T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN ÖĞRENME TABANLI NESNE TAKİBİ

Bilen BAŞARIR
Orcid No: 0000-0002-6459

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

(Danışman)

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

BURSA - 2019

TEZ ONAYI

Bilen BAŞARIR tarafından hazırlanan "Derin Öğrenme Tabanlı Nesne Takibi" 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

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

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

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

Yukarıdaki sonucu onaylarım.

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
18../9/2019 **(Tarih)**

B.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.

18./9./2019
Bilen BAŞARIR
İmza

ÖZET

Yüksek Lisans Tezi

DERİN ÖĞRENME TABANLI
NESNE TAKİBİ

Bilen BAŞARIR

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ahmet Emir DİRİK

Bu tezde, derin öğrenme tabanlı SSD (Single Shot Multibox Detector) algoritmasını kullanarak, hareket eden kişileri takip eden ve bir lazer işaretçi ile hareket eden kişiye nişan alan bir sistemin tasarlanması ve gerçeklenmesi amaçlanmaktadır. SSD yöntemi nesne tespit konusunda literatürdeki en başarılı yöntemlerden biridir. Geliştirilen sistemin nişangahının yatay ve dikey hareketleri 2 adet adım motoru ile kontrol edilmektedir. Geliştirilen sistemim performansı ve isabetli vuruş istatistikleri deneysel testlerle ölçülmüştür. Bulunan sonuçlar gerçek zamanlı olarak bilgisayar ortamında kaydedilerek akabinde sonuçlar istatistik olarak yorumlanmıştır.

Anahtar Kelimeler: Derin öğrenme, Evrişimsel ağlar, SSD, Nesne tanıma, Adım motoru, Arduino
2019, x + 76 sayfa

ABSTRACT

M.Sc. Thesis

DEEP LEARNING BASED
OBJECT DETECTION

Bilen BAŞARIR

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

Supervisor: Assoc. Prof. Dr. Ahmet Emir DİRİK

In this thesis, using deep learning based SSD (Single Shot Multibox Detector) algorithm, it is aimed to design and implement a system that follows and aims the moving people a laser pointer. SSD method is one of the most successful methods of object detection in the literature. The horizontal and vertical movements of the developed system are controlled by 2 step motors. Improved system performance and accurate hit statistics were measured by experimental tests. The results were recorded in real time in a computer environment and interpreted as statistically.

Keywords: Deep learning, CNN, SSD, Object detection, Step motor, Arduino
2019, x + 76 pages

TEŞEKKÜR

Akademik hayatım boyunca her daim beni destekleyen merhum annem ve babama, beni motive ederek her zaman ne kadar yoğun olursa olsun bana zaman ayıran danışman hocam Sayın Doç. Dr. Ahmet Emir Dirik'e, makinanın dizaynı ve bazı parçaların üretimi konusunda desteklerinden dolayı Ali Osman Güler beye ve takip sisteminin üretimi ve montajı konusunda hiçbir yardımı esirgemeyen sevgili dostum Levent Durak beye teşekkürlerimi bir borç bilirim.

Bilen BAŞARIR
.../.../2019

İÇİNDEKİLER

Sayfa

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
YSA	Yapay Sinir Ağı
CNN	Evrişimsel Sinir Ağı
mAP	Ortalama Hassasiyet
GPU	Grafik işlemci ünütesi

Çeviriler	Açıklama
Labelling	Etiketleme
Clasification	Sınıflandırma
Gradient Descent	Kademeli iniş
Momentum	Ağırlık düşürme katsayısı
Stochastic	Rastgele
Overfitting	Aşırı öğrenme
Underfitting	Yetersiz Öğrenme
Regularization	Düzenlileştirme
Restricted	Sınırlı
Dropout	Seyreltme
Batch Normalization	Yığın normalleştirme
Vanishing gradient	Kaybolan eğim
Average Pooling	Örneklemelerin ortalaması
Segmantasyon	Bölütleme
Bounding box	Sınırlandırma kutusu
Ground Truth	Zemin gerçeği
Stride	Kaydırma
Zero-Padding	Sıfır ekleme
Confidence	Güven
True Positive	Doğru Pozitif
False Positive	Yanış Pozitif
Bias	Yanlılık
Recall	Geri çağırma
Precision	Kesinlik
Backpropagation	Geriye Yayılım
Epoch	Tur
Nonlinearity	Doğrusal olmama

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Yapay zekâ günümüzde birçok alanda ve uygulamada başarı ile kullanılmaya başlanmıştır. Dolayısıyla geleceği yapay zekânın şekillendirmesi kaçınılmaz görülmektedir. Yapay zekânın gelecekte nereye varacağı hem merak uyandıran hem de korkutan bir konudur. McCarthy ve ark. (2006), 1956 da bulunan bir kavram olan yapay zekâ ancak 2000'li yıllarda uygulanabilmeye başlanmıştır. Bunun sebeplerinden biri yapay zekânın ihtiyaç duyduğu işlem gücünün çokluğudur. Zaman içerisinde matematiksel işlemleri gerçekten hayret verecek hızlarda yapabilen grafik işlemci birimi (GPU) gibi donanımların oluşmasıyla gerekli bu işlem gücü ancak sağlanabilmiştir. Tabii ki yapay zekanın gelişiminin tek sebebi bu da değildir. Yapay zekânın ihtiyacı olan verilerin artması, veriye ulaşımın kolaylaşması ve bilgisayar dillerinin gelişimi yapay zekânın uygulanabilmesini sağlamıştır.

Mühendislik alanında bir çok kişiye mucize gibi görülen gelişmelerin aslında çok basit kavramlara dayandığını görürüz. Yapay zekâ konusun da aynen böyledir. Bilgisayarlar ikili sayı sisteminde sayıları toplayabilir, çıkarabilir, çarpabilir ve bölebilirler. Yapay zekâ denilen olgu da aslında tamamen matematiksel çeşitli işlemlerden oluşur. Yapay zekâ matematiksel ve olgusal alanlarda biyolojideki sinir hücrelerinin yani Latince adıyla nöronların çalışma prensibinden etkilenmiştir. Nöronları basit bir şekilde matematiksel olarak taklit ederek çalışır. Birçok mühendislik uygulaması doğada var olan kavramların anlaşılarak taklit edilmesi üzerine kuruludur. Nasıl düşündüğümüzün aslında bizim bildiğimiz tam bir cevabı olamamasına rağmen beyin ve sinir hücreleri üzerinde yapılan çalışmalarda sinir hücrelerini birbirlerine akson ve dendrit denilen bağlantılarla bağlanarak gerçekten çok karmaşık bir yapı oluşturdukları ve bu hücrelere bir elektrik sinyali verildiğinde bir hücreden diğer bir hücreye bu sinyalin belirli şartların gerçekleşmesi ile arttırılarak yada azaltılarak veya hiçbir şekilde geçmediğinin gözlemlenmesiyle "düşündüğümüz" anlaşılmıştır. Bu yapının matematiksel olarak modellenmesi ile de yapay zekâ yöntemleri/sistemleri geliştirilmektedir. Yapay zekâ beynimizin fonksiyonlarını bilgisayarda gerçekleştirmeye çalıştığına göre bunu belirli kriterlere göre başardığı ölçüde insanın

yerini alacaktır. Aynı robot kolların kaynak, taşıma ve montaj alanlarında fabrika işçilerinin yerini aldığı gibi. Tabi bu durum sadece işçilik alanında kalmayarak tıp, adalet, mühendislik, sanat, askeriye gibi birçok alanda insanın yerini yapay zekâ alabilecektir. Bu tezde yapay zekâ tabanlı bilgisayarla görme sistemleri geliştirilerek, kişi/nesne takibi uygulaması yapılmıştır. nesne tanıma ve takip görmenin ve yapay zekânın kesiştiği ve gerçek hayat uygulamalarının da bir hayli olduğu bir konudur. Geliştirilen sistem ile birçok nesne sınıfının tespiti ve takibi yapılabilir ve hem kuramsal hem de maddi önemi olabilecek konular uygulamalara yol gösterilebilir. Bu tez 5 kısımdan oluşmaktadır. Bu bölümlerin başlıkları ve içerikleri şu şekildedir.

1-GİRİŞ: Bu kısımda tez konusuna genel bir bakış açısıyla yaklaşılmakta problemin tanımı, önemi ve amacı konusunda açıklamalarda bulunulmaktadır.

2-KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI: Bu kısmında ise nesne tespitinin gelişimi ve bu alanda kullanılan konvansiyonel yöntemlerden genel olarak bahsedilmekte, nesne takibinin temel konuları olan yapay zekâ ve derin öğrenme, evrişimsel sinir ağları gibi konular detaylı bir şekilde anlatılmaya çalışılmaktadır.

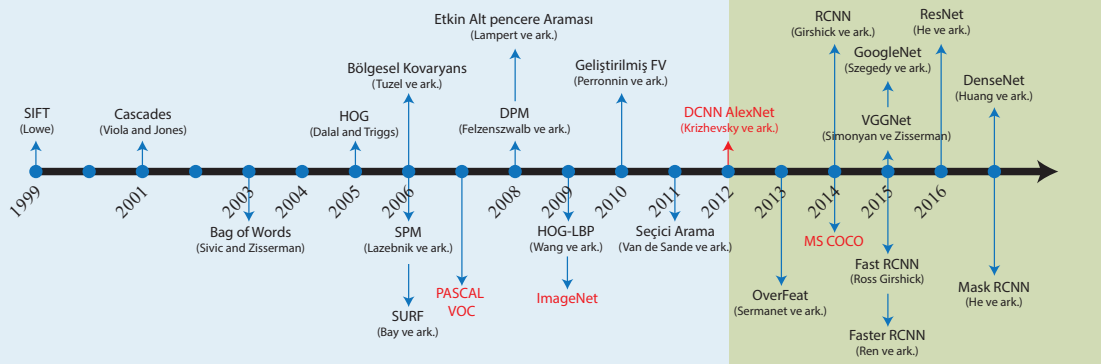
3-MATERYAL ve YÖNTEM: Bu kısımda güncel nesne tespit yöntemlerinden bahsedilmekte SSD yönteminin detaylarına girilerek fiziksel olarak hareketli bşr nesneyi takip edebilen bir cihazın tasarım, donanım ve yazılımsal detaylarından bahsedilmektedir.

4-BULGULAR: Bu kısımda geliştirilen nesne takip sisteminin performans sonuçları deneysel verilen ile sunulmaktadır.

5-TARTIŞMA ve SONUÇ: Bu kısımda bulunan neticelerin değerlendirilerek yorumlaması yapılmış, geliştirilen sistemin eksik ve kuvvetli yanları ortaya konulmuştur. Ayrıca geliştirilen sistemle ilgili geliştirilebilecek noktalar ve gelecekte neler yapılabilineceğinden bahsedilmiştir.

2. KURAMSAL TEMELLER VE KAYNAK ARAŞTIRMASI

2.1. Nesne Tespiti Yöntemlerinin Gelişim Aşamaları

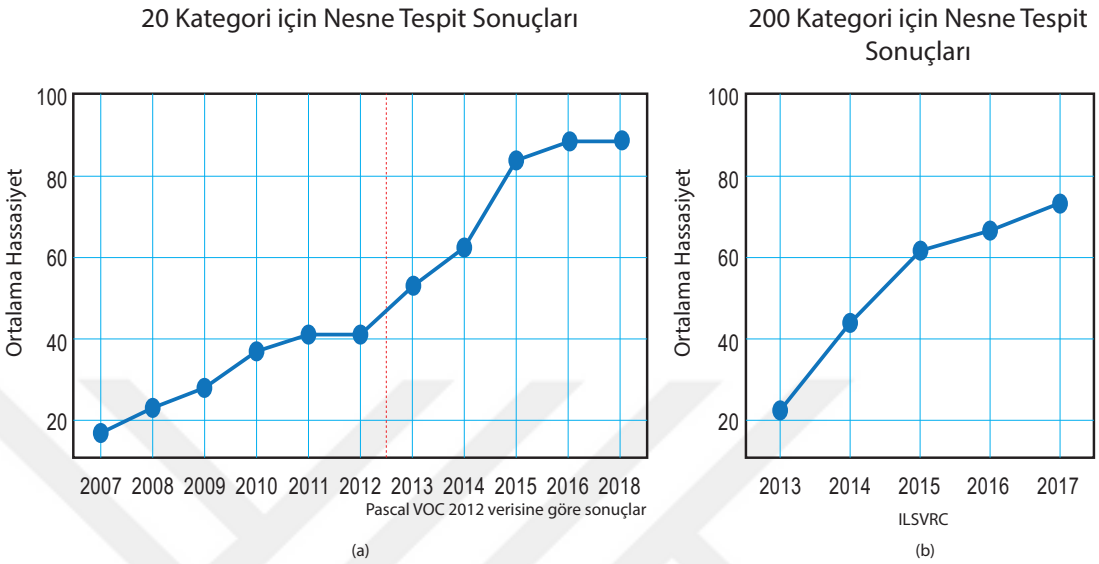


Şekil 2.1. Yapay zekâ ve öncesi kullanılan metotlar[1]

Yapay zekâ mimarilerinden önce bilinen en iyi nesne tespit yöntemleri, güvenilir temel seviye özniteliklerin bulunup kombine edilerek kullanılmasına dayanmaktaydı (Şekil 2.1 mavi kısım). Fakat bu metotlar nesnede elastik deformasyonlar olduğunda iyi sonuç vermiyordu. İyi bir nesne detektöründen nesnenin ötelenmesinden, döndürülmesinden, ölçeklendirilmesinden ve eğilip bükülmesinden etkilenmeden her durumda nesnenin yerini ve türünü tespit edebilmesi isteniyordu. Nesne tanıma teknolojisinin, her sene yapılan çeşitli yarışmalarla nasıl geliştiği dünya tarafından dikkatlice takip edilmekteydi. Fakat büyük uğraşlara rağmen, en büyük ilerlemelerin bile %30, %40'lar da olan mevcut başarı oranını yıllık sadece %5 geliştirebildiği görülmekteydi ve yeni yöntemler gerçek zamanlı nesne tespitinden çok uzaktı. Yarışmalar nesne sınıflandırma, nesne bulma (lokalizasyon) ve nesne bölütlemesi gibi çeşitli alanlarda yapılmakta idi. 2012 yılında Alexnet bu yarışma kategorilerinin hepsinde yapay zekâ algoritmalarını kullanarak bir çığır açtı (Şekil 2.1

[1] https://tinyurl.com/y6r429ab, Erişim Tarihi:14-07-2019, Saat: 13:00

kahverengi kısım). Bundan sonraki yıllarda yapay zekâ algoritmaları hızla gelişerek birçok alanda başarı oranlarını yüzde %90'ların üzerindeki bir seviyeye ulaştırdı.



Şekil 2.2. Yapay zekâ kullanılması ile başlayan gelişim süreci[2]

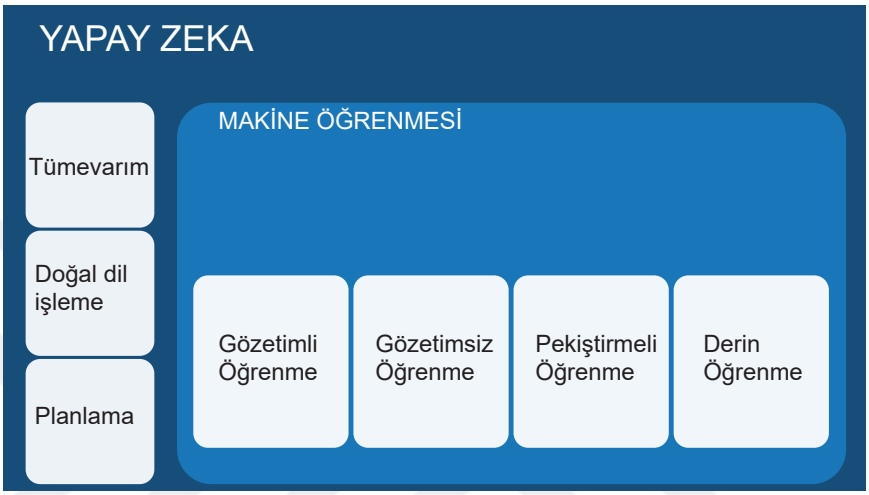
Şekil 2.2'de görüldüğü gibi yapay zekânın kullanılmaya başlandığı 2012 yılından önce teknolojinin çok yavaş ilerlediği hatta 2011 ve 2012 yıllarında kayda değer bir gelişmenin olmadığı sonraki yıllarda ise yapay zekânın kullanılmasıyla birlikte ortalama kesinlik (mean average precision)'in nasıl geliştiği görülmektedir. ILSVRC 200 kategori 2017 nesne tanıma yarışmasının [3] sonuçlarında (b) yine aynı şekilde yapay zekâ kullanılmaya başlamasıyla başarımın nasıl geliştiği gözükmektedir.

[2]https://tinyurl.com/y6r429ab, Erişim Tarihi:14-07-2019, Saat: 13:00

[3]https://tinyurl.com/yxcm8xut, Erişim Tarihi: 11-04-2019, Saat: 17:30

2.2. Yapay Zekâ

Yapay zekâ kendi içerisinde bir çok alt uzmanlık konusunu içermektedir. Makine öğrenmesi yapay zekâ kavramlarının bazı alt konularını içerir. Derin öğrenme de makine öğrenmesinin sadece bazı kavramlarını içerir ve bazen üst konuları metot olarak kendi bünyesine alır ve kullanır.



Şekil 2.3. Yapay zekâ, makine öğrenmesi konu kapsamları[4]

Andrew Moore' a göre: Yapay zekâ, bilgisayarların, son zamanlarda insan zekâsını gerektirdiği şekilde davranmasını sağlayan bilim ve mühendisliktir. Tom M. Mitchell'e göre makine öğrenmesi ise: Bilgisayar programlarının deneyim yoluyla otomatik olarak gelişmesini sağlayan bilgisayar yöntem ve algoritmadır. Makine öğrenmesi şu soruyu cevaplamaya çalışır: "Deneyimle otomatik olarak gelişen bilgisayar sistemlerini nasıl kurabilir ve tüm öğrenme süreçlerini yöneten temel yasaları nasıl bulabiliriz?" Derin öğrenme çok katmandan oluşan yapay sinir ağlarına verilen genel bir isimdir, bahsettiğimiz gibi makine öğrenmesinin bir alt kümesidir. Fakat bir alt dal olsa da günümüzde en çok kullanılan yapı derin öğrenme yapılarıdır.

