha başarılı sonuçlar sergilediği görülmüştür.

5. TARTIŞMA ve SONUÇ

Bu tez çalışmasında araştırılan ana araştırma sorusu şuydu: Sadece kamera kullanılarak sürüş pistindeki şeritlerin ortasında otonom bir aracı tutmak için, DL Tabanlı bir model ve PID kontrol tabanlı bir sistem nasıl geliştirilebilir? Gerçekleştirilen eğitim ve testlerde bu sorunun cevabı alınmıştır. DL tabanlı farklı model kullanılarak eğitim gerçekleştirilmiş ve başarılı sonuç üreten modellerin sürüş simülatöründe gerçek zamanlı otonom olarak testleri gerçekleştirilmiştir. Aynı zamanda PID kontrol parametreleri de düzgün bir şekilde ayarlanarak sürücüsüz aracın sürüş pistinde otonom hareketi sağlanmıştır. Testler sonunda PID kontrol ile de başarılı sonuçlar elde edilse de, en başarılı test sonucu DL tabanlı model ile gerçekleşmiştir. PID kontrol ile otonom sürüş sırasında bazen salınımlar gerçekleşmiştir. DL tabanlı sistemde fazla salınım gerçekleşmemiştir.

DL modellerinde Bölüm 3.2'de açıklandığı gibi, aktivasyon fonksiyonu olarak elu ve en iyileme yöntemi olarak $Adam$ iyileştirici kullanılarak daha önce toplanan veri kümesi eğitilerek, ortalama hata karesi ve ortalama mutlak hata değerleri hesaplanmıştır. Hesaplanan değerlerin grafikleri oluşturulmuştur. İlk aşamada gerçek zamanlı çıkarım sırasındaki sonuçlar, beklendiği gibi iyi gerçekleşmemiş ve sürücüsüz araç sürüş pistinde kalmayı başaramamıştır. Modelin farklı platformlarda da eğitimi sağlanmıştır ve eğitim yapılan makinenin özelliklerinin düşük olması model başarısını etkilemiştir. Aynı zamanda bilgisayar özellikleri düşük olduğunda, gerçek zamanlı olarak sürüş testleri sırasında sistemden cevaplarda gecikmeler yaşandığı için aracın yolda kalmasında zorluklar ortaya çıkmıştır. DL modelinin üretmiş olduğu en iyi sonuçlar değerlendirilerek, gerçek zamanlı sürüş simülatöründe sürücüsüz araç iki şeridin tam ortası olan, yörünge merkezinden gitmeyi öğrenmiştir. Aracın sürüş parkurundaki bir tam turda gerçek zamanlı olarak özerk hareketi kaydedilmiştir ve başarım değerlendirmesi Bölüm 4.2'de yapılmıştır (Aki 2019a). Gerçekleştirilen test sürüşü oluşturulan modelin uygun olduğunu göstermiştir, ancak modelin, daha başarılı bir özerk sürüşü için daha fazla özelliğe ihtiyaç duyduğu görülmüştür. Daha farklı ve daha fazla örnek verilerek CNN eğitimi gerçekleştirilirse modelin başarısı daha yüksek olacaktır. Farklı yol koşulları ve farklı hava şartlarında eğitim sürüşleri gerçekleştirilip, daha başarılı sonuçlar elde edilebilir.

Deneysel sonuçlardan da görülebileceği gibi, kullanılan model araca monte edilmiş üç kameradan gelen görüntülerle eğitilmiştir. Gerçekleştirilen sistemde daha fazla teknolojiye ihtiyaç duyulsa da CNN'deki düğüm sayısı, ağırlıkların sayısı ve eğitilebilir parametreler oldukça yüksektir. Bunun nedeni, çıkış katmanında çok sayıda düğüm üreten evrişim katmanı için, tüm giriş görüntüsünün kullanılmasından kaynaklanmaktadır. Aynı zamanda görüntü kareleri CNN modeline girmeden önce, görüntülerin parlaklık değerleri, direksiyon açıları ve yönleri değiştirilerek veri artırma işlemleri uygulanmıştır. Veri artırma işlemi modelin aşırı uyumunu önlemiş ve başarım oranının yüksek olmasını sağlamıştır. Derin öğrenme tabanlı sürücüsüz araç sistemleri için bu tez çalışmasından bazı sonuçlar çıkarılmıştır.