Şekil 2.3'de yapay zekâ makine öğrenmesi ve derin öğrenme konuları ayrımı görülmektedir.

[4]https://tinyurl.com/ycdwes9f, Erişim Tarihi:01-07-2019, Saat: 09:00

Yapay zekânın konvansiyonel programdan en büyük farkı öğrenebilmesi kendisini geliştirebilmesidir. Bunu bir örnek vererek açıklayalım. Birçok kişi tarafından bilinen “Counter-Strike" oyununda sanal karakterlerden oluşan bir oyunda, konvansiyonel programlama örneğinde bu sanal karakterler sağa git, düşman yoksa ilerle, sonra sola git, düşman yoksa tekrar ilerle varsa ateş et, geri git gibi hep aynı şekilde hareket eden veya bunların hep aynı şekilde programcı tarafından oluşturulan varyasyonlarını yapan şekilde programlanır. Bu programlama yapısı belirli bir duruma çok uygun olsa dahi gelişemez ve güçlenemez. Eğer bir kere o sanal karakterleri yenmenin yolu bulunursa hep aynı şekilde yenilebilir. Burada önemli ifade “hep aynı şekilde” cümlesidir. İşte yapay zekânın üstünlüğü burada ortaya çıkmaktadır. Derin öğrenme algoritmaları kullanılarak programlanan sanal karakterler kendilerini muhtemel oluşabilecek bütün durumlarda oyuncuyu yenecek şekilde geliştirilebilirler. Kendisini sürekli yenecek bir sanal karakter ile kim savaşmak ister bilinmez ama gerçek dünyadaki uygulamaları düşündürücüdür. Düşünün, hiç sıkılmayan, korkmayan, mücadele gücünü yitirmeyen, aynı konuya süresiz odaklı, morali bozulmayan, acımayan, sadece başarıya odaklı ve gelişen bir şey ile ileride dost olmayan bir şekilde karşılaşabiliriz! Yapay zekânın öğrenme ve gelişme aşamaları aynı bir çocuğun kedinin kedi olduğunu öğrenme aşamalarına benzer. Muhtemel birkaç senaryo olsa da genellikle çocuk bir kedi görür işaret ederek bir ses çıkarır. Çevresindekiler ona kedi olduğunu söyler. Bu birkaç kere tekrarlandıktan sonra çocuk ona kedi demeye başlar. Sonra televizyonda bir kaplan gördüğünde ona da kedi der fakat çevresindekiler ona onun kedi olmadığını kaplan olduğunu söyler. Bir çocuk için kaplan ile kedinin karıştırılması gayet doğaldır. Çünkü benzer özellikleri vardır. Bu yanlışlar tekrarlanır, vaşak ile karıştırılır, çita ile karıştırılır ama her seferinde çocuk birbirine benzeyen sınıfları birbirinden daha iyi ayırt ederek öğrenir.

Gözetimli Öğrenme (Supervised Learning): Çocuğun kedi kavramını öğrenme aşamasında bahsettiğimiz kavram gözetimli öğrenme kavramıdır. Yani bir kavramın bütün varyasyonlarının bir öğretici tarafından anlatılarak öğretilmesi kavramına gözetimli öğrenme denir. Sistem belirli bir giriş verisi için bir hedef çıktısı üretecek şekilde uyarlanır. Aynı

kedi örneğinde olduğu gibi ne kadar çok kedi gösterirsek ve kedi olmayan fakat kediye benzeyen canlıları da gösterirsek, çocuk kediyi daha iyi öğrenir. Buradaki kediyi belirtme işlemi etiketleme (labeling) olarak adlandırılır. Amaç, yeni bir girdi için sistemi hedef çıktıyı tahmin edebilmesi için uyarlamaktır. Yani çocuğun daha önce hiç görmediği desenleri veya şekli farklı olan bir kediyi, kaplandan ve vaşaktan ayırarak kedi demesi amaçlanır.

Gözetimsiz Öğrenme (Unsupervised Learning): Gözetimsiz öğrenmede ise amaç; Barlow (1989)'e göre verilerin, benzerlik ve farklılıklarını bulmaya çalışmaktır. Örnek olarak çok uzaktaki bir gezegene ait ve bildiğimiz hiçbir yaşam formuna benzemeyen canlıların fotoğraflarının olduğu bir veri seti düşünelim. Bu canlıların ne oldukları bilinmemektedir. Fakat elimizdeki görüntüler değerlendirilerek canlı türlerinin benzerlikleri veya farklılıkları bulunabilir. Bu tür amaçlar için kullanılan algoritmaya gözetimsiz öğrenme denir. Sadece veri vardır ve bu verilerin ne olduğu bilinmemekte, yani etiketlenememektedir. Sadece kalıpları çıkarabilir. Bir başka deyişle denetimsiz öğrenme, yazılımın doğru cevaplar verilmeksizin verilerden öğrendiği makine öğrenmesi için bir yaklaşımdır. Uygun müşterileri hedeflemek için pazar bölümlemesi, bankacılık sektöründe anormallik, sahtekarlık tespiti, resim bölütlemesi gibi alanlarda kullanılır.

Yarı Gözetimli Öğrenme (Semi supervised learning): Bu öğrenme çeşidinde ise Chapelle ve ark. (2009)'de bahsedildiği gibi aynı denetimsiz öğrenmedeki gibi etiketlenmemiş veriler kullanılarak öğrenmeye başlanır ve bu verilerden özellikler çıkarılır. Bu çıkarılan özellikler sanki bir etiketmiş gibi kullanılarak gözetimli öğrenme görevlerinde kullanılır. Bu az miktardaki etiketlenmiş veriyle etiketlenmemiş, veriyi etiketlemeye çalışırız.

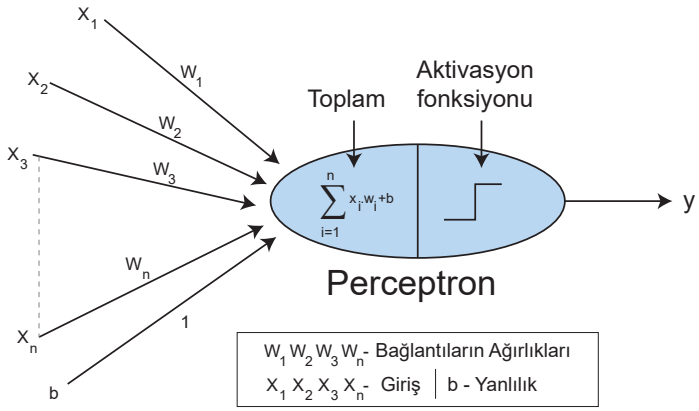
Pekiştirmeli Öğrenme (Reinforcement Learning): Bu yapıya en güzel kendi kendine bir oyunu oynayıp mükemmelleşen öğrenme yapılarıdır. Algoritmaya oyunun kuralları verilir ve başarı şartı belirtilir. Kaelbling ve ark. (1996)'de belirtildiği gibi derin öğrenme algoritması başarıyı garanti etmeye veya skoru maksimize etmeye çalışır. Günümüzde

konuyla ilgili en çarpıcı örnekler Google Deep-Mind ekibinden gelmektedir. Geliştirdikleri Alpha-Star kendi kendine Starcraft oyununu oynayarak Protoss adı verilen uzaylı ırkı ile dünya şampiyonlarını yenmiştir [5]. Aynı şekilde Tian ve ark. (2019)'da belirtildiği gibi Google Deep-Mind ekibi tarafından programlanan Alpha-Zero adlı program satrançta dünya şampiyonlarını ve şimdiye kadar konvansiyonel metotlarla geliştirilmiş en ileri satranç yazılımı olan Stokfish8'i yenmiştir. Üstelik sadece 4 saatlik kendi kendine oynadığı oyunlardan elde ettiği bilgiler ile bunu başarmıştır. Bu açıkça yapay zekânın gücünü göstermektedir. Olayın en ilginç tarafı StokFish8 yazılımı her hamle sonunda Alpha-Zero'yu ve kendisini değerlendirerek kendine ve karşıdaki oyuncuya kazanma şansının ne olduğunu belirtmiş ve oyunun son hamlelerine kadar kendisinin yendiğini iddia ederken birden aslında yenildiğini fark etmiştir [6]. Bu yapıya sürekli pekiştirmeli yapay sinir ağı (recurrent reinforcement neural network) denir. Birçok kişin tehdit olarak algıladığı esas algoritma aslında budur.

Nöron ve Perceptron Kavramları: İnsanoğlunun düşünme sisteminin en küçük parçaları nöron denilen beynimizde bulunan hücrelerdir. Bu nöronların trilyonlarcasının anlamlı bir şekilde birleşmesi sayesinde düşünebilmekteyiz. Aynı bu yapıya benzer şekilde yapay zekâ da Perceptron kavramı kullanılır. Bu kavram ilk olarak Rosenblatt (1958) adlı makalede, bir psikolog olan Frank Rosenbaltt tarafından kullanılmıştır. Bir Perceptronun yapısı aşağıdaki Şekil 2.4'ki gibidir:

[5] https://tinyurl.com/y3k47l7r, Erişim Tarihi: 14-07-2019, Saat: 18:30

[6] https://tinyurl.com/yd4y9p76, Erişim Tarihi: 01-08-2019, Saat: 00:30



Şekil 2.4. Nöronun matematiksel modeli (Perceptron)[7]

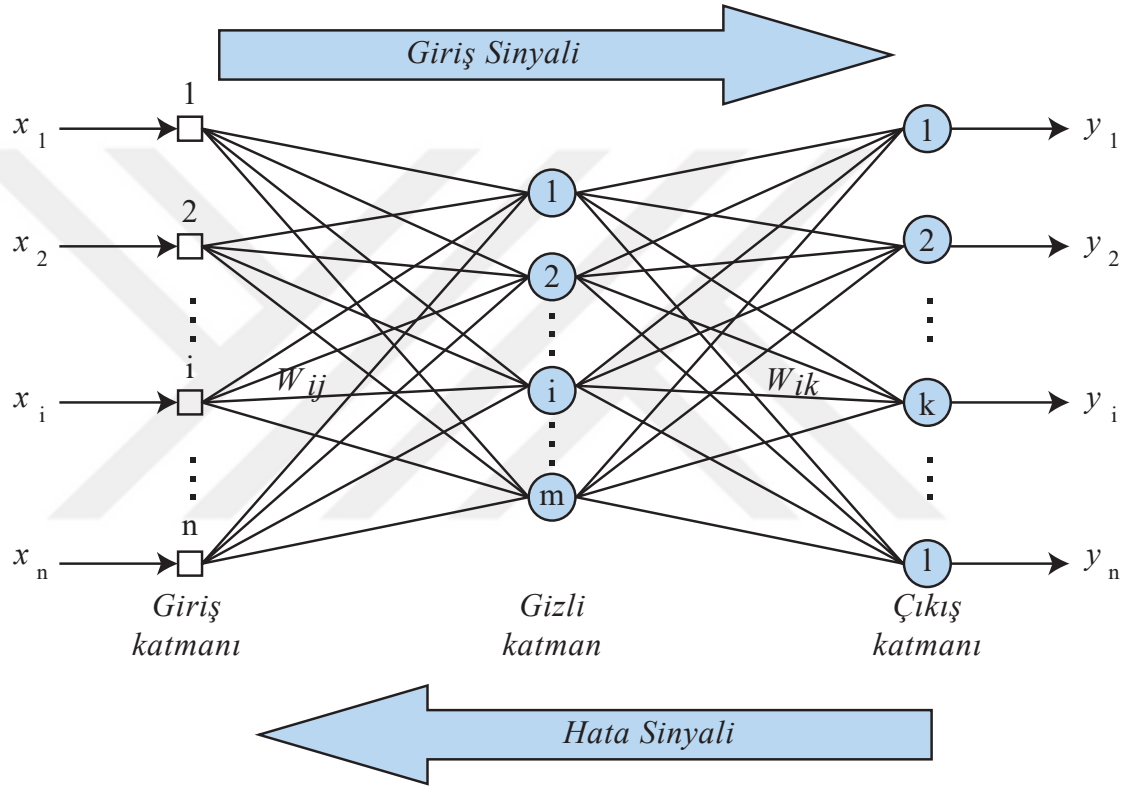
Perceptron şu şekilde çalışır. Şekil 2.4 aynı hat üzerindeki x_i değerleri ile W_i değerleri çarpılır sonra hepsi birden toplanır. Bu sayısal bir sonuç verir. Bu sayısal sonuç aktivasyon fonksiyonuna gönderilir ve aktivasyon fonksiyonundan geçen değer bize y çıkış değerini verir. Perceptron Şekil 2.4'de görüldüğü gibi gelen verileri belirli işlemler yaparak y sonuç değerine dönüştüren bir fonksiyondur. Perceptronlardan birçoğu yan yana ve alt alta eklenirse bir sinir ağı oluşur ve buna yapay sinir ağı denilir. Bir perceptronda x_i'ler sayısallaştırılmış giriş datalarıdır. x_i'ler bir resim matrisinin tüm değerlerini, müşteri memnuniyet anketindeki müşteri verilerini veya bir kanser hücresinin özelliklerini sembolize eden hücre kalınlığını, hücre yuvarlaklık değerini, mitakondri sayılarını temsil edebilirler. W_i değerleri ise yapay zekânın algoritmalarının yaptığı binlerce optimizasyon sonrası bulduğu sayılardır ve yapay zekâ algoritmalarının asıl amacı bu katsayıları bulmaktır. Başlangıçta W_i değerleri yapay zekâ algoritması tarafından rastgele verilir. b (yanlılık) değeri y değerini başlangıçta sıfır olmamasını böylece hesaplamaların her hâlükârda başlamasını sağlamak için sistemi dizayn eden tarafından verilir. b değeri de rastgele seçilen bir sayıdır ve her zaman verilmesi şart değildir; yine optimizasyonlar neticesinde b sayıları da optimize edilirler. Duruma göre sıfır da olabilirler. Aktivasyon fonksiyonu ise sistemimizin gelen değere göre çıktı değerini manipüle eden bir eşik filtresi gibidir. Ve sisteme eğrisellik (non-linearity) katmak için kullanılır. Sigmoid, TanHiberbolik, ReLU olarak adlandırılan birçok farklı türü vardır. Sonucunda da y değeri üretilir. Bilindiği üzere $f = aw + b$

[7]https://tinyurl.com/yxt8pgxn, Erişim Tarihi:10-07-2019, Saat: 15:00

şeklindeki bir fonksiyon doğru denklemini verir. Aktivasyon fonksiyonundan geçen doğru denklemi belirli bir şekilde lineerden eğilerek sistemi doğrusal olmayan bir hale getirir. Böylece sınıf farklılıklarını daha iyi bir şekilde ayırt etme kabiliyeti kazanır.

2.3. Yapay Sinir Ağları (Neural Network)

Yapay sinir ağlarında perceptronlar birbirine bağlanarak bir sinir ağını oluştururlar.



Şekil 2.5. Yapay sinir ağı yapısı[8]

Yukarıdaki Şekil 2.5'de yapay sinir ağı yapısı gösterilmektedir. Eğer bir veya iki katmandan oluşan bir yapı kurulmuş ise literatürde sığ, üç ve daha fazla katmandan oluşan bir yapı kurulmuş ise derin öğrenme yapısı olarak sınıflandırılır. Liang ve Bose (1996)'e göre geçmişte sığ yapılı mimariler basit problemlerin çözümü için yeterliyken, günümüzde daha fazla katmanlı mimariler karışık problemlerin çözümünde yeterli olabilmektedir.

[8]https://tinyurl.com/y57tfwpk, Erişim Tarihi:11-07-2019, Saat: 11:00

Sistem karışık problemlerin çözümünde başarı sağlayamıyorsa katman sayısı, perceptron sayısı ve/veya veri sayısı arttırılarak tekrar denenir. Sistem hiç bilmediği verileri sınıflayabilmeye ve nesneleri bulabilmeye başlayana kadar bu parametreler üzerinde değişiklikler yapılır. Doğru bir şekilde programlanabilmiş 5-6 katmanlı ve 150-200 perceptrona sahip bir derin öğrenme algoritması eğer yeterli sayıda veri verilirse %80 üzerinde bir başarıyla sınıflandırma (clasification) problemlerinde başarı sağlayabilmektedir. Örneğin MNIST (insanların el yazı karakterlerinin olduğu fotoğraf kümesi) veri seti üzerinde birçok algoritmanın başarı seviyelerini ilgili kaynaktan görebilmek mümkündür[9]. MNIST veri kümesinin resim çözünürlüğü 28×28'dir. İki katmanlı 1000 tane perceptrondan oluşan bir yapay sinir ağının bile %95 başarı gösterdiği görülebilmektedir. Perceptron konusunda bahsettiğimiz gibi bir yapar sinir ağı yine perceptonlara gelen x_i giriş değerleri ile başlar. Her perceptron kendisine gelen değerleri değerlendirerek bir y çıkış değeri üretir bu değerler kendisinden sonra gelen gizli katman için giriş sinyallerini oluşturur. Bütün gizli katmanlar bittikten sonra çıkış katmanındaki y_i değerleri oluşur. Eğer istenilen değerle üretilen aynı değilse bir hata sinyali oluşturularak W_i değerleri geriye yayılım metoduyla güncellenir tâ ki istenilen sonuçlar üretilinceye kadar sistem tekrar tekrar güncellenir.