- Sürücüsüz aracın eğitim sürüşü sırasında, farklı yol koşulları ve farklı hava şartlarında veri toplanması eğitimin başarısını yüksek tutacaktır.
- Yolun eğimi tek bir yöne doğru olduğunda model o yönde öğrenme gerçekleştireceği için, elde edilen verilerde belli bir eşik değerine göre kırpma gerçekleştirilmiştir. Bu tarz problemlerle karşılaşmamak için eğitim sürüşünün gerçekleştirileceği yolun eğiminin dengeli olması gerekmektedir.
- Eğitim verileri toplandıktan sonra eğitim için gerekli olmayan görsellerin kırpma yöntemiyle temizlenmesi modelin başarısını artıracaktır.
- Modele verilecek girişlerin her biri için aynı değer aralığına sahip olması için toplanan verilerde normalizasyon yapılmalıdır. Normalizasyon işlemi eksik öğrenme ve aşırı öğrenmeyi engelleyecektir.
- Modelin önyargısı ve aşırı öğrenmeyi engellemek için seyreltme işlemi kullanılmalıdır.
- Evrişim katmanları özellik çıkarımlarından sorumlu olduğundan daha fazla evrişim katmanı kullanılabilir.

Bu tez çalışmasında sürüş simülatöründe gerçekleşmiştir. DL tabanlı bir sinir ağı ve bol miktarda veri büyütme teknikleri kullanarak, sürücüsüz bir araç için direksiyon açısını güvenilir bir şekilde tahmin eden modelin gerçekleştirileceği gösterilmiştir. Veri sayısı az olduğundan, modelin eğitim başarısı yükseltmek için veri artırma işlemleri

uygulanmıştır. 10mph hızında başarılı sonuçlar elde edilmesine rağmen gelecekte aşağıda verilen alanlarda çalışmalar gerçekleştirilecektir.

- Sürücüsüz aracın hızı ve gaz kelebeği açısı sinir ağı ile kontrol edilecektir.
- Sürücüsüz aracın hızının 10mph'den daha hızlı olması sağlanacaktır.
- Transfer öğrenme (Transfer Learning) yoluyla VGG, ResNets ve Inception'ı temel alan modellerle deneme yapılacaktır.
- Pekiştirmeli öğrenme ve tekrarlayan sinir ağlarını (Recurrent Neural Networks) kullanılarak sinir ağı eğitimi gerçekleştirilecektir.

Görülebileceği gibi, bu tez çalışmasını daha da ilerletmek ve daha da ikna edici sonuçlar elde etmek için araştırma yapılacak birçok alan vardır.

KAYNAKLAR

Aki., K. 2019a. Sürüş Simülatöründe Derin Öğrenme Tabanlı Sürücüsüz Araç Testi https://www.youtube.com/watch?v=01kLVx6xMQQ-(Erişim tarihi: 07.08.2019).

Aki., K. 2019b. Sürüş Simülatöründe PID Tabanlı Sürücüsüz Araç Testi, https://www.youtube.com/watch?v=QtRFsFQRv3g-(Erişim tarihi: 01.09.2019).

Aldana, K. 2013. U.S. Department of Transportation Releases Policy on Automated Vehicle Development. http://www.nhtsa.gov.edgesuite-staging.net/About+NHTSA /Press+Releases/U.S.+Department+of+Transportation+Releases+Policy+on+Automated +Vehicle+Development-(Erişim Tarihi: 05.08.2019).

Alzantot, M., Chakraborty, S., Srivastava, M. B. 2017. Sensegen: A Deep Learning Architecture for Synthetic Sensor Data Generation. https://arxiv.org/pdf/1701.08886.pdf-(Erişim tarihi: 05.08.2019).

Amidi, A., Amidi, S. 2018. Supervised Learning Cheatsheet, https://stanford.edu/~ shervine/teaching/cs-229/cheatsheet-supervised-learning-(Erişim tarihi: 06.08.2019).

Anonim, 2013. National Highway Traffic Safety Administration. Preliminary Statement of Policy Concerning Automated Vehicles. https://www.nhtsa.gov/staticfiles/rulemaking /pdf/Automated_Vehicles_Policy.pdf-(Erişim tarihi: 05.08.2019).

Anonim, 2014. National Economic Council and President's Council of Economic Advisers. An economic analysis of transportation infrastructure investment. https://obamawhitehouse. archives.gov/sites/default/files/docs/economic_analysis_of_transportation_investments. pdf-(Erişim tarihi: 05.08.2019).