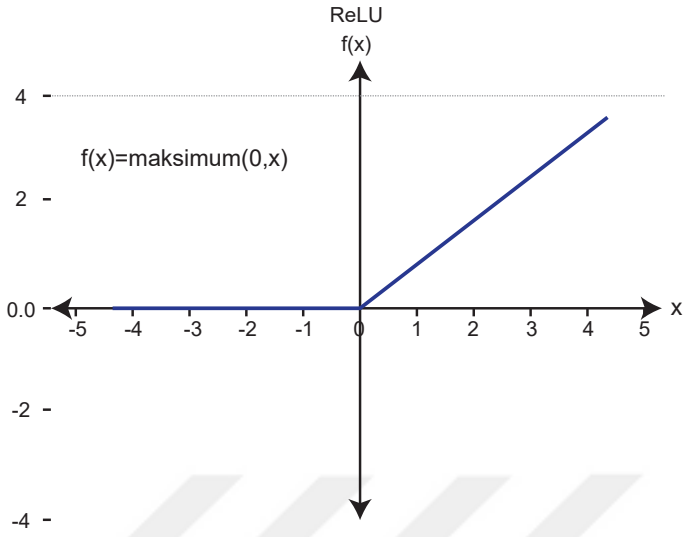
Görüntü işleme: Veriler yapay sinir ağlarına çoğunlukla bir ön işlem kümesinden geçirilerek verilir. Görüntü işleme olarak bahsedilen kavram dijital bir imajın kalitesini ve/veya özelliklerini iyileştirmek ve belirginleştirmek için kullanılan bir yöntemler bütünüdür. Bu yöntemlerle dijital bir imajın bulanıklığı, keskinliği, renkleri, ebatları gibi özellikler manipüle edilerek istenilen bir seviyeye getirilmesi amaçlanır. Genellikle çözünürlüğü daha yüksek bir kamera ile daha iyi nesne tespitinin yapılacağını düşünülebilir. Fakat bu yöntem güvenilirliği arttırarak olumlu bir etki yaratırken hızı önemli bir ölçüde düşürülür. Gerçekte nesne tespiti konusu, güvenilirlik ve hız arasında tercih yapılması gereken bir optimizasyon problemidir. Öğrenmenin hızlanması için kullanılan en popüler mimariler günümüzde 512×512'den daha büyük çözünürlük istemezler. Bu sebeple fotoğrafın genellikle ölçüleri küçültülür. Görüntüler derin öğrenme katmanlarına verilmeden önce

[9]https://tinyurl.com/cpbyrhs , Erişim Tarihi: 11-02-2019, Saat: 19:00

çeşitliliği arttırmak için ayna görüntüsünün alınması, bir eksen etrafında döndürme, renklerinin değiştirilmesi ve bozukluklara karşı dayanıklılığın arttırılması için bozma ve/veya düzeltme işlemlerine maruz bırakılabilirler. Bu işleme veri büyütme denir. Görüntü işleme yapıldıktan sonra kurulan mimariye göre değişmekle birlikte genellikle derin öğrenme katmanlarının ilki evrişimsel sinir ağları (CNN) katmanıdır.

Aktivasyon fonksiyonları: Aktivasyon fonksiyonu aslında yapay sinir ağlarında karar değeri üretmek için kullanılır. Yani perceptrona gelen değerler toplamı belirli bir değerin üzerindeyse o hücre aktif olur. Veya gelen değer belirli bir fonksiyona bağlı olarak sınırlandırılır ve dönüştürülür. Eğer değer görece küçük ise üretilen değer o hücrenin temsil ettiği eşiği aşmamış anlamına gelir. Eğer yapay sinir ağının ilerleyen bölümlerindeki perceptronlar da bulunan eşikler de aşılırsa sonunda bir karara ulaşılmış olur. Aktivasyon fonksiyonu YSA sistemine karmaşık problemlerin çözümü için gerekli olan doğrusal olamama (nonlinearity) özelliği kazandırır. Genel olarak sistemin çalışma şekli budur. Aktivasyon fonksiyonların en çok kullanılan biçimleri RELU, Sigmoid, TanH, SoftMax, SWISH tir.

1. ReLU

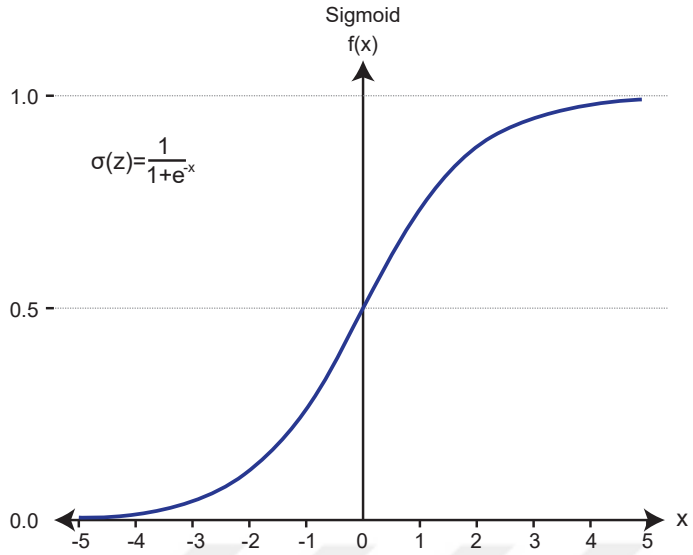


Şekil 2.6. ReLU

ReLU fonksiyonu olarak Hahnloser ve arkadaşları, 2000, biyolojik kavramların matematiksel ispatları için ve daha sonra Jarrett ve arkadaşları obje tespiti için önerilmiştir. Daha sonra sınırlı Boltzmann makinasında Nair ve Hinton (2010) tarafından kullanılarak popülerleştirilmiştir. Menteşe fonksiyonu da denir. Çünkü Şekil 2.6 bakılırsa şekli bir menteşeye benzer. Perceptrona gelen değer toplamı sıfırdan büyükse değerini korur, küçükse sıfır olur. YSA dizaynına başlamak için en uygun fonksiyondur. Bütün mimarilerde kullanılan en genel aktivasyon fonksiyonudur. Ve diğerlerine göre hesaplaması kolay olduğundan hızlıdır. Bu yüzden sistemin başarısı üzerinde çok büyük bir fark yaratmasa da hızından dolayı kullanılır.

2. Sigmoid

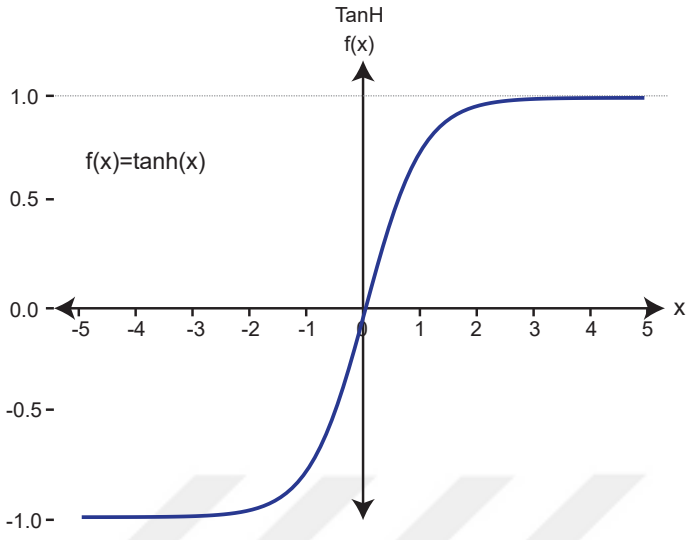
Sigmoid fonksiyonu Şekil 2.7'ye bakılırsa $x = 0$ da $y = 0,5$ değerini alır. Değerler $-\infty$'da 0'a yaklaşırken $+\infty$'da 1 değerine yaklaşır. Değerlerin değişme aralığının -4 ile +4 arasında olduğu durumda fonksiyoneldir. Çünkü bu değerlerin dışında sonuçları 0 ila 1'e çok yaklaştırır. Özellikle bir olasılığı çıktı olarak tahmin etmemiz gereken modeller için



Şekil 2.7. Sigmoid fonksiyonu

kullanılır. Bir durumun olasılığı sadece 0 ile 1 arasında ise, sigmoid fonksiyonu seçilebilir. İkili tercih durumunda softmax yerine kullanılabilir. Bazı uygulamalarda öğrenmenin takılmasına 0 ile 1'e çok yaklaşma yüzünden sebebiyet verdiğinden genellikle softmax tercih edilir. Softmax'ın diğer bir güçlü tarafı da çoklu sınıflandırma yapabilmesidir. İkili sınıflandırma sistem çıktısının ya 0 ya da 1 olduğu sadece iki değerin sistemden çıkabileceği durumlardır. Çoklu sınıflandırma ise çok sınıfın ayrıt edilmesi demektir.

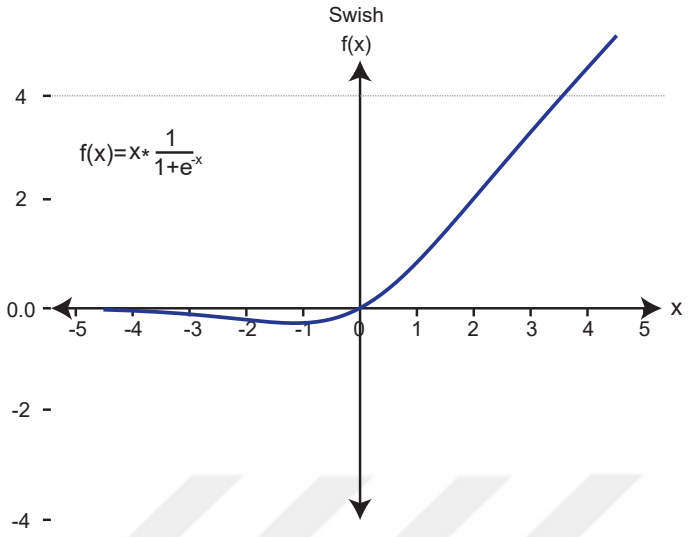
3. TanH



Şekil 2.8. TanH fonksiyonu

Şekil 2.8'da görülen TanH fonksiyonu aynı sigmoid fonksiyon gibi davranır tek farkı değerler simoid fonksiyonundaki gibi 0 ila 1 arasında değil de -1 ila 1 arasına ölçeklenir. Değerlerin değişme aralığının -3 ile +3 arasında olduğu yerlerde iyi sonuç verir. Bundan büyük ve küçük değerlerde -1'e +1'e çok yaklaşık sonuçlar verdiğinden sınıfları ayırt etmekte zorlanır.

4. SWISH



Şekil 2.9. Swish fonksiyonu

Şekil 2.9'de görüldüğü gibi swish fonksiyonu ReLU aktivasyon fonksiyonuna çok benzer. Aynı ReLU'da olduğu gibi 0 dan büyük değerler kendi değerine çok yakın değerler alırken 0'a çok yakın eksi değerler sıfırdan farklı ama sıfıra çok yakın değerler alır ve Pascanu ve ark. (2012) göre kaybolan eğim (vanishing gradient) problemini önler. Nwankpa ve ark. (2018) RELU ya alternatif olarak swish aktivasyon fonksiyonunu önermiştir. Bunun sebebi ImageNet gibi bazı bilinen mimarilerde doğrulukta ve hızda %1'lik bir kazanç sağladığından dolayıdır [10].

[10]https://tinyurl.com/yykjzttl, Erişim Tarihi: 17-08-2018, Saat: 11:20

5. SoftMax

$$q(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Softmax genellikle yapay sinir ağının çıkış katmanında sınıflandırma için kullanılır. Softmax fonksiyonu, çoklu sınıflandırma için kullanılan daha genelleştirilmiş bir aktivasyon işlevidir. Kendisine gelen elemanlar vektör şeklindedir ve 0 ila 1 arasındaki yüzde olasılıklarını gösterir. Örneğin 5 adet hayvan türüne ait (kedi, köpek, ördek, tavşan, tavuk gibi) sınıflandırma yapılmak istendiğinde. Yapay sinir ağımıza giren fotoğrafın ne olduğunu yapay sinir ağımızdan çıkan sınıf yüzdelerin içerisindeki en yüksek değer kullanılarak karar verilir.

Çizelge 2.1. Sınıflar

	SINIF	OLASILIK
1.	KEDİ	%15
2.	KÖPEK	%15
3.	ÖRDEK	%1
4.	TAVŞAN	%68
5.	TAVUK	%1

Eğer doğru çalışan bir yapay sinir ağımız varsa yukarıdaki Çizelge 2.1'de görüldüğü gibi yapay sinir ağımıza giren fotoğrafın Tavşan sınıfı olduğu tespit edilmiştir. Softmax foksiyonundan çıkan en büyük değer yapay sinir ağına giren görüntünün sınıf adı olarak seçilir.

Çizelge 2.2. Sınıfların softmax fonksiyonuna göre hesaplanması

1.	x=Son katmana gelen sayılar	2,708	2,708	0	4,22	0
2.	x sayıları e^x'e göre hesaplanır	15,00	15,00	1,00	68,03	1,00
3.	e^x'e göre hesaplanan sayılar toplanır	100,03				
4.	2. aşamadaki sayılar sırasıyla toplama bölünür	0,15	0,15	0,01	0,68	0,01

Yukarıdaki Çizelge 2.1'de görüldüğü gibi son katman kedi, köpek, ördek, tavşan, tavuk sınıflarının yüzde değerlerini vermiştir. En yüksek değer olan sınıf %68 ile tavşan sınıfına

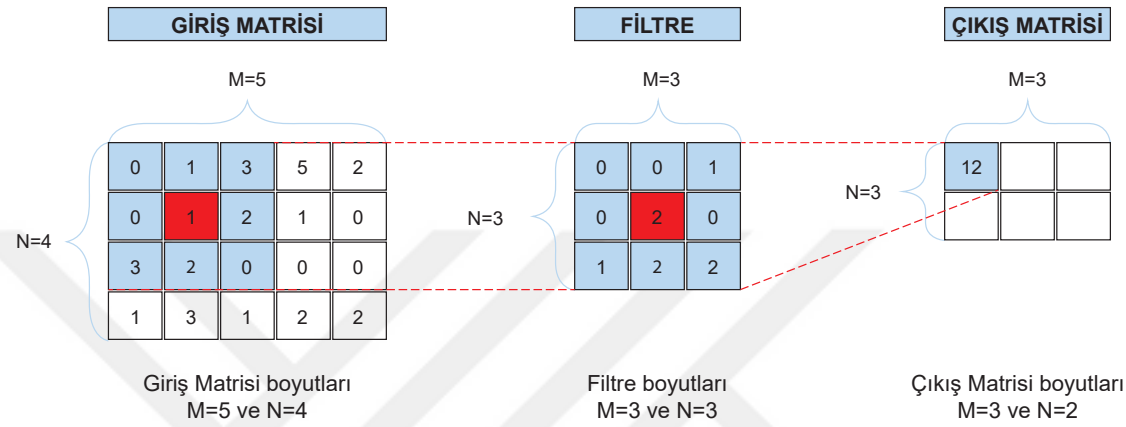
ait olduğu için yapay sinir ağına giren görüntünün tavşan olduğuna hükmedilir. Görüldüğü gibi Softmax genellikle yapay sinir ağının son katmanında kullanılan ve çoklu sınıfları ayırt etmemizi sağlayan ve yaygın olarak kullanılan bir fonksiyondur. Çizelge 2.2'de ise bu olasılıkların nasıl bulunduğu gösterilmektedir. Genel olarak bir yapay sinir ağı kurgulanırken hangi aktivasyon fonksiyonunu kullanılacağına karar vermek için sistemde çeşitli aktivasyon fonksiyonları denenmelidir. Denenen aktivasyon fonksiyonları içerisinde hatayı en az yapan aktivasyon fonksiyonu seçilmelidir. Aşağıda verilen bağlantıda [11] görüldüğü gibi belirtilenlerden başka aktivasyon fonksiyonları da vardır. Eğittiğimiz yapay sinir ağındaki başarı ve hız sonuçları gibi performans göstergeleri değerlendirilerek, seçilen aktivasyon fonsiyonu değiştirilmelidir.

2.4. Evrişimsel Sinir Ağları (CNN)

CNN'i anlamak için önce konvolusyon işlemi hakkında bilgi sahibi olmak gerekmektedir. İki fonksiyon arasında özel tanımlanmış bir işlemdir ve bu iki fonksiyondan üçüncü bir fonksiyon oluşturulur. Bu üçüncü fonksiyon ikinci fonksiyonun birinci fonksiyonu nasıl biçimlendirdiğini gösterir. Evrişim işlemi olasılık, istatistik, bilgisayarlı görü, doğal dil işleme, görüntü ve sinyal işleme, mühendislik ve diferansiyel denklemler gibi birçok mühendislik problemlerinde kullanılır. Asterix veya yıldız işlemi ile tanımlanır. $(f * g)(t)$ şeklinde gösterilir. Tek boyutlu fonksiyonlar için bu işlem f fonksiyonundan g fonksiyonu geçirilerek zaman içerisinde bu ikisinin kesişmesinden oluşan alanın eğrisini verir. Aynı tek boyutlu evrişim işlemleri olduğu gibi iki, üç boyutlu ve daha çok boyutlu evrişim işlemleri de mevcuttur. Krizhevsky ve ark. (2012)'da belirtildiği gibi nesne sınıflarını bulmada gözetimli öğrenme metodları ile birlikte en çok kullanılan yöntemdir. Krizhevsky ve ark. (2012)'nın yaptığı çalışmada nesne sınıflarını bulmada imagenet kullanılmıştır. Yapay zekâ uygulamaları ve bilgisayarlı görü konusunda en çok kullanılan evrişim işlem biçimi 2 ve 3 boyutlu evrişim işlemleridir. Kameralardan elde ettiğimiz renkli fotoğraflar 3 katmanlıdır (Kırmızı, yeşil, mavi) ve matris formundadır. Bu kanallar görüntü işleme

[11] https://tinyurl.com/h5kq28p, Erişim Tarihi: 16-07-2019, Saat: 12:30

ve yapay zekâ da sınıflandırma, tespit ve bölütleme gibi işlemlerde girdi olarak kullanılır. Nesne tespiti için Oquab ve ark. (2015) çalışmasında evrişimsel sinir ağları kullanılarak nesne tespitinin nasıl yapıldığından bahsetmektedir. 2 boyutta konvolusyon işlemi ise özet olarak aşağıdaki gibi yapılır.