Anonim, 2016a. Waymo, Google Self-Driving Car Project, https://www.google.com/ selfdrivingcar-(Erişim tarihi: 06.08.2019)

Anonim, 2016b. Tesla Motors: Model S Press Kit, https://www.teslamotors.com/ presskit/autopilot.-(Erişim tarihi: 06.08.2019)

Anonim, 2016c. Crashes avoided front crash prevention slashes police-reported rear-end crashes. *Insurance Institute for Highway Safety and Highway Loss Data Institute*, 51(1). https://www.iihs.org/api/datastoredocument/status-report/pdf/51/1-(Erişim tarihi: 05.08.2019)

Anonim, 2019a. U.S. Department of Transportation. Automation white paper. http://www. its.dot.gov/research_areas/pdf/WhitePaper_automation.pdf-(Erişim tarihi: 05.08.2019)

Anonim, 2019b. The 6 Levels of Vehicle Autonomy https://www.synopsys.com/ automotive /autonomous-driving-levels.html-(Erişim tarihi: 05.08.2019).

Anonim, 2019c. Ziegler–Nichols Method, https://en.wikipedia.org/wiki/Ziegler%E2 %80%93Nichols_method-(Erişim tarihi: 11.08.2018).

Anonim, 2019d. PID Kontrol, https://en.wikipedia.org/wiki/PID_controller-(Erişim tarihi: 01.09.2018).

Anonim, 2019e. İki Yönlü İstemci-Sunucu Bağlantısı. https://flask-socketio.readthedocs. io/en/latest/-(Erişim tarihi: 01.09.2018).

Anonim, 2019f. PID Sürüş Simülatörü. https://github.com/udacity/self-driving-car-sim/releases-(Erişim tarihi: 01.09.2018).

Blanco, M., Atwood, J., Vasquez, H. M., Trimble, T. E., Fitchett, V. L., Radlbeck, J., Morgan, J. F. 2015. Human Factors Evaluation of Level 2 and Level 3 Automated Driving Concepts. https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/812182_humanfactor seval-l2l3-automdrivingconcepts.pdf-(Erişim tarihi: 05.08.2019).

Broggi, A., Cerri, P., Felisa, M., Laghi, M. C., Mazzei, L., Porta, P. P. 2012. The Vislab Intercontinental Autonomous Challenge: An Extensive Test for a Platoon of Intelligent Vehicles. *International Journal of vehicle Autonomous System*, 10: 147-164.

Broogi, A., Cerri, P., Debattisti, S., Laghi, M. C., Medici, P., Molinari, D., Panciroli, M., Prioletti, A. 2015. Proud-Public Road Urban Driverless-Car Test. *Intelligent Transportation Systems*, 16(6): 3508-3519.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L., Monfort, M., Muller, U., Zhang, J. 2016. End to End Learning for Self-Driving Cars. https:// arxiv.org/pdf/1604.07316.pdf-(Erişim tarihi: 06.08.2019).

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., Muller, U. 2017. Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. https://arxiv.org/pdf/1704.07911.pdf-(Erişim tarihi: 06.08.2019).

Buehler, M., Iagnemma, K., Singh, S. 2009. The DARPA Urban Challenge: Autonomous Vehicles in City Traffic: Springer, Ed.: Buehler, M., Singh, S., Iagnemma, K., NY, USA, pp:125-162.

Cerri, P., Soprani, G., Zani, P., Choi, J., Lee, J., Kim, D., Yi, K., Broggi, A. 2011. Computer Vision at the Hyundai Autonomous Challenge. International IEEE Conference on Intelligent Transportation Systems (ITSC), 5-7 Oct. 2011, Washington, DC, USA.

Chandni, C.K., Sajith Variyar, V.V., Guruvayurappan, K. 2017. Vision Based Closed Loop PID Controller Design and Implementation for Autonomous Car. International Conference on Advances in Computing, Communications and Informatics (ICACCI). 13-16 September 2017, Udupi, India.

Chavez-Garcia, R.O., Aycard, O. 2016. Multiple Sensor Fusion and Classification for Moving Object Detection and Tracking. *IEEE Transactions on Intelligent Transportation Systems Journal*, 17(2): 525–534.

Chen, C. 2016a. Extracting Cognition out of Images for the Purpose of Autonomous Driving. *PhD thesis*, Department of Operations Research and Financial Engineering, Princeton University, ABD.

Cho, H., Seo, Y., Vijaya Kumar, B.V.K., Rajkumar, R.R. 2014. A Multi-Sensor Fusion System for Moving Object Detection and Tracking in Urban Driving Environments. Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), 31 May–7 June 2014, Hong Kong, China.

Chollet, F. 2018. Deep Learning with Python, Ed: Arritola, T., Gaines, J., Taylor, T., NY, USA, pp: 4-23.

Chowdhary, P., Gupta, V., Gupta, D., Jadhav, A., Mishra, V. 2018. Design of Two Wheel Self Balancing Robot Using PID Controller. *International Journal of Engineering Research & Technology*, 5(1): 1-3

Copot, D., Ghita, M., Ionescu, C. M. 2019. Simple Alternatives to PID-Type Control for Processes with Variable Time-Delay. *Processes*, 7(3): 1-16.

Çavdar, A., Uçar, M., Kılıçaslan, İ. 2008. Trafik Kazalarına Sebep Olan Yüksek Hız Kusurlarının Denetimi ve Aktif Güvenlik Sistemler ile Kontrolü. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*. 23(1): 187-198.

Dickmanns, E.D., Zapp, A. 1987. Autonomous High Speed Road Vehicle Guidance by Computer Vision. *IFAC Proceedings Volumes*, 20(5): 221–226.

Emirler, M. T., Uygan, İ. M. C., Güvenç, B. A., Güvenç, L. 2014. Robust PID Steering Control in Parameter Space for Highly Automated Driving. *International Journal of Vehicular Technology*, 2014(3): 1-8.

Eygü, H. 2018. Trafik Kazalarını Etkileyen Faktörlerin Yapısal Eşitlik Modeli İle İncelenmesi. *Electronic Journal of Social Sciences*, 17(66): 837-850.
Gurghian, A., Koduri, T., Bailur, S. V., Carey, K. J., Murali, V. N. 2016. DeepLanes: End-To-End Lane Position Estimation Using Deep Neural Networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 26 June-1 July 2016, Las Vegas, NV, USA.
Habib, K. 2017. Odi Resume the Automatic Emergency Braking. Technical report. https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.pdf-(Erişim tarihi: 05.08.2019).
Han, G., Fu, W., Wang, W., Wu, Z. 2017. The Lateral Tracking Control for the Intelligent Vehicle Based on Adaptive PID Neural Network*, Journal of Sensors*, 17(6): 1244.
He, K., Zhang, X., Ren, S., Sun, J. 2016. Deep Residual Learning For Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 27-30 June 2016, Las Vegas, NV, USA.
Hinton, G. E., Salakhutdinov, R. R., 2006. Reducing the Dimensionality of Data with Neural Networks, *Computer Science. New Life for Neural Networks*, 313: 504-507.
Jiao, J., Chen, W. W., Shao-Wen, L.I., Wang, J. X. 2008. Self-adaptive PID Control for Intelligent Vehicle Steering System Based on IPSO. *Journal of Anhui University*, 29:1325–1327.
Jigang, H., Jie, W., Hui, F. 2017. An Anti-Windup Self-Tuning Fuzzy PID Controller for Speed Control of Brushless DC Motor. *Journal for Control, Measurement, Electronics, Computing and Communications*, 58(3): 321-335.
Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Vasudevan, R. 2017. Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks? http://arxiv.org/pdf/1610.01983.pdf-(Erişim tarihi: 05.08.2019).
Kanade, T., Thorpe, C., Whittaker, W. 1986. Autonomous land vehicle project at CMU. Proceedings of the 1986 ACM Fourteenth Annual Conference on Computer Science; 4–6 February 1986, OH, USA.
Khare, Y. B., Yaduvir, S. 2010. PID Control of Heat Exchanger System. *International Journal of Computer Applications*, 8(6): 22-27.
Kızrak, M. A. 2019a. Derin Öğrenme İçin Aktivasyon Fonksiyonlarının Karşılaştırılması, https://medium.com/@ayyucekizrak/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4 % B1 r%C4%B1lmas%C4%B1-cee17fd1d9cd-(Erişim tarihi: 06.08.2019).
Kızrak, M. 2019b. https://medium.com/@ayyucekizrak/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-(Erişim tarihi: 06.08.2019).
Kocić, J., Jovičić, N., Drndarević, V. 2019. An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms. *Sensors Journal*, 19(9): 2064.
Krasheninnikov, D. 2017. Autonomous Control of a RC Car With a Convolutional Neural Network, Bachelor's Thesis, South Eastern Finland University of Applied Sciences, Information Technology, Kouvola, Finland.
Krizhevsky, A., Sutskever, I., Hinton, G.E. 2012. Imagenet Classification With Deep Convolutional Neural Networks. *Neural Information Processing Systems Foundation*, 25: 1097–1105.