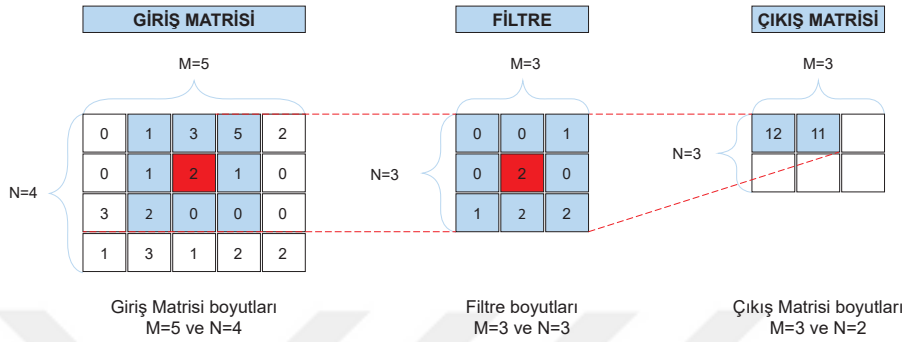
Şekil 2.10. İki Boyutlu evrişim işlemi 1. aşama

Şekil 2.10'de görüldüğü üzere eni ve boyu $M \times N$ olan üç adet matris vardır. Bu matrislerden birincisi giriş matrisidir. İkinci matris giriş matrisi üzerinde işlemler yapacağımız filtre matrisidir ve ebadı 3×3 dür. Giriş matrisi ile filtrenin evriştirilmesi ile üçüncü matrisimiz olan çıkış matrisimiz bulunur. Çıkış matrisimizin boyutu seçilen bazı parametrelere göre otomatik olarak hesaplanır. İki matrisin evriştirilmesi işleminde filtre matrisindeki orta ve kırmızı ile işaretlenmiş 2 değeri giriş matrisindeki ilk ve yine aynı renkteki 1 değerinin üzerinde oturtulur. Ve denklem 2.1'deki işlemler gerçekleştirilir.

$$0 \times 0 + 1 \times 0 + 3 \times 1 + 0 \times 0 + 1 \times 2 + 2 \times 0 + 3 \times 1 + 2 \times 2 + 0 \times 2 = 12 \quad (2.1)$$

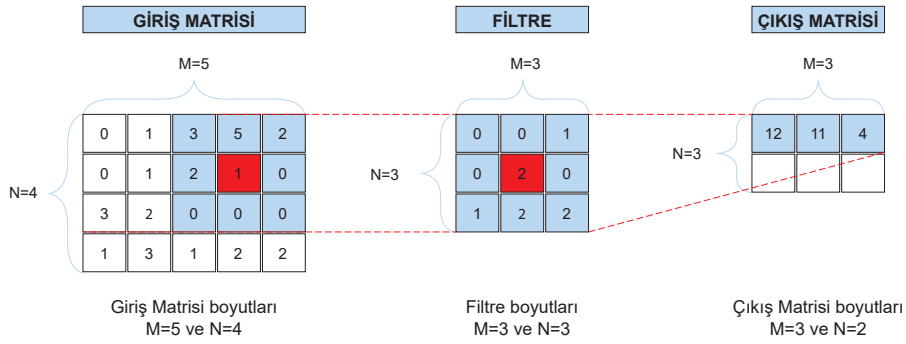
Görüldüğü gibi denklem 2.1'e göre çıkış matrisimizin en üst sol değeri 12 olmuştur. Bu işlem aynı şekilde filtreyi sağ tarafa doğru birer birer sonrasında tekrar başa dönüp

bir aşağıya ve yine sağ tarafa doğru birer birer kaydırarak tâ ki tüm giriş matrisi filtre ile evriştirilene kadar devam eder. Çıkış matrisinin üst orta değeri aşağıdaki gibi hesaplanır.



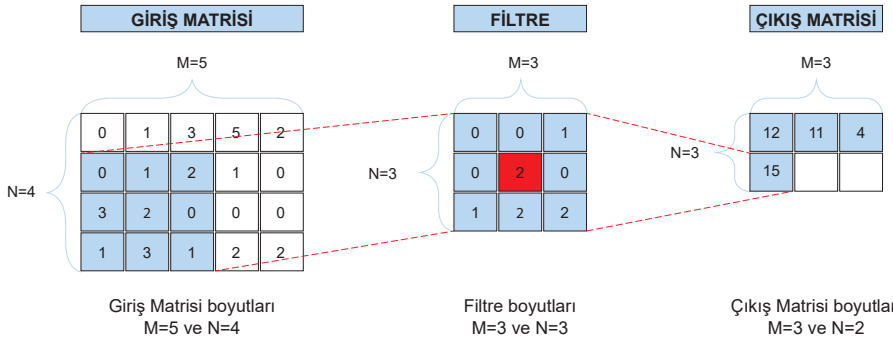
Şekil 2.11. İki Boyutlu evrişim işlemi 2. aşama

İşlem bir önceki gibi giriş matrisinin üzerinde filtrenin bir sola kaydırılmasıyla devam eder.



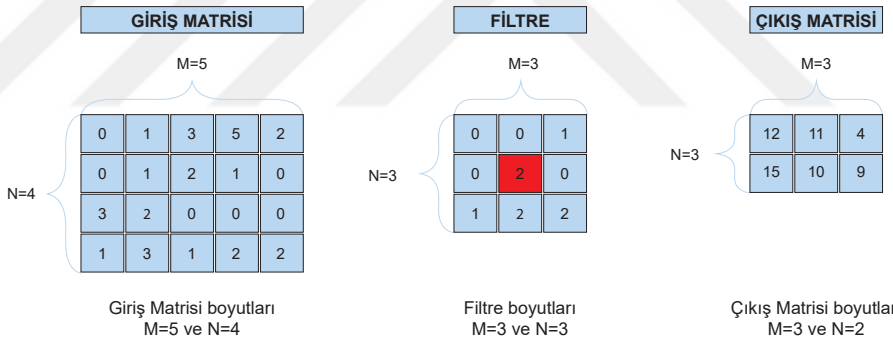
Şekil 2.12. İki boyutlu evrişim işlemi 3. aşama

Şekil 2.12'de görüldüğü gibi üst satır tamamen evrişim işlemine tabi tutulmuştur. Artık filtre en sağa ve bir alta kaydırılarak denklem 2.1'deki işlemler tekrarlanır.



Şekil 2.13. İki boyutlu evrişim işlemi 4. aşama

Şekil 2.13'da görüldüğü gibi filtre görüntü matrisinde sağ ve bir alt satıra kaydırılmış ve denklem 2.1'deki işlemler uygulanmıştır.



Şekil 2.14. İki boyutlu evrişim işlemi son aşama

Şekil 2.14'da görüldüğü gibi giriş matrisinin bütününe filtre yardımıyla evrişim işlemi uygulanmıştır. Evrişim işlemi görüldüğü gibi basittir fakat gerçek uygulamalardaki etkileri çok büyüktür. Bu sayede iki fonksiyonun anlamlı bir birleşimi elde edilir. Oluşan çıkış matrisi aslında giriş matrisine filtre uygulanarak bazı özelliklerinin öne, bazı özelliklerinin ise geriye bırakılmasını sağlar. Şekil 2.10 örneğinde evrişim işlemi uyguladıktan sonra

çıkış matrisinin boyutunun bazı parametrelere bağlı olduğu belirtilmişti. Bu parametreler giriş matrisinin boyutları, filtrenin boyutları, kaydırma ve doldurma işlemlerinin miktarıdır.

Kaydırma (Stride): Şekil 2.10 ve devamındaki evrişim işlemi örneklerine bakılırsa filtremiz giriş matrisinde sadece bir birimlik sola ve aşağıya kaydırılarak çıkış matrisi elde edilmiştir. Bu işleme kaydırma denir. Kaydırma ne kadar çok olursa çıkış matrisimiz o kadar küçülür. Kaydırma maksimum örnekleme (maxpooling) işlemi denilen ve bir matristeki seçilen belirli bir alandaki değerlerin en büyüklerinin alınarak yeni bir matris oluşturulması olarak tarif edebileceğimiz işlemin yerine artık günümüzde daha yaygın olarak kullanılmaktadır. Bunun sebebi hem hız hem de giriş matrisinin daha iyi özetleyebilme kabiliyetidir.

Sıfır Ekleme (Zero-Padding): Giriş matrisinin boyutlarını büyüterek çıkış matrisinin boyutunun giriş matrisi ile aynı olmasının istendiği durumlar için giriş matrisinin çevresine sıfır değeri eklenerek giriş matrisinin boyutlarının büyütülmesi durumudur.

Kaydırma ve dolgulama kavramlarının çıkış matrisinin boyutları arasındaki formül: ÇB=Çıkış matrisinin boyu, GB=Giriş matrisinin boyu, FB=Filtrenin boyu, K=Kaydırma miktarı, D=Dolgulama miktarı olarak alırsak şu şekildedir:

$$\text{ÇB} = \frac{\text{GB-FB} + 2 * D}{\text{K}} + 1 \tag{2.2}$$

Dolayısı ile Şekil 2.6'daki örneği ele alırsak denklem 2.2'ye göre

$$\text{ÇB} = \frac{5 - 3 + 2 * 0}{1} + 1 = 3 \tag{2.3}$$

İki boyutlu matrisimizin eninin çıkış boyutu denklem 2.3'de hesaplanmıştır. Hesaplamamız gereken diğer boyut da giriş matrisimizin boyudur.

$$\text{ÇB} = \frac{4 - 3 + 2 * 0}{1} + 1 = 2 \tag{2.4}$$

denklem 2.4'de görüldüğü gibi hesaplanmaktadır. Eğer Kaydırma miktarımız 2 olsaydı ve dolgulama miktarımız 0 olsaydı bu durumda:

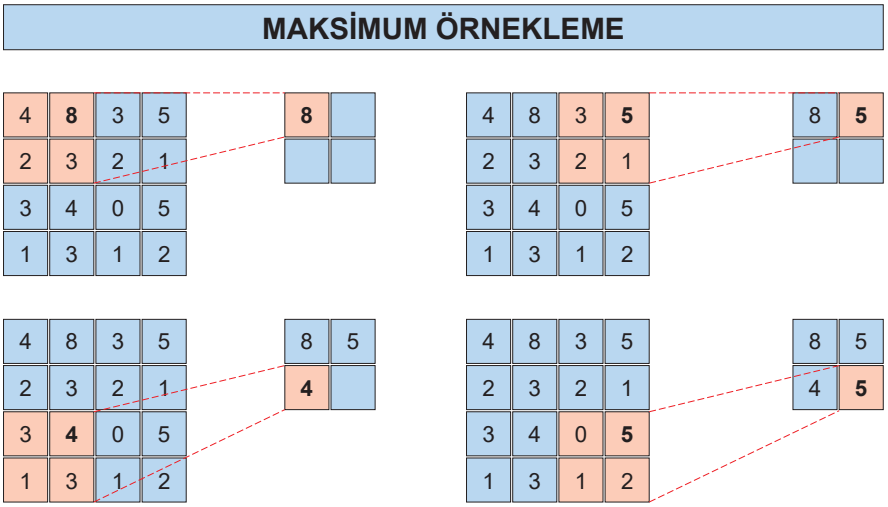
$$\text{ÇB} = \frac{5 - 3 + 2 * 0}{2} + 1 = 2 \tag{2.5}$$

Aynı durumda çıkış matrisimizin boyuda

$$\text{ÇB} = \frac{5 - 3 + 2 * 0}{2} + 1 = 1,5 \text{ olur} \tag{2.6}$$

Dikkat edilirse denklem 2.6'den sonuç olarak 1,5 değeri çıkmıştır. Gerçekte 1,5 gibi bir değerin piksel değeri olamayacağı için bu gibi durumlar için kendinden küçük en yakın tam sayıya yuvarlatılarak 1 değeri kullanılır. Aslında bu durum 2 kaydırma yapılamayacağı anlamına gelmektedir. Eğer muhakkak 2 kaydırma yapılmak isteniyorsa sıfır ekleme işleminin yapılması gerektiği ortaya çıkar.

Maksimum Örnekleme (Maxpooling): CNN de kullanılan katmanlardan biri de maksimum örnekleme katmanıdır. Bu katmanın kullanılmasının sebebi evrişim uyguladığımız katmanı anlamlı bir şekilde küçültmek ve hesaplamaları hızlandırmaktır. Evrişim sonucu hesaplanan özellik haritaları tekrar bir evrişim işlemine tabi tutularak yeni filtreler ve yeni özellik haritaları elde edilir. İşte bu özellik haritalarında belirli bir alan seçilerek bu alan altındaki en büyük değer kullanılmak için alınır. Bunun amacı özellik haritasını anlamlı bir şekilde küçültmek ve daha az sayı kullanarak eğitin/test işini hızlandırmaktır. Maksimum örnekleme işlemi özellik haritasından bir özet çıkarmak gibi düşünülebilir. Fakat yapılan son çalışmalarda maksimum örnekleme katmanı kullanmak yerine bulunan filtrelerin kaydırma (stride) miktarını 2 veya daha fazla yapmanın daha iyi sonuçlar verdiği şeklinde bir eğilim vardır. Maksimum örnekleme şu şekilde yapılır.



Şekil 2.15. Maksimum Örnekleme İşlemi

Şekil 2.15'da görüldüğü gibi ilgili alanın en büyük değeri alınarak yeni bir matris oluşturulmuş. Ve 4×4 boyutunda bir matriksden tam dört kat küçük olan bir matris elde edilmiştir. Daha küçük matrislerle uğraşmak daha hızlı işlem süresi demektir.

İyileştirici (Optimizer): İyileştirici kavramı bulunan kayıp değerinin nasıl minimize edileceğinin metodudur. Byrd ve ark. (2012)'na göre kullandığımız networkteki ağırlıkları güncellemek için bir yol izlemelidir ki kullandığımız filtreler istediğimiz nesneleri bulabilsin gerekli özellikleri çıkarabilsin ve yaptığımız tahminler doğru çıksın. En çok kullanılan optimizasyon metotları şunlardır:

a) Kademeli İniş Metodu (Gradient Descent):

Ruder (2016) yaptığı çalışmada kademeli iniş, bir fonksiyonun minimumunu bulmak için birinci mertebeden türevlerini alarak yapılan optimizasyon algoritmasını kullanır. Eğim inişini kullanarak yerel minimum bir nokta bulmak için, geçerli noktadaki fonksiyonun gradyanının negatifiyle orantılı adımlar atılır. Kademeli iniş metodu aynı zamanda en dik iniş olarak ta bilinir. Kademeli iniş metodu, makine öğrenmesi alanında popüler bir yöntemdir, çünkü makine öğrenmesinin amaçlarından biri, eğitim verisi için, en yüksek doğruluğu bulmak ve hata oranını en aza indirmektir. Kademeli iniş metodu "hata fonksiyonunu "en aza indirgeyerek minimum hatayı değerini bulmak için kullanılır.

b) Ağırlık Düşürme Katsayısı (Momentum):

Rastgele gradyen inişi (stokastik gradyen descent) yöntemi ile birlikte kullanılan çok genel bir teknik ağırlık düşürme metodudur. Aramaya rehberlik etmek için yalnızca geçerli adımın gradyenini kullanmak yerine, gidilecek yönü belirlemek için geçmiş adımların gradyeni de hesaba katılır.

c) RMSprop:

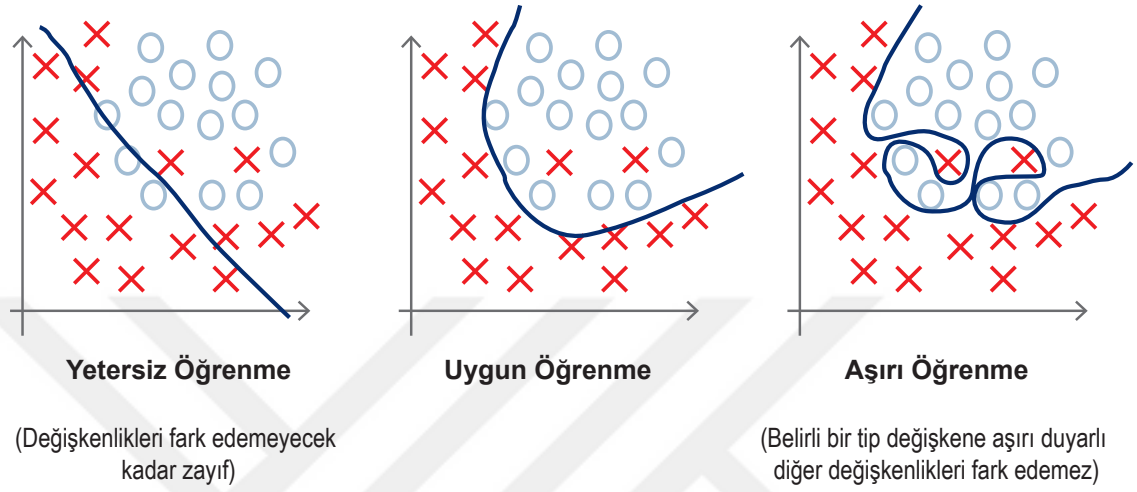
RMSprop momentumla indirgeyen iniş algoritmasına benzer. RMSprop salınımları dikey yönde kısıtlar. Bu nedenle, öğrenme oranımızı artırılabilir ve algoritma yatay doğrultuda daha hızlı yaklaşırken daha büyük adımlar atabilir. Netice olarak RMSprop ve gradyan inişi arasındaki fark, gradyanların nasıl hesaplandığına bağlıdır.

d) Adam:

Adam ilk olarak Kingma ve Ba (2014) yılında yayınlandı. Adam, RMSprop ve rastgele gradyen inişi yöntemlerinin momentum ile bir kombinasyonu olarak görülebilir. Adam RMSprop gibi öğrenme oranını ölçeklendirmek için karesel gradyenleri kullanır ve momentum ile rastgele gradyen inişi gibi gradyen yerine gradyenin hareketli ortalamasını kullanarak momentumdan yararlanıp sonuçları bulur.

CNN Eğitim Aşaması: Yapay bir sinir ağını eğitirken en önemli şeylerden birisi de eğitim kümesidir. Glorot ve Bengio (2010)'nun yaptığı çalışmada CNN' nin eğitim aşamasının nasıl yapılacağından ve zorluklarından bahsetmektedir. Özellikle gözetimli öğrenmede eğitim kümesinin kalitesi çok önemlidir. Bu kalite iki özelliğe bağlıdır bunlar çeşitlilik ve miktardır. Eğer bu özelliklerin yeteri miktarda ve çeşitlilikte olması sağlanamaz ise kullandığımız mimari ne olursa olsun başarıya ulaşılamaz. Sebebi yapay zekânın yeteri derecede eğitilememesidir. Ya çok kısıtlı bir veri biçimine duyarlı olur veya tüm verilere duyarsız olur. Bir veri seti oluşturulurken bir kısmı doğrulama seti olarak ayrılır. Bu mimarimizin ne kadar doğru olarak çalıştığını gözlemlemek için ayrılmış bir veri kümesidir.

Eğitim ve doğrulama adlı eğitim kümeleri etiketlenmiş kümelerdir. Mimarinin katmanlarını içeren CNN yapısını oluşturur. Eğitim seti bu mimariye öğrenmesi için gönderilir ve doğrulama seti ile kıyas edilerek mimarinin ne kadar başarılı olduğu anlaşılır.



Şekil 2.16. Aşırı öğrenme, Normal öğrenme ve Yetersiz öğrenme[12]

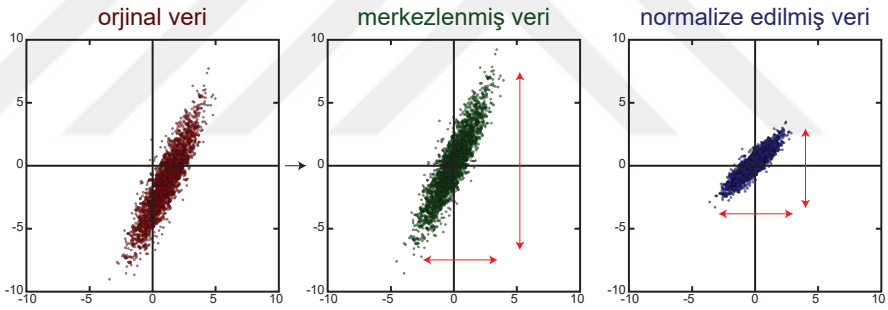
Hem eğitim hem de doğrulama kaybı düşüyorsa mimarinin iyi ve öğreniyor olduğu anlaşılır. Akabinde bu mimarinin test edilmesi gerekir bunun içinde test seti olarak adlandırılan etiketlenmemiş verilerden oluşan bir veri seti kullanılır. Test veri seti mimarinin işini yerine getirip getirmediğinin anlaşılması içindir. Başarısı hiç görmediği bir veriyi doğru bir şekilde etiketleyebilmesi ile anlaşılır. Başarısızlık durumunda veri setimiz ve/veya mimarimiz uygun olmayabilir. Bu gibi durumlarda eğitim setimizde hata değeri hızlı bir şekilde düşerken doğrulama setimizde hata değerimiz yükselebilir. Bu bir şeylerin yanlış gittiğinin göstergesidir. Birkaç ağaç türünün birbirine çok benzeyen yapraklarından 3-5 tane olduğu ve sınıflamanın doğru şekilde yapıldığının sanıldığı fakat yapay sinir ağının bu yapraklardan başka resimler gördüğünde tanıyamadığı gibi bir durumdur. Test aşamasını ise gerçek bir sınav gibi düşünülebilir. Hawkins (2004)'de yaptığı çalışmada belirtildiği gibi eğitim setinden hesaplanan hatanın düşük, doğrulama setinden hesaplanın hatanın ise

[12]https://tinyurl.com/y2ljr34t, Erişim Tarihi: 16-06-2019, Saat: 17:20

yüksek olduğu özel bir hali gösterir. Buna aşırı öğrenme (overfitting) denir. Şekil 2.16'da görüldüğü gibi aşırı öğrenme aslında sadece bir veri tipini çok iyi öğrenip bu veri tipine ait olan başka tipleri bilemediği bir durum olarak tarif edilebilir. Bu da istediğimiz bir durum değildir. Aynı Şekil 2.16'da görüldüğü gibi az öğrenme (underfitting) denilen bir kavram da vardır. Bu durum ise kümeleri birbirine karıştırıp uygun bir ayrımın yapılamadığı yani öğrenmenin olamadığı bir durumu gösterir. Eğitim ve doğrulama sürecinde hatalar hep yükselerek seyreder. Şekil 2.16'de görüldüğü gibi aşırı öğrenmede noktasal verilere grafik çok fazla uyum sağladığı için tek bir veri tipine aşırı duyarlı olurken diğer veri tipleri için gerekli ayırmayı yapamamaktadır. Yetersiz öğrenmede ise veri tiplerinin hassas ayırmı yapılamamaktadır. Olması istenilen öğrenme grafiği Şekil 2.16'da ortadaki gibidir. Düzenlileştirme ('Girosi ve ark. (1995)' Regularization), seyreltme ('Hinton ve ark. (2016)' Dropout), yığın normalleştirme ('Ioffe ve Szegedy (2015)' Batch Normalization) ve mimaride yapılacak değişikler gibi yöntemleri kullanarak aşırı öğrenme ve yetersiz öğrenme gibi problemler çözülmeye çalışılır.

Seyreltme (Dropout): Hinton ve ark. (2016) yaptığı çalışmada belirttiği üzere seyreltme tam bağlantılı yapay sinir ağların da aşırı öğrenme durumunun engellenmesi için tam bağlantılı yapay sinir ağlarında ki bazı hücrelerin belirli bir oranda kapatılması demektir. Sistemde yetersiz öğrenme var ise seyreltme kullanmak yetersiz öğrenmeyi arttırır. Normal olarak bu düzenlileştirme (reguralization) metodu sadece aşırı öğrenme riski olan yerlerde kullanılır. Bilindiği üzere aşırı öğrenme ağının çok büyük yani çok katmanlı, eğitimin çok uzun sürdüğü veya yetersiz miktarda veri olduğu zamanlarda meydana gelir. Seyreltme ancak CNN katmanlarının sonunda genellikle kullanılan tam bağlantılı yapay sinir katmanlarına uygulanabilir. Dikkat edilmesi gereken nokta CNN katmalarına seyreltme uygulanmaz, sadece tam bağlı yapay sinir ağı katmanında seyreltme işlemi uygulanabilir. CNN katmanlarına yığın normalleştirme (bacth normalizasyon) uygulanabilir. Yığın normalleştirme sistemimizi düzenlileştirirken aynı zamanda daha kararlı hale de getirir.

Yığın Normalleştirme (Batch Normalization): Yığın normalleştirme evrişimsel yapay sinir ağlarında düzenlileştirme için kullanılan bir metottur. Ioffe ve Szegedy (2015)'e göre bu metot aynı zamanda evrişimsel yapay sinir ağlarını kaybolan eğime (vanishing gradiente) daha dayanıklı hale getirirken daha iyi bir performans ve daha az bir eğitim zamanı elde etmemizi sağlar. Hatta aşırı öğrenmeden endişelenmediğimiz durumlarda bile yığın normalleştirme uygulamak iyi bir fikirdir. Son zamanlarda seyreltmeye göre avantajlarından dolayı modern mimarilerde daha fazla kullanılmaktadır. Bunun en büyük sebebi yığın normalleştirmenin seyreltmeye göre düzenlileştirme işlemini daha iyi başarmasıdır. VGG16 gibi mimari modellerin son katmanlarını oluşturan tam bağlantılı yapay sinir ağlarında seyreltme metoduna güvenilse dahi daha yeni modellerde tam bağlantılı katmanlar genelde örneklerin ortalaması (average pooling) yöntemi ile değiştirilerek modelin büyüklüğü azaltılırken performansta iyileştirme sağlanmaktadır.



Şekil 2.17. Normalizasyon[13]

Yukarıdaki Şekil 2.17'de görüldüğü gibi dağınık veriler merkezlenmiş ve genliği düşürülmüştür. Böylece daha düzenli ve küçük sayılarla uğraşılmış olur. Netice olarak evrişim katmanlarında seyreltme kullanılmamalıdır. Sadece tam bağlantılı katmanlarda kullanılmalıdır. Eğer evrişim katmanında kullanılırsa hiç kullanılmamasına göre daha kötü bir sonuç elde edilir.

[13]https://tinyurl.com/y4gmmb4m, Erişim Tarihi: 18-05-2019, Saat: 13:45

Geriye Yayılım (Back Propagation): Rumelhart ve ark. (1995)'de yaptığı çalışmada belirttiği üzere geriye yayılım metodu ile bulduğumuz toplam hatayı mümkün olabildiğince küçültmek amaçlanır. Bunun için her değişkenin toplam hataya etkisini bulunarak, toplam hatadan istenilen değişkene ulaşana kadar olan bütün diğer değişkenleri geriye doğru güncellemektir. Bu değişkenler birbirine oranlanarak, bir başka değişle birbirlerine göre kısmi türevleri alarak yapılır.

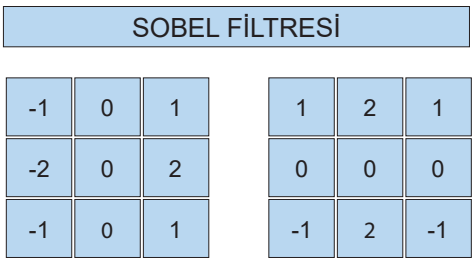
Maliyet ve Hata Fonksiyonları: "Minimize etmek veya maksimize etmek istenilen fonksiyona nesnel fonksiyon veya kriter adı verilir. Minimize ederken, aynı zamanda maliyet fonksiyonu, veya hata fonksiyonu olarak da adlandırılabilir ve bu terimler eşanlamlıdır. Maliyet fonksiyonu daha çok optimizasyon probleminde kullanılırken hata fonksiyonları daha çok parametre tahmininde kullanılır.

Hata fonksiyonu tek bir deney örneği için geçerlidir; maliyet fonksiyonu ise tüm deney seti için geçerlidir. Bu nedenle, bir hata fonksiyonu, maliyet fonksiyonunun bir parçasıdır.

Kayıp fonksiyonu genellikle bir veri noktası, tahmin ve etiket üzerinde tanımlanmış bir fonksiyondur ve cezayı ölçer. Maliyet fonksiyonu genellikle daha geneldir. Eğitim setindeki kayıp fonksiyonların toplamı olarak nitelendirilebilir.

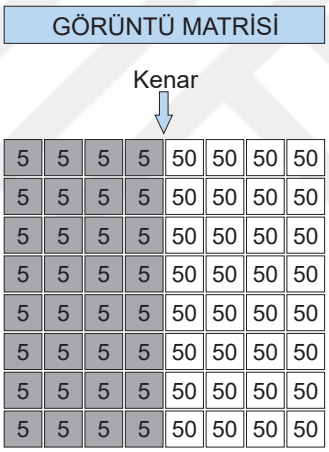
2.5. Derin Öğrenme

Filtreler birer matristir ve bu matrisleri görüntü matrisini manipüle etmek için kullanırız. Görüntünün üzerinden başlayarak filtrenin ve görüntü matrisinin sayılarını kullanarak filtreyi görüntü matrisinin üzerinde kaydırmaya başlarız ve amacımız doğrultusunda ortalama alma, evrişim gibi bir sıra işlemler uygularız. Böylece görüntüdeki sayıları manipüle ederek görüntüyü istediğimiz bir şekle getirmeye veya görüntüdeki istediğimiz bir özelliği bulmayı amaçlarız. Bir görüntüdeki kenarları bulmak için çok bilinen filtrelerden biri Sobel filtresidir .Vincent ve Folorunso (2009)'nun yaptığı çalışmada Sobel filtresinin uygulaması anlatılmaktadır. Ve bu uygulamadan elde edilen kazanımla derin öğrenme konusunu daha iyi anlanabilir.



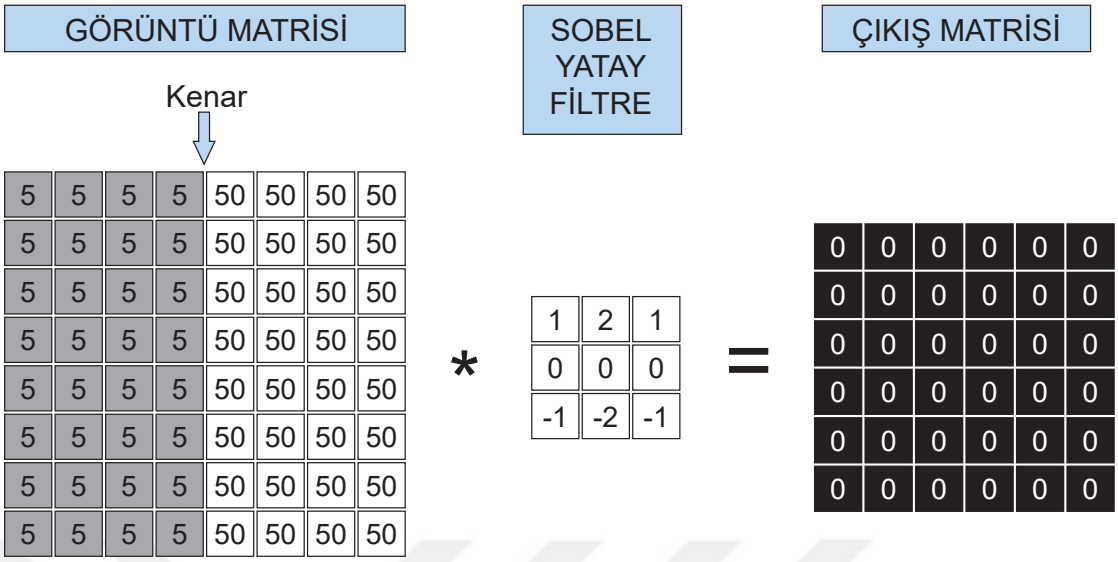
Şekil 2.18. Dikey ve Yatay Sobel filtreleri

Dikkat edilirse Şekil 2.18'da ikinci filtre birinci filtrenin saat yönünün tersine 90 derece döndürülmüş halidir. Soldaki filtre yatay yöndeki kenarları sağdaki filtrede düşey yöndeki kenarları bulmamızı sağlar. Ve görüntü matrisine evrişim işlemi yapılarak uygulanır.



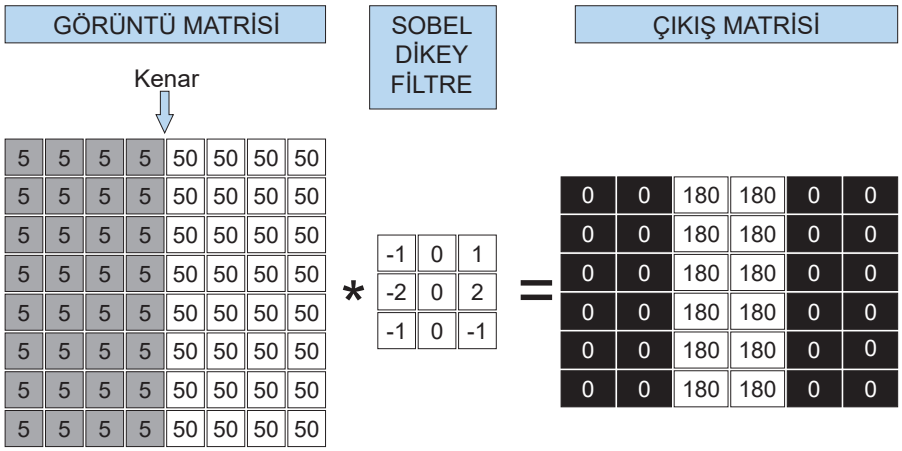
Şekil 2.19. 8×8 Görüntü matrisi

Şekil 2.19'de görüldüğü gibi 5 olan değer yine Şekil 2.19'deki "Kenar" işareti ile belirtilen noktadan sonra 50 değerini almaktadır. Bu tek kanallı 0 ila 255 arası değerlerin değiştiği gerçek bir resimde neredeyse siyah olan alandan daha beyaza yakın bir alana geçişi gösterir. Bu açıkça dikine kenar olan bir yeri göstermektedir.



Şekil 2.20. Yatay Sobel filtre uygulaması

Şekil 2.20'de dikine kenar olan bir görüntü matrisine yatay kenarları bulan Sobel filtresi uygulandığında çıkış matrisinin bütün değerlerinin sıfır olduğu görülmektedir. Bu görüntü matrisinde herhangi bir yatay kenar yok demektir.



Şekil 2.21. Dikey Sobel filtre uygulaması

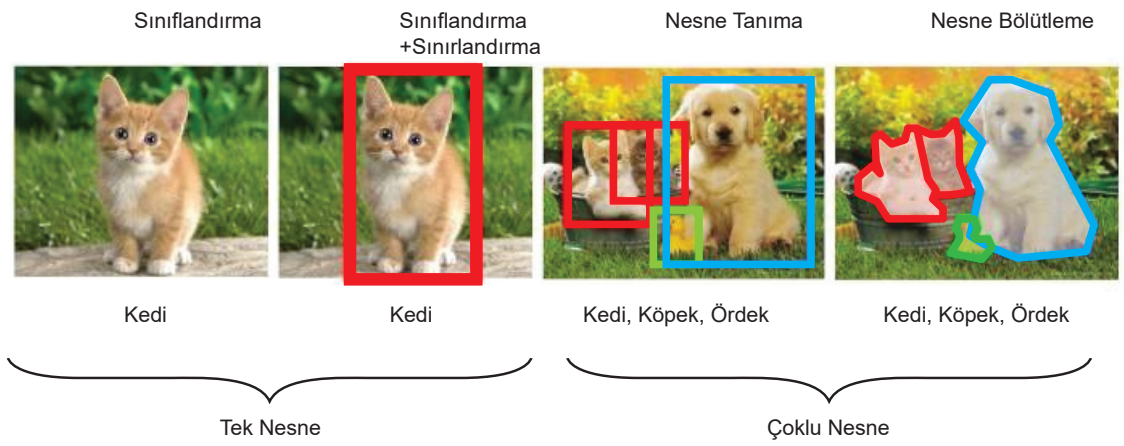
Şekil 2.21'de görüldüğü gibi uygun olan Sobel filtresi uygulandığında kenar olan yerdeki değerler açıkça göstermektedir ki görüntü matrisinde dikine bir kenar vardır.