LeChun, Y., Bottou, L., Bengio, Y., Haffner, P. 1998. Gradient Based Learning Applied to Document Recognition, *Proceedings of the IEEE*, 86(11): 2278 – 2324.
Lie, G., Zejian, R., Pingshu, Ge., Jing, C. 2014. Advanced Emergency Braking Controller Design for Pedestrian Protection Oriented Automotive Collision Avoidance System. *The Scientific World Journal*, 2014: 1-11.
Locraft, G., Wood, A., Weiss, K. 2013. Autonomous cars: Self-driving the new auto industry paradigm. *Morgan Stanley Blue Paper*, https://orfe.princeton.edu/~alaink/SmartDrivingCars /PDFs/nov2013morgan-stanley-blue-paper-autonomous-cars%ef%bc%9a-self-drıvıng-the-ne w-auto-ındustry-paradıgm.pdf-(Erişim tarihi: 05.08.2019).
Montemerlo, M., Thrun, S., Dahlkamp ,H., Stavens, D., Strohband, S. 2006. Winning the DARPA Grand Challenge with an AI Robot. Proceedings of the 21st National Conference on Artificial Intelligence, 16–20 July 2006, MA, USA.
Nie, L., Guan, J., Lu, C., Zheng, H., Yin, Z. 2018. Longitudinal Speed Control of Autonomous Vehicle Based on a Self-Adaptive PID of Radial Basis Function Neural Network. *IET Intelligent Transport Systems*, 12(6): 485-494.
Oh, S., Kang, H. 2016. Fast Occupancy Grid Filtering Using Grid Cell Clusters from LIDAR and Stereo Vision Sensor Data. *IEEE Sensors Journal*, 16(19): 7258–7266.
Omijeh, B. O., Ehikhamenle, M., Promise, E. 2015. Simulated Design of Water Level Control System. Computer Engineering and Intelligent Systems. 6(1): 30-40.
Ravankar, A., Ravankar, A.A., Kobayashi, Y., Hoshino, Y., Peng, C.-C. Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges. *Journal of Sensors*, 18(9): 3170.
Russakovsky, O., Deng, J., Su H., Krause J., Satheesh S., Ma, S., Berg, A.C. 2015. Imagenet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*., 115(3): 211–252.
Shanker, R., Jonas, A., Devitt, S., Huberty, K., Flannery, S., Greene, W., Swinburne, B., Wallace, R., Stentz, A., Thorpe, C., Moravec, H., Whittaker, W., Kanade, T. 1985. First results in robot road-following, Proceedings of the 9th International Joint Conference on Artificial Intelligence, 18–23 August 1985, CA, USA.
Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, J., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. 2017. Mastering The Game of Go Without Human Knowledge. *Nature*., 550-354.
Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lillicrap, T. 2017. Mastering Chess and Shogi By Self-Play with a General Reinforcement Learning Algorithm. https://arxiv.org/pdf/1712.01815.pdf-(Erişim tarihi: 05.08.2019).
Simonyan, K., Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale İmage Recognition. https://arxiv.org/pdf/1409.1556.pdf-(Erişim tarihi: 05.08.2019).
Sungur, İ., Akdur, R., Piyal, B. 2014. Türkiye'deki Trafik Kazalarının Analizi. *Ankara Medical Journal*, 14(3): 114 – 124.
Surendharan, S., Jennifer Ranjani, J. 2016. Environment Conscious Automated Vehicle Navigation System using PID Controller. *Indian Journal of Science and Technology*, 9(48): 1-5.
Szegedy, C., Liu W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D. 2015. Going Deeper With Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 7–12 June 2015, Boston, MA, USA.

Thorlabs, 2019. PID Kontrol Kazanç Değerlerinin Hesaplanması, https://www.thorlabs.com/tutorials.cfm?tabID=5dfca308-d07e-46c9-baa0-4defc5c40c3e-(Erişim tarihi: 11.08.2018).