Bu uygulama gösterir ki eğer Sobel gibi bir filtre ile kenarlar bulunabiliyorsa. Başka türlü filtrelerle yuvarlaklıkları, köşeleri ve sadece o şekle özgü karmaşık figürler de yine filtreler yardımıyla bulunabilir. Bu noktadan hareketle CNN ve derin öğrenme yapıları kullanılarak bir sınıfı tarif eden ve o sınıfa özel filtreler bulunarak şekiller sınıflandırılabilir. CNN'de ebatları ve miktarları önceden belirlenen filtreler; yapay zekânın kendini sürekli yenileyerek ağırlık bulma özelliği yenilenir. Böylece sınıflandırılmak istenen şekli tarif eden çok özel filtreler bulunur. İşte derin öğrenmenin bütün püf noktası burasıdır. Evrişimsel yapay sinir ağında evrişim katmanlarının hepsinde önceden rastgele olan filtrelerin değerleri ağırlık güncelleme metoduyla sürekli güncellenerek şekli tarif eden filtreler bulunur. O şeklin birçok türüne uygulayarak ilgili sınıfı belirleyen filtreler elde edilmiş olur. Derin öğrenme bilindiği üzere çok katmanlı mimarilerin kullanıldığı bir yapıdır. Katmanlar bir öncekinin aynısı olabilirken, gerekli aşamalarda farklı katmanlar da kullanılır. Derin öğrenmede kullanılan çok katmanlı mimarilerde her katmanda ilgili özellikler çıkarılarak daha derine inilir. Kurulan mimarinin yeteri kadar özellik çıkardığı noktaya kadar mimari derinleştirilir. Bu özellikler birinci katmanda daha genel iken katman sayısı arttıkça giriş katmanındaki veriye ait daha kompleks özelliklerin çıkarıldığı özelliklere ulaşılır. Bağlantıdaki örnekte [14] MNIST veri setine ait bir videoda öğrenme sırası (epoch) dediğimiz ağırlıkların güncellenmesi aşamalarında bulunan filtrelerin nasıl değiştiği görülmektedir. Öğrenme miktarı arttıkça şekli tarif eden filtrelerin daha belirgenleştiği görülmektedir. Aynı sobel örneğinde olduğu gibi bulunan bu filtreler ilgili sayıların üzerinde geldiğinde adeta tepkimeye girip yüksek sonuçlar vermektedirler. Bu yüksek sonuçlar orada aranılan verinin olduğunu gösterir ve istenilen sınıf veya özellik bulunmuş olur.

[14] https://tinyurl.com/yxgx8wvz, Erişim Tarihi: 06-06-2019, Saat: 18:30

3. MATERYAL VE YÖNTEM

Nesne tespit işlemi bir pencerenin bir matrisin görüntü yüzeyince kaydırılarak, üzerinde bulunduğu alanda ilgili bir sınıfa dair işaret olup olmadığının araştırılmasıdır. Pencere denilen kavram ise herhangi bir ebatta belirlenmiş matrislerdir ve filtre olarak isimlendirilir. Kabaca bunu bir görüntüde çok ölçekte yaparak yani kayan pencerenin ebatlarını değiştirerek bir nesnedeki sınıflar bulunabilir. Fakat böyle bir yaklaşım uygun değildir çünkü hem çok yavaştır hem de güvenilir olmayabilir. Çünkü görüntüdeki nesneler birçok boyutta çok farklı yerlerde üst üste binmiş olabileceğinden tespit için çok fazla sayıda ve boyutta pencereye ihtiyaç olur. Bunun yerine başarılı bir tespit ebatları ve özellikleri çok iyi ayarlanmış bir filtre yardımıyla görüntüdeki sınıfların doğru özetlerini, imajın ilgili alanlarından çıkarılması şeklinde olmalıdır. Bu şu demektir: öyle filtreler ile özellik haritaları bulunmalıdır ki bu filtreler görüntüyü çok anlamlı bir şekilde küçültürken özellik haritaları da en özet şekilde cisimleri tarif edebilmeli ve en az filtre miktarı ile büyük ve küçük cisimleri bulabilmelidir.



Şekil 3.1. Nesne sınıflandırma ve tanıma konu başlıkları[15]

[15]https://tinyurl.com/y3l2ddop, Erişim Tarihi: 06-03-2019, Saat: 18:30

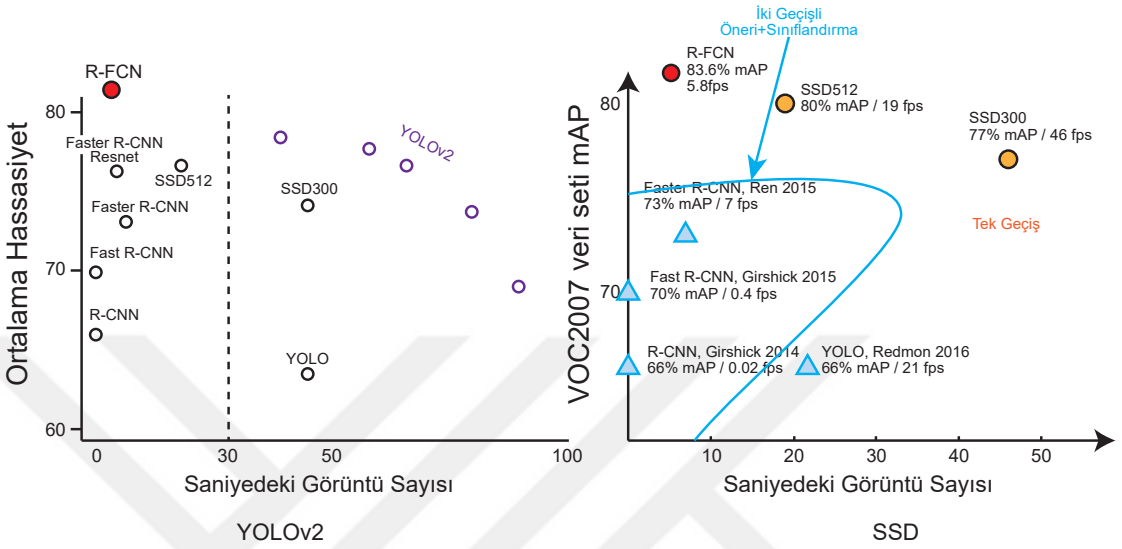
Şekil 3.1'de görüldüğü üzere nesne tespiti dört ana kısımdan oluşmaktadır. En soldaki fotoğrafta sınıflandırma işlemi görülmektedir ve sadece fotoğrafta ne olduğu ile ilgilenilir nerede olduğu ile ilgilenilmez ve fotoğrafı ilgili sınıf adı ile etiketler.

İkinci resimde ise bir nesnenin nerede olduğu ve ne olduğu bulunmaya çalışılır. Görüldüğü üzere Şekil 3.1'in ikinci fotoğrafında kedi bir kutu içerisine alınmış ve fotoğraf kedi olarak etiketlendirilmiştir. Bu tek bir sınıf için yapılan işlemdir. Üçüncü fotoğraf da ise birçok farklı nesnenin ne olduğu ve nerede olduğu ile ilgilenilir ve hepsinin bir sınır kutusu çizilmeye çalışılır. Dördüncü fotoğrafta nesnenin çevresine bir sınırlayıcı kutu çizilmektense kontur çizgileri çizilmeye çalışılır. Buna bölütleme (segmantasyon) işlemi denir. Bu tezin konusu üçüncü resim ile örtüşmektedir. Bu tezde derin öğrenme metotlarından biri olan tek geçişte birçok nesnenin tespitinin yapıldığı (Single Shot MultiBox Detector) metodu ile görüntüdeki hem nesnelerin ne olduğunu hem de nerede olduğunu saptamaya çalışılarak gerçek zamanlı olarak çalışan mekanik bir sistem tasarımı gerçekleştirilmiştir.

3.1. Güncel Nesne Tespit Yöntemleri

Burada bahsedilecek nesne tespit yöntemlerinin tamamının amacı Şekil 3.1'in üçüncü resminde bir görüntüdeki nesneye sınırlandırma kutusu çizerek gerçekleştirmektir. Nesnenin türünü ve yerini, o nesneye ait olan sınırlayıcı kutusunun herhangi bir yerine yazarak gösterir. Nesne tespit ediciler, tek geçişte bütün görevleri gerçekleştiren ve iki geçişte bütün görevleri gerçekleştiren olmak üzere temelde iki kısma ayırılır. İki geçişli algoritmalar küçük nesneleri bulmakta tek geçişli algoritmalara göre daha iyidirler. Diğer bir yaklaşım ise bölgesel alan önerme aşaması olan birinci aşamayı elemine ederek direkt olarak tespit aşamasını, özellik haritasındaki nesnenin olabileceği yoğun bölgelerde tespit etmeye çalışmaktır. Bu tür algoritmalara tek geçişli algoritmalar denir. Bu algoritmalar hızlı ve basit olmasına rağmen sınıf bulmada güvenilirlikte ve küçük cisimlerin tespitinde bazı handikaplara sahiptir. Fakat bu dezavantajlarına rağmen hızlıdırlar ve nesne tespiti konusunda yine de yeterli güvene sahiptirler. İki geçiştekilere örnek olarak R-CNN ailesini

verilebilir. Bunlar R-CNN (Dai ve ark. (2016), Fast-RCNN (Girshick (2015) ve Faster-RCNN (Ren ve ark. (2015) dir. Tek geçiştekilere örnek olarak da YOLO (Redmon ve ark. (2016)) ve SSD (Liu ve ark. (2016) mimarilerini verebiliriz.



Şekil 3.2. R-CNN,YOLO ve SSD karşılaştırılması[16]

Şekil 3.2'de görüldüğü gibi tek geçişli algoritmalar iki geçişli algoritmalara göre oldukça iyi konumdadırlar. Her ne kadar R-CNN ailesi nesneleri çok daha iyi tespit edebilse de hız olarak çok yavaş kalmaktadır. Bu sebeple gerçek zamanlı nesne tespit için YOLO ve SSD kullanılmaktadır.

3.1.1. R-CNN

R-CNN ailesinin hepsi bölge bazlıdır ve tespit prosesi iki aşamalıdır bunlar:

1. Model, belirlenmiş kadar alakalı bölgeyi yine belirlenmiş bölgesel bazlı öneriler sunan bir network ile işlemlere tabi tutar. Önerilen bölgeler şekli çevreleyen sınır kutuları teorik olarak sınırsız olabilir. Bunu belirli bir sayıda tutarak sadece şekli tespit edebilecek kadar sınır kutusu elde edilir.

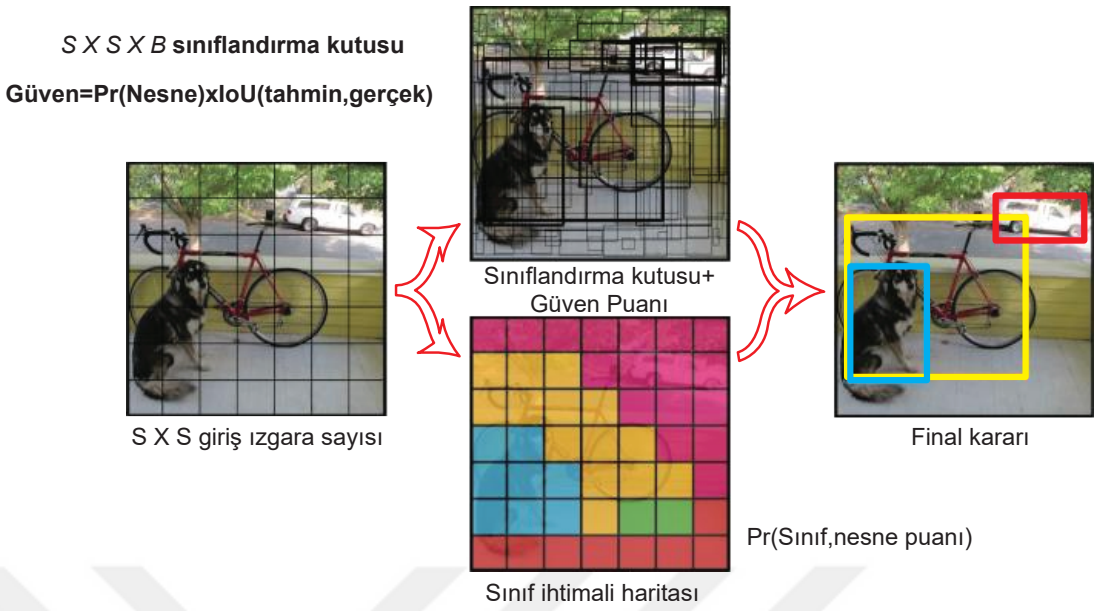
[16]https://tinyurl.com/y3es5e9s, Erişim Tarihi: 06-09-2018, Saat: 14:30

2. Sınıflandırıcı sadece şekli çevreleyen sınır kutusu adaylarını, aday bölgede nesnenin varlığını ve güvenilirliğini hesaplar. İki geçişte nesnenin tespitini yapan R-CNN gibi algoritmalar nesneleri daha iyi tespit edebilir. Fakat çok yavaş olduklarından gerçek zamanlı uygulamalar için elverişli değildirler.

3.1.2. YOLO

YOLO (You only look once), obje tespit metodu Redmod ve Arkadaşları tarafından 2016 yılında önerilmiştir. Gerçek zamanlı obje tespiti denemelerinin ilklerindendir. YOLO bölgesel öneri adımını uygulamadan sadece kısıtlı sayıda sınırlayıcı kutu tahminlerini yaparak çok hızlı bir şekilde sonuç alır. Daha öncede bahsedildiği gibi hızını tek geçişte yer ve sınıf tespit edebilmesine borçludur. İlk YOLO mimarisi sonrasında çeşitli versiyonlarla geliştirilerek YOLO hem güvenilirlik hem de hız konusunda oldukça ilerlemiştir. İlk YOLO-V1'in çıkmasından sonra YOLO-V2, YOLO-9000, YOLO-V3 olmak üzere çeşitli farklı versiyonları vardır.

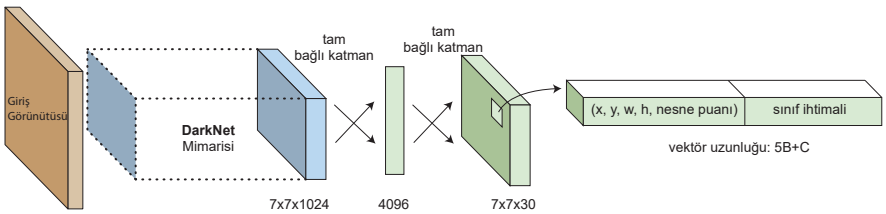
CNN ağı görüntü sınıflandırması için eğitilir. Orijinal YOLO'da nesne sınıflandırılması için ImageNet kullanılmıştır. İş akışı şu şekildedir: Görüntü hücrelere bölünür. Eğer objenin merkezi herhangi bir hücrenin içine düşerse, bu hücre objenin tespitinden sorumlu hücre olarak belirlenir. Her hücre sınır kutuyu tahmin etmekten, güven puanın belirlenmesinden ve sınırlayıcı kutuda bir nesnenin varlığına bağlı olarak sınıflandırılmış nesne sınıfı olasılığını bulmaktan sorumludur. Sınırlayıcı kutunun koordinatları 4 adet veri kümesi ile belirlenir. Bunlar merkezin x, y koordinatı, genişlik ve yükseklikliktir. Ayrıca görüntü genişliği ve yüksekliği 0 ile 1 arasına normalleştirilir. Güven puanı ile belirtilen değer incelenen hücrenin içerisinde herhangi bir nesnenin olup olmadığını gösterir. Eğer hücrenin içerisinde herhangi bir nesne var ise bütün nesnelere ait olabilirlik puanı hesaplanır eğer yok ise hiç bir hesaplama yapılmaz böylece işlem hızlandırılmış olur. Bu aşamada, model, sınırlayıcı kutuların sayısına bakılmaksızın, hücre başına yalnızca bir sınıf olasılık seti belirler. Son kat evrişim katmanının tensör haline çevrilmiş şeklidir.



Şekil 3.3. Redmon ve ark. (2016)

Şekil 3.3'da görüldüğü gibi orijinal resim 5×5'lik ızgaralara bölünmüş güvenilirlik ve sınıf puanları hesaplanarak ilgili sınıfların çevresine sınırlayıcı kutuları çizilmiştir.

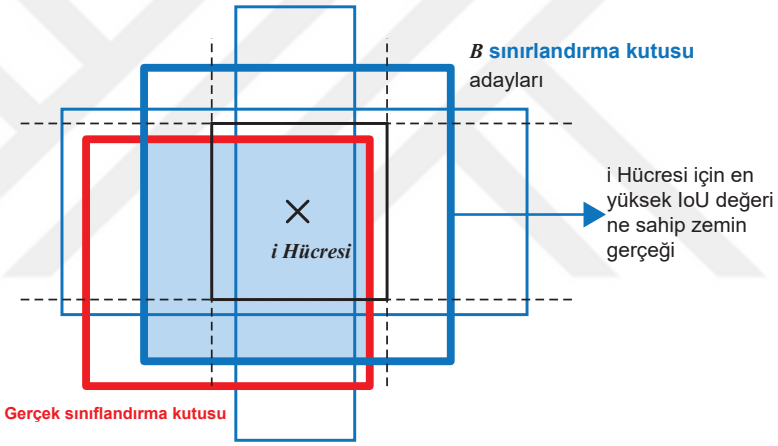
Ağ Mimarisi: Temel model GoogleNet'e benzer fakat başlangıç modülünün 1x1 ve 3×3 evrişim katmanları ile değiştirilmiştir. Şeklin son tahmin kısmında, tüm evrişim özellik haritası üzerinde tamamen birbirine bağlı iki katman kullanılarak tahminler yapılır.



Şekil 3.4. YOLO ağ mimarisi[17]

Şekil 3.4'de YOLO mimari yapısı verilmektedir. Giriş katmanı $448 \times 448 \times 3$ tür. Ve son katmanında SSD'den farklı olarak tamamen bağlı yapay sinir ağı yapısı kullanılmıştır.

Hata Fonksiyonu: Hata miktarı tespit edilen nesnenin çevresine çizilecek sınır kutularını denkleştirmek için sınırlama kutusunun bulunması ve sınıfların ne olduğunun anlaşılması olmak üzere iki kısımdan oluşur. Her iki kısımda karesel ortalamaların toplamı hesaplanarak hata değeri bulunur. Sınırlayıcı kutu koordinat tahminlerinden gelen hatayı ne kadar artırmak istediğimizi ve nesne olmayan kutular için güven puanındaki tahmin kaybını ne kadar azaltmak istediğimizi kontrol etmek için iki ölçek parametresi kullanılır. Arka plan sınırlayıcı kutuları nesne içermediğinden hesaplara belirli bir oranın üzerinde katılmaması hesaplamanın hızlı yapılabilmesi için önemlidir.



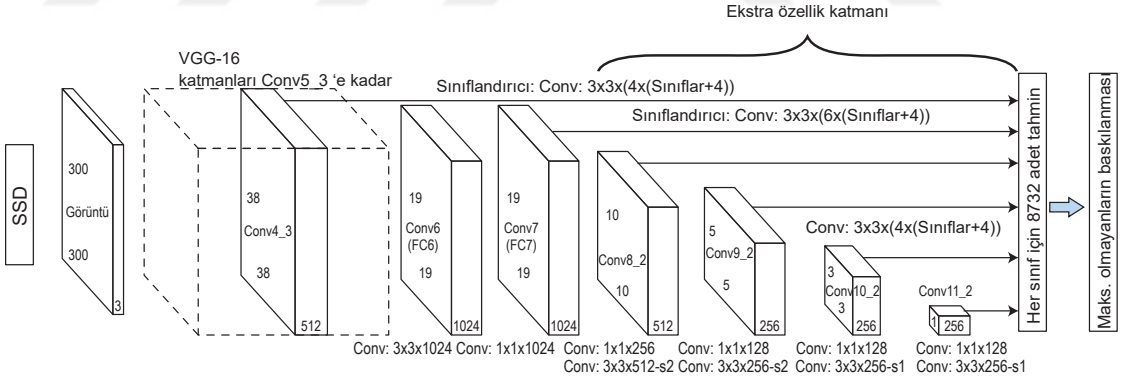
Şekil 3.5. Hata fonksiyonuna göre çizilmiş sınırlandırma kutusu[18]

Şekil 3.5'de hata fonksiyonu, yalnızca kesik çizgilerle gösterilmiş hücrede bir nesne varsa, sınıflandırma hatasını düzenler. Aynı zamanda, sınırlayıcı kutu koordinat hatasını bu öngörülen temel gerçek kutusu için "sorumlu" ise, düzenler. Sonuç olarak bir geçişte tespit yapabilen nesne detektörü olarak, YOLO çok hızlıdır, ancak sınırlı sayıda sınırlayıcı kutu adayı nedeniyle düzensiz şekilli nesneleri veya küçük ebatlı grup nesneleri tanımakta iyi değildir.

[17]https://tinyurl.com/y2auy6nt, Erişim Tarihi: 06-04-2019, Saat: 14:30
[18]https://tinyurl.com/y2auy6nt, Erişim Tarihi: 06-04-2019, Saat: 14:30

3.1.3. SSD

SSD'de aynı YOLO gibi tek geçişte bütün görevleri yerine getirmeyi amaçlar. Aynı YOLO gibi temel bir sınıflandırma mimarisi kullanarak görüntüdeki sınıfları bulmaya çalışır. YOLO 'da ImageNet mimarisi kullanılırken SSD'nin orijinal makalesinde VGG16 kullanılır. Proje orijinal makaleye uygun olarak geliştirilmiştir. Fakat gerçek uygulamalarda herhangi bir zorunluluk yoktur. Sınıflandırma için istenilen mimari hatta sonrası için yani SSD katmanları içinde farklı mimariler kullanılabilir. SSD'de YOLO'nun aksine son katmanlarında herhangi bir tam bağlı katman bulunmamaktadır. Onun yerine sabit öncül kutuların bulunduğu yerlere evrişim fitrelerinden gelen filtreler ve bu filtrelerin sonuçlarının yorumlandığı VGG16 ile paralel çalışan ve orijinal VGG16 ya bazı ek katmanlarının da sonuçlarının eklendiği kesişimlerin birleşimlere oranı (IoU) ve maksimum olmayanların bastırılması (NMS) denilen özel bir yorumlama ve netleştirme algoritmasıyla sonuca ulaşıldığı bir mimari kullanılır.



Şekil 3.6. SSD algoritmasının mimari yapısı. Liu ve ark. (2016)

Şekil 3.6 bize SSD mimarisini gösterir. İlk iki kutu yani görüntü yazan ve sonrasındaki kesikli kutu, bir farkla VGG16 deki orijinal mimarinin aynısıdır. O fark VGG16 da giriş katmanı 256×256 iken SSD'de giriş katmanı 300×300 olarak seçilmiştir. Bunun 512×512

olan versiyonu da vardır. Fakat SSD stratejisini kullanarak ihtiyaca göre istenilen giriş ebatları katmanlar ve mimariler seçilebilir. VGG16 katmanındaki özel bir katman olan $38 \times 38 \times 512$ ebatlarındaki Conv4_3 ve sonrasındaki $19 \times 19 \times 1024$ olan katmanlar SSD ye ait olan SSD tarafından ilave edilmiş katmanlardır. Bu katmanlardan çıkan ok ile gösterilen ek evrişim katmanları da bizi tespit ve NMS yani objenin ne olduğuna ve nerede olduğuna hüküm verilen katmanlara götürür. Ve bunların sonucunda karar verilir. Dikkat edilmesi gereken husus test aşamasında elimizde sadece görüntü vardır. Objenin nerede olduğu ve ne olduğu bilinmemektedir. Bunlar SSD'nin eğitimi sırasında görüntülerden bulunan filtreler ile yine SSD mimari yapısı kullanılarak bulunmaya çalışılır. SSD'nin eğitilmesi ve de veri kümesinin hazırlanması bir hayli zahmetli olduğundan bu zahmeti ellerindeki süper bilgisayar ve iş gücü imkanlarıyla aşan Google benzeri şirketlerin son kullanıcı için hazırladığı bedava eğitim verileri kullanılarak genelde geliştirmeler yapılır. Bu eğitim setlerinde olmayan özel sınıflar için ve/veya değişik bir mimari için geliştirme yapıyorsa bu aşamaların bizzat sahip olunan imkanlarla gerçekleştirilmesi gerekmektedir. Bu hazır veriler sadece belirli sınıflarda ve mimarilerde hazırlandığından kullanılmadan önce kendi mimarimize uygun olup olmadığı bilinmelidir.

Çizelge 3.1. SSD eğitim sistemi özellikleri

Network.	SSD300	SSD512
Eğitim Hızı (görüntü/saniye)	145	65
IoU=0,50 Maks. Tespit=100	21,9	25,7
Eğitim Zamanı (100 epochs)	23,5 saat	42,5 saat
Kullanılan Sistem:	i7-7820X CPU	$4 \times$ 1080 Ti
İşletim Sistemi:	Ubuntu 18.04	LTS Versiyon

Yukarıdaki belirtilen Çizelge 3.1'deki sistemle yapılan eğitim süreleri ve başarı oranları görüldüğü gibidir. Bu sürenin içerisinde eğitim verisinin hazırlanma süresi yoktur. SSD ye gönderilecek bir eğitim verisi hazırlanırken bulunması istenilen sınıfların ne olduğu ve nerede olduğu yardımcı etiketleme programlarıyla etiketlenerek hazırlanır. SSD'nin orijinal referans programı Pascal-Voc veri setiyle hazırlandığı için SSD'ye kendi hazırladığımız data setinin de Pascal-Voc data verisi tarzında hazırlaması önemlidir. Yoksa eğitim yapılamaz ve program hata verir.

SSD adında olduğu gibi görüntüyü çok anlamlı bir şekilde küçültürken özellik haritalarını özet bir şekilde bularak en az filtreyle nesneleri takip edebilen bir algoritmadır.Ve bunu görüntünün içinde bulunan bilgileri çok efektif bir şekilde çıkararak güvenilir bir şekilde yapar. Derin öğrenme yaklaşımı eski yöntemlerde kullanılan coğrafi dönüşüm yöntemlerine göre nesnelerin tespitini çok daha güvenilir bir şekilde yapar. SSD görüntü sınıflandırmasını derin öğrenme tabanlı yaptığından bu konuda çok güvenilir bir yapı sunar ve kayan pencere yöntemini kullanarak görüntüde filtre altında kalan alıcı alanları kısmi arama pencereleri gibi kullanır. Fakat SSD'nin aynı zamanda evrişim ve havuzlama operasyonunun vasıtasıyla da karar verdiğinden sınırlı bir çözüme sahiptir. Çünkü alıcı alanların hedefin dışında kaldığı yerlerde görüntüden zayıf örnekleme bilgileri çıkarır. Fakat çıkarılan çok kompakt özellikler sayesinde bu SSD'nin sınıflandırma performansını etkilemez. Görüntünün sadece bir parçası görülse bile nesneyi tespit edebilir.

Derin evrişimsel yapay sinir ağları sadece nesnenin ne olduğunu görmez aynı zamanda kesin bir şekilde nerde olduğunu da görür. Bunları kullanarak, SSD, nesnenin sınıf puanını ve sınırlayıcı kutuyu temsil eden dört numaradan oluşan bir vektörde birleştirir. Bu çok önemli bir şeydir. Algılama şimdi tanımlanmış şekillerden arındırılmıştır, dolayısıyla çok daha az bir hesaplama ile çok daha doğru yerelleştirmeler elde edilir. SSD kavramının anlaşılması kolaydır, ancak gerçekleştirme birçok ayrıntı ve karar sonucunda mümkün olur. SSD giriş sabit boyutlu bir görüntüdür, örneğin, SSD300 için 300×300 dür. Etkin bir eğitim yapılabilmesi için görüntünün giriş boyutu sabit boyutlu olmak zorundadır. Böylece evrişimsel yapay sinir ağları üzerinde çalıştırılarak gerekli bilgiler çıkarılabilir. SSD'nin katmanlarındaki çıktılar tespitte maksimum olmayan tahminleri (NMS) elendiği bir aşamaya giden tahmin haritasıdır. Bu haritadaki her yer, sınıf ve sınırlayıcı kutu bilgilerini depolar. Elbette birçok yetersiz tespit vardır, bu yüzden basit sezgisel taramalara dayanarak en olası tahminin bir listesini seçmek için IoU denen başka bir işlem kullanılır. Burada eğitimden çıkan ve doğru bir şekilde objenin yerlerini bulabilen filtreler kullanıldığı için bu filtrelerin maksimum sonuç verdiği yerlerle evrişim işleminden gelen öncül kutuların kesişimler ine ve birleşimlerine oranı karşılaştırılarak en büyük değerli olan yer istenilen nesneyi çevreleyen sınır kutu olarak tespit edilir. Burada bir yaklaşım metodu kullanıldığı

için sınır kutuları her zaman objenin tam olarak sınırından geçmeyebilir.

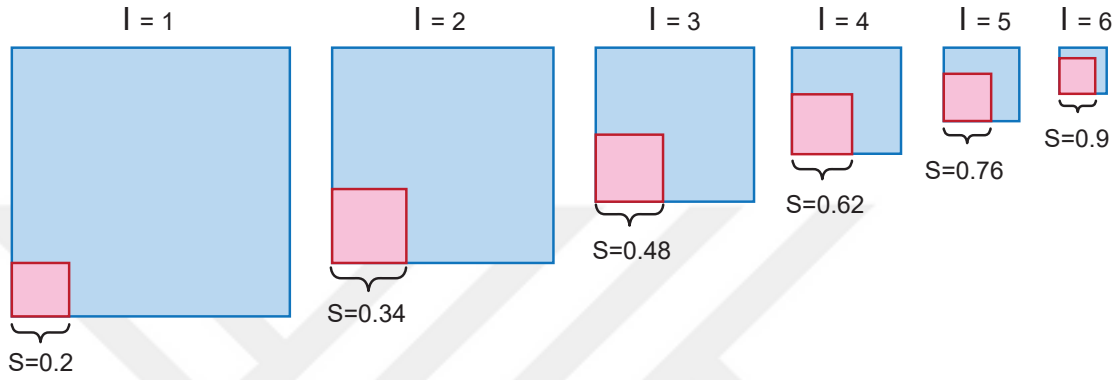
Ağı eğitmek için yer gerçeği dediğimiz daha önceden etiketlenmiş nesnelerin yerlerinin ve sınıflarının işaretlendiği bir yapı kullanılır. Tahmin haritası ile yer gerçeği karşılaştırılır. Yer gerçeği: Test sırasında SSD den bulmasını istediğimiz ve eğitim sırasında manuel olarak yardımcı programlar vasıtasıyla bir görüntüdeki nesnelerin yerlerinin ve türlerinin etiketlenmiş halidir. Evrişimsel yapay sinir ağlarının farklı katmanlarından tahmin haritaları oluşturularak çok ölçekli tespit gerçekleştirilir. Örneğin, SSD300, sırasıyla

$$38 \times 38 \times 4 + 19 \times 19 \times 6 + 10 \times 10 \times 6 + 5 \times 5 \times 6 + 3 \times 3 \times 4 + 1 \times 1 \times 4 = 8732$$

adet 6 tahmin haritası çıkarır. Sahnenin arkasında 8732 adet "yerel tahmin" vardır. Her tahminin girişi esas resim üzerindeki algılayıcı alanlardır. Çıkışı da o algılayıcı alana ait tahminlerdir. Bunlar öncül kutular için gerekli olan sınırlayıcı kutu tahminleridir ve nesnelerin belirli bir ölçekte ortalama şeklini görür. Her bir tahminin yerel gerçekliğini seçmek için o şekle ait belirlenen öncül kutular kullanılır. Şekilleri bulmak için tahmin haritasında her yere öncül kutular koyulur ve yer gerçeği ile öncül kutular arasında sürekli olarak IoU 'yu hesaplanır. Bu hesaplamalar çok miktarda olduğundan bazı değerleri elemine etmek için IoU değerlerine bir eşik değeri uygulanır ve bu eşik değerinden büyük olan IoU değerleri yer gerçeği olarak kabul edilir. Eğer tahminde sınıflara ait herhangi bir değer yok ise arka plan olarak kabul edilerek yerelleştirme hatası hesaplanmaz. Temel olarak öncül kutular ile herhangi bir yer gerçeği nesnesi arasında önemli bir çakışma varsa yer gerçeği bu konumda kullanılır. Yer gerçeğini yerelleştirme için kullanıldığı gibi sınıf hatasını bulmak için de doğrulan kullanılabilir. Yer gerçeğini sınır kutular (bounding box) ile öncül kutular (prior box) arasındaki tahmin edilen yerin yani lokalizasyon hatasını hesaplamak için kullanılır. Öncül kutuların büyüklüğü detektörün ne kadar "yerel" olduğuna karar verir. Daha küçük olan öncül kutuların, detektörün IoU dan dolayı daha yerel davranmasını sağlar. Farklı ölçeklerde tahminler için farklı boyutlar kullanmak iyi bir uygulamadır. Örneğin, SSD300, 5 adet hedef görünüş oranı kullanır bunlar 1,2,3,$\frac{1}{2}$,$\frac{1}{3}$.

SSD her farklı ölçek için ekstra olarak varsayılan kutu (default box) ekler:

$$\text{Ölçek}=\sqrt{\text{Ölçek} * \text{Bir sonraki seviye ölçek}}$$



Şekil 3.7. SSD'nin her katmanda ölçeklendirilme biçimi[19]

Yukarıdaki Şekil 3.7'de görüldüğü gibi her katmandan sonra özellik haritasındaki küçülmeye bağlı olarak nesnelerin arandığı ölçekte küçülmektedir. Bu sayede gerçek görüntüde değişik ebatlardaki nesnelerin aranabilme durumu çok daha küçük haritalarda gerçekleştirilmiş olur. Bağlantıda[20] ilgili kod kısmında her katmanda ölçek boyutunun nasıl hesaplanması gerektiği görülebilir. Uygulamada tek bir katman için SSD değişik ölçeklerde ve görünüm oranında (aspect ratio) birçok öncül kutuyu objenin yerini bulmak için kullanır. Farklı şekillerde nesneleri algılamak için farklı öncül kutular oluşturur. SSD512'de 7 tane tahmin katmanı için 4, 6, 6, 6, 6, 4, 4 tip değişik öncül kutu üretir ve bunların görünüm oranı $\frac{1}{3}, \frac{1}{2}, \frac{1}{1}, \frac{2}{1}, \frac{3}{1}$'dir. Bunlar aynı algılayıcı alan için karar verseler de farklı davranırlar çünkü bunlar değişik parametreleri "evrişimsel filtreler" ve farklı öncül kutuların getirdiği farklı yerel gerçeklikleri hesaplamak için kullanırlar.