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G. 2006. Stanley: The Robot That Won the DARPA Grand Challenge. *Journal of Field Robot*, 23(9): 661–692.

Treat, J. R., Tumbas, N. S., McDonald, S. T., Shinar, D., Hume, R. D., Mayer, R. E., Stansifer, R. L., Castellan, N. J. 1979. Tri-level study of the causes of traffic accidents. https://deepblue.lib.umich.edu/handle/2027.42/64993 (Erişim tarihi: 05.08.2019).

Udacity, 2018, Udacity Inc. Self-Driving Car Simulator.https://github.com/udacity/self-driving-car-sim-(Erişim tarihi: 06.08.2019).

Velazquez, M., Cruz, D., Garcia, S. 2016. Velocity and Motion Control of a Self-Balancing Vehicle Based on a Cascade Control Strategy. *International Journal of Advanced Robotic Systems*, 13(3): 1-11.

Visin F., Kastner, K., Cho, K., Matteucci, M., Courville, A., Bengio, Y. 2015. Renet: A Recurrent Neural Network Based Alternative to Convolutional Networks. https://arxiv.org/pdf/1505.00393.pdf-(Erişim tarihi: 05.08.2019).

Xin, J., Wang, C., Zhang, Z., Zheng, N. 2014. China Future Challenge: Beyond the Intelligent Vehicle, *IEEE Intelligent Transportation Systems.* 16(2): 8-10.

Xu, D., Jain, A., Anguelov, D. 2018. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation; Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 18–23 June 2018, Salt Lake City, UT, USA.

Yang, Q., Li, G., Kang, X. 2008. Application of Fuzzy PID Control in The Heating System. 2008 Chinese Control and Decision Conference, 2-4 July 2008, Yantai, Shandong, China.

Yiping, C., Wang, J., Li, J., Lu, C., Luo, Z., Xue, H., Wang, C. 2018. LiDAR-Video Driving Dataset: Learning Driving Policies Effectively; Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 18–23 June 2018, Salt Lake City, UT, USA.

Zhang, X., Fu, Y., Zang, A., Sigal, L., Agam, G. 2015. Learning classifiers from synthetic data using a multichannel autoencoder. http://arxiv.org/pdf/1503.03163.pdf-(Erişim tarihi: 05.08.2019).

Zhao, P., Chen, Y., Song, Y., Tao, X., Xu, T., Mei, T. 2012. Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID. *International Journal of Advanced Robotic Systems*, 9(44): 1-11.

Zhou, X., Gao, H., Jia, Y., Li, L., Zhao, L., Yu R. 2019. Parameter Optimization on FNN/PID Compound Controller for a Three-Axis Inertially Stabilized Platform for Aerial Remote Sensing Applications. *Journal of Sensors*, 2019(2):1-15.

Ziegler, J. ve ark. 2014. Making Bertha Drive an Autonomous Journey on a Historic Route. *Intelligent Transportation Systems Magazine*, 6(2): 8-20.

Zimit, A. Y., Yap, H. J., Hamza, M. F. Siradjuddin, I. 2018. Modelling and Experimental Analysis Two-Wheeled Self Balance Robot Using PID Controller. International Conference on Computational Science and Its Applications, 2-5 July 2018, Melbourne, VIC, Australia.

ÖZGEÇMİŞ

Adı Soyadı : Koray AKİ
Doğum Yeri ve Tarihi : Mersin / 1985
Yabancı Dil : İngilizce

Eğitim Durumu
Lise : Yahya Günsür Mesleki Ve Teknik Anadolu Lisesi, 2003
Lisans : Çanakkale Onsekiz Mart Üniversitesi, Bilgisayar ve Öğretim Teknolojileri Öğretmenliği, 2009

Çalıştığı Kurum : Bursa Uludağ Üniversitesi Orhangazi Yeniköy Asil Çelik MYO

İletişim (e-posta) : korayaki@uludag.edu.tr

Yayınları :

Aki, K., Karasulu, B. 2014. Esnek Hesaplamada Sinirsel Bulanık Sinerjiyi Temel Alan Sistemler ve Yaklaşımlar Üzerine Bir İnceleme. *Cumhuriyet University Faculty of Science Science Journal (CSJ)*, 35(2): 54-86