[19] https://tinyurl.com/y2auy6nt, Erişim Tarihi: 06-04-2019, Saat: 14:30

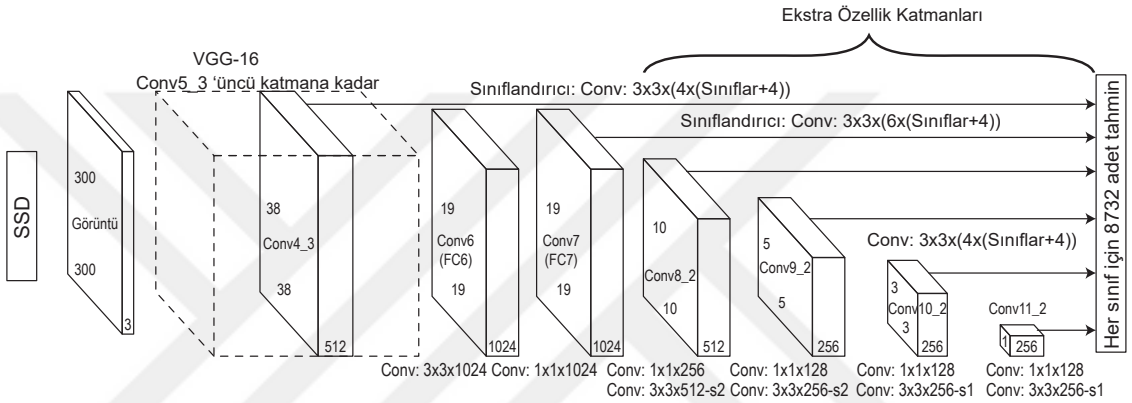
[20] https://tinyurl.com/y457sy2a, Erişim Tarihi: 11-06-2019, Saat: 11:00

Sert Negatif Madencilik: Öncül kutular, herhangi bir sınıfla eşleşen hiçbir nesnenin bulunamadığı arka planlar da dahil olmak üzere temel gerçek tahminleri oluşturmak için basit ve mesafeye dayalı sezgisel tarama kullanır. Bununla birlikte, ön plan örnekleri ve arka plan örnekleri arasında bir dengesizlik olabilir, çünkü arka plan genelde daha çoktur. Sonuç olarak, SSD'nin gerçek bulunmasını istediğimiz nesnelerinin az olmamasından dolayı birçok yanlış negatif (false negatif) üretebilir. Bu sorunu çözmek için SSD sert negatif madencilik kullanmaktadır. Tüm arka plan örnekleri artan sırayla tahmin edilen arka plan puanlarına göre sıralanır. Hatanın hesaplanmasına devam etmek için sadece en üstteki K örnekleri tutulur. Ön plan örnekleri ve arka plan örnekleri arasında $\frac{1}{3}$ oranını korumak için her parti için anında hesaplanır.

Veri büyütme: SSD'de Perez ve Wang (2017)'da belirtilen bir sıra veri manipulasyon stratejisi kullanılır. Büyük nesnelerin tespitindeki performansı artırmak için "yakınlaştırma" stratejisi uygular. Rastgele bir alt bölge görüntüden seçilir. Ve eğitim için ağı beslenmeden önce standart boyuta SSD300 için 300×300'e ölçeklendirilir. Bu büyük nesneler için ekstra örnekler oluşturur. Benzer şekilde, küçük nesnelerin algılanma performansını artırmak için de "uzaklaştırma" stratejisi uygular. Boş bir tuval, orijinal görüntünün 4 katına kadar oluşturulur. Orijinal görüntü daha sonra rasgele olarak tuval üzerine yapıştırılır. Bundan sonra, eğitim için ağı beslenmeden önce tuval standart boyuta ölçeklendirilir. Bu, küçük nesneler için ekstra örnekler yaratır ve SSD'nin MSCOCO veri setindeki performansı bu şekilde arttırılmıştır.

Önceden Eğitilmiş Özelliklerin Çıkarılması ve Normalizasyonu: Önceden eğitilmiş diğer özellik çıkarıcıları kullanmak mümkün olsa da orijinal SSD makalesinde VGG16 modeli kullanılmıştır. Bununla birlikte, VGG16'da birkaç değişiklik yapılarak kullanılmıştır. Bu parametreler FC6 ve FC7'den alt örneklenir, daha büyük bir alıcı alan için FC6'ya 6'nın genişletme(dilate) işlemi uygulanır. Normalleştirme değişkenlerinin de eğitilebilir olduğu Conv4_3 katmanının çıkışına da L2 normalizasyonu uygulanır. Son olarak tahmin haritası doğrudan tespit sonucu olarak kullanılamaz. SSD300 de 8732 adet tahmin kullanılır. SSD,

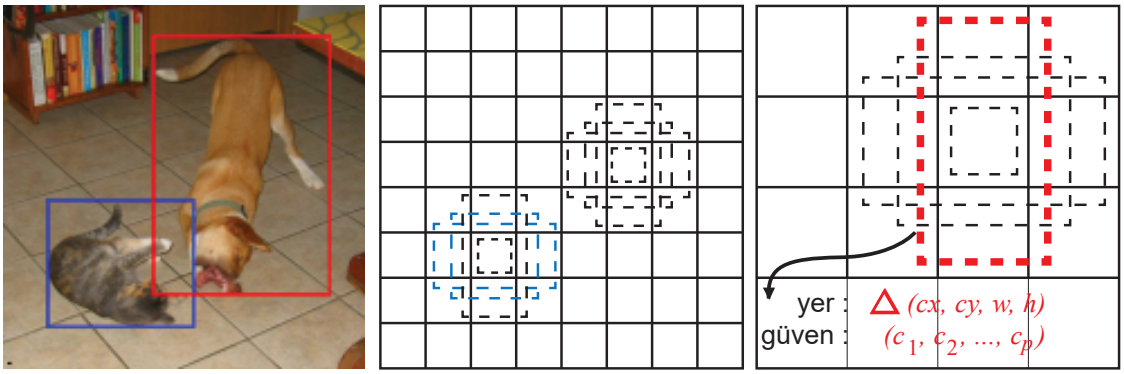
tahminlerin çoğunu filtrelemek için bazı basit sezgisel taramaları kullanır. İlk önce güven puanı eşiğine sahip zayıf algılamayı atar, ardından sınıf başına maksimum olmayanların baskılaması metodunu gerçekleştirir ve ilk 200 algılamayı seçmeden önce tüm sınıflardaki sonuçları değerlendirir. Son çıktı olarak mAP hesaplamak için ve yüksek hatırlama elde etmek için güven puanında düşük bir eşik kullanılabilir ($0,01$ gibi). Gerçek uygulamalarda $0,5$ gibi çok yüksek bir eşik kullanılır.



Şekil 3.8. SSD Liu ve ark. (2016)

Şekil 3.8'de standart bir SSD algoritması görülmektedir. Ekstra özellik haritaları ve VGG16 özellik haritasına eklenen ekstra katmanlar ile nesnelerin yerleri ve özellikleri çıkarılır.

Çoklu pencere stratejisi SSD'lerin sınırlayıcı kutuları çakışsa bile, farklı sınıflardaki nesneleri nerede olduğunun anlaşılmasına olanak tanır. Birçok nesne algılama algoritmasında, farklı sınıflardaki üst üste binen nesneler genellikle en yüksek güvenle tek bir sınırlama kutusuna konmaya çalışılır. Son olarak, detektör terimi, yalnızca bir görüntüdeki bir nesne kümesinin (x,y) koordinatlarını yerelleştiremeyeceğimiz aynı zamanda sınıf etiketlerini de geri döndüreceğimiz anlamına gelir.



(a) Zemin Gerçekliği ile sınırlandırılmış nesneler

(b) Özellik haritası

(c) 4x4 Özellik haritası

Şekil 3.9. Çeşitli özellik haritalarında bulunan sınırlayıcı kutular. Liu ve ark. (2016)

Liu ve arkadaşlarının Şekil 3.9 (b) kısmında görülen 8×8 özellik haritasında mavi ile işaretlenmiş öncül kutularında kedi bulunmuşken 4×4'lük özellik haritasında kırmızı ile işaretlenmiş öncül kutusunda köpek bulunmuştur. Görüldüğü gibi köpek kediden daha büyük bir objedir. Bu sebeple daha küçük bir özellik haritasında bulunur. Bu özellik haritalarının coğrafi boyutları CNN'nin son katmanlarında oluşturulması süreci, değişen görünüş oranlarıyla birleştirilmekte, değişen ölçeklerdeki, bakış açılarındaki ve oranlardaki nesneleri verimli bir şekilde lokalize etmeye izin vermektedir. Ayrıca, Faster R-CNN'de olduğu gibi, ham (x,y) koordinatlarını tahmin etmek yerine, her sınıf etiket için sınırlayıcı kutu ofsetleri olan deltalar tahmin edilir. Bunu daha iyi anlamak için x, y koordinatlarında b kadar sabit sınır kutularında ve c kadar da sınıf olduğunu kabul edersek. Hesaplanması gereken her özellik haritası başına miktar: Toplam=özellik haritası*sabit sınır kutu sayısı*sınıf+4 tanedir. Yine, öngörülen her sınır kutu için, tüm sınıflar arasında yalnızca en büyük olasılıkla sınırlayıcı kutuyu tutmak yerine, bölgedeki tüm sınıf etiketlerinin olasılığı hesaplanır. Sınır kutuların sınıfsal bir şekilde hesaplanması ve korunması, potansiyel olarak çakışan nesnelerin de algılanmasını sağlar.

SSD'yi eğitirken çoklu pencere algoritmasında iki bileşenli hata fonksiyonu kullanılır.

1. Sınıflandırma Hatası

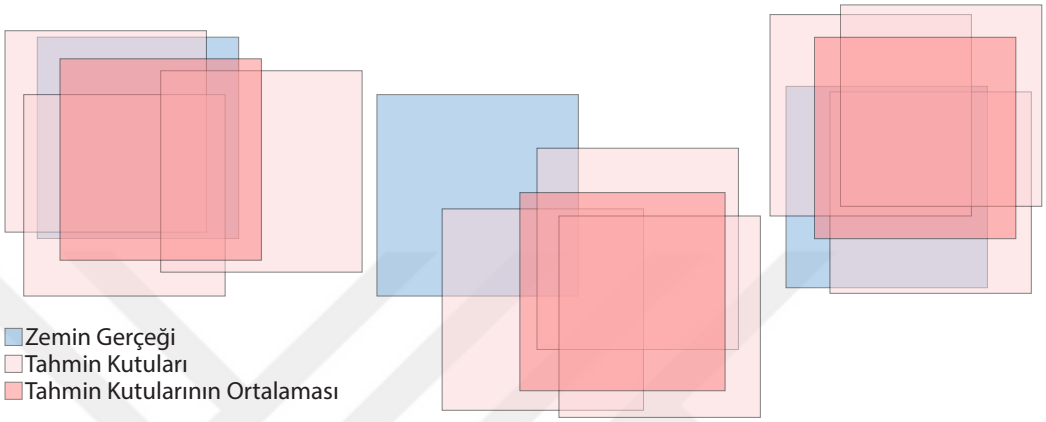
2. Yerelleştirme Hatası

Kategorik çapraz düzensizlik (Categorical Cross Entropy) kaybı, sınırlayıcı kutu için sınıf etiketi tahminimiz de doğru olup olmadığımızı ölçtüğü için güven için kullanılır. Faster R-CNN'ne benzer şekilde L1 hata fonksiyonu yerelleştirme kayıplarını bulmak için kullanılır. Yerelleştirme kaybı neticesinde bulunan sınır kutuları yeterince yakın olsa yeter. Zemin gerçeği kadar yakın olması beklenmemelidir. Ağı eğitmeden önce SSD tarafından kullanılacak varsayılan kutu kaç tane olacağına dikkat edilmelidir. Orijinalinde 4 veya 6 olması tavsiye edilmesine rağmen isterse daha farklı ebatta ve görüntü oranında ve sayıda varsayılan kutu eklenebilir. Bu daha fazla obje tespiti yapmanızı kesin olmamakla birlikte sağlayabilirken, zaman olarak büyük kayıp yaşatacağı kesindir. Aynı durumlar özellik haritalarının sayısı içinde geçerlidir. Daha fazla sayıda evrişimsel katman networke eklenebilir ve bu daha fazla nesne tespit etmeyi ve sınıflandırmayı sağlayabilir. Fakat zamanda negatif etkiler yapacaktır. Bu kalite ve hız arasındaki bir değiş tokuş gibidir. SSD ayrıca, eğitim doğruluğunu artırmak için sert negatif madencilik gibi ortak bir nesne algılama konseptini de içermektedir. Eğitim sürecinde, düşük IoU değerine sahip hücreler olumsuz örnekler olarak kabul edilir. Tanım olarak, bir görüntüdeki çoğu nesne görüntünün küçük bir bölümünde bulunur. Bu nedenle, görüntünün çoğunun olumsuz örnekler olarak kabul edilmesi olasıdır. Eğer hepsini kullanırsa bir girdi imgesi için negatif örnekler, negatif eğitim örneklerine pozitif oranları son derece dengesiz olacaktır.

Olumsuz örneklerin sayısının olumlu olanların sayısından fazla olmamasını sağlamak için Liu ve ark. Olumsuz örneklerin pozitif örneklere oranının $\frac{3}{1}$ civarında tutulmasını tavsiye eder. Kullanacak çoğu SSD uygulaması bu örneklemeyi varsayılan olarak yapar veya ağ için ayarlanabilir bir parametre olarak kullanır. Rastgele gradyen inişi optimize metodu olarak orijinal makalede kullanılmiştır. Fakat başka MSprop veya Adam gibi optimize ediciler de kullanılabilir. Tahmin süresince maksimum olmayanların baskılanması metodu kullanılarak bir cismin çevresinde yığılmış tahminlerden en yükseği seçilirken diğerleri

elemine edilir. Böylece cismin çevresine en çok uyan sınır kutusu bulunmuş olur.

Maksimum Olmayanların Baskılanması(NMS): CNN yapısını kullanarak nesne tespiti yapan yapılar SSD, YOLO veya R-CNN gereğinden daha fazla miktarda sınır kutusu üretir. Bunlardan en uygun olanının seçilmesi için NMS algoritması kullanılır.



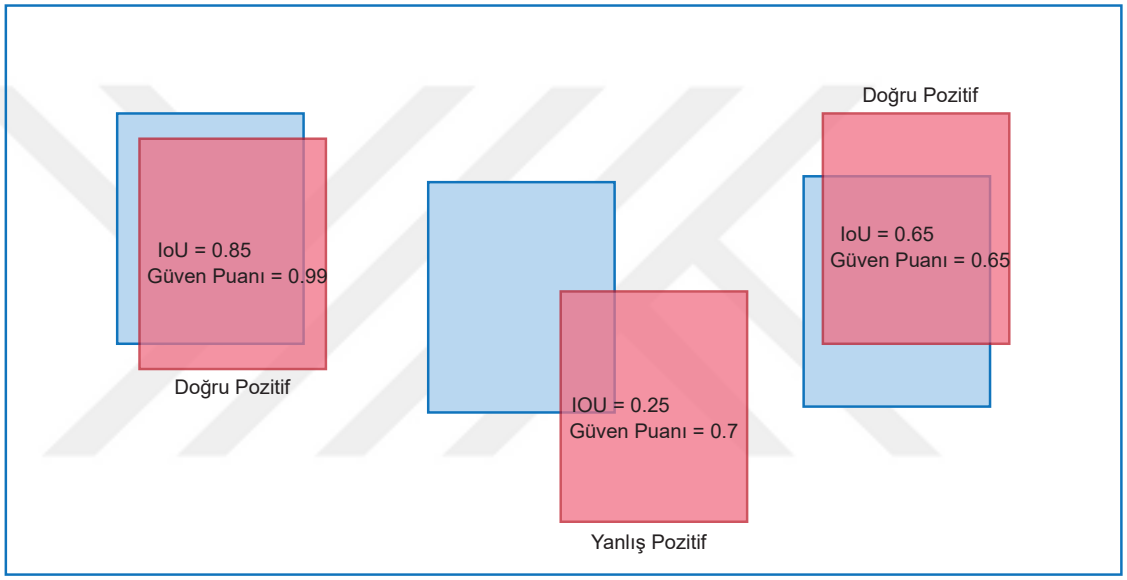
Şekil 3.10. NMS algoritması neticesinde çıkan sonuç[21]

Şekil 3.10'de Zemin gerçeği (Mavi), Tahmin Kutuları (Açık Pembe), Tahmin Kutularının ortalaması (Kırmızı) olarak belirtilmektedir. Mavi ile belirtilmiş kutunun içerisinde bizim tarafımızdan nesne tespit algoritmasına referans alması için verilen yer gerçeği vardır. Açık pembe renkli kutular ise bize nesne tespit algoritma sının tespit ettiği sınır kutuları göstermektedir. Ayrıca pembe kutuların hepsinde nesne tespit algoritmasının sınıf tespit için kabul ettiği eşik değeri aşılmıştır. Kırmızı renkli kutu ise bu sınıf skorlarının aşıldığı pembe kutuların ortalama değeridir. Bu ortalaması alınarak tek bir kutuda birleştirme işlemine maksimum olmayanların baskılanması denir.

Her görüntü için, yüzlerce başlangıç sınırlama kutusunun tahmin edilir fakat 0,5'in üzerinde nesne değerine sahip yalnızca 6 öngörü saklanır. Orjinal SSD'de bazı katmanlarda 4, bazı katmanlarda 6 adet sınırlama kutusu saklanır. 6 sınırlama kutusundan, birbirleriyle örtüşen (en yüksek IOU) ve aynı sınıfı öngören sınırlayıcı kutuların birlikte ortalaması alınır. Nihai tahminler yapıldıktan sonra, öngörülen sınırlayıcı kutular ile zemin gerçekleri nesne tespit

[21] https://tinyurl.com/y5pmw9mz, Erişim Tarihi: 06-07-2019, Saat: 14:30

edicinin ne kadar iyi çalıştığını anlamak için mAP puanı dikkate alınarak ölçülebilir. Bunu yapmak için, doğru pozitiflerin sayısının belirlenmesi gerekir. Tahmin edilen sınırlayıcı kutu bir zemin gerçeği sınırlayıcı kutusu ile IoU 0,5'den fazla örtüşürse, başarılı bir algılama olarak kabul edilir ve öngörülen sınırlayıcı kutu gerçek bir pozitif olur. Öngörülen bir sınırlama kutusu, eşik değerden daha düşük bir yerel gerçek (ground truth) ile örtüşmüşse, başarısız bir tespit olarak kabul edilir ve öngörülen sınırlama kutusu yanlış bir pozitiftir. SSD'de kesinlik ve tekrar çağırma hem doğru pozitiflerden hem de yanlış pozitiflerden hesaplanır:



Şekil 3.11. NMS algoritması sonuçları[22]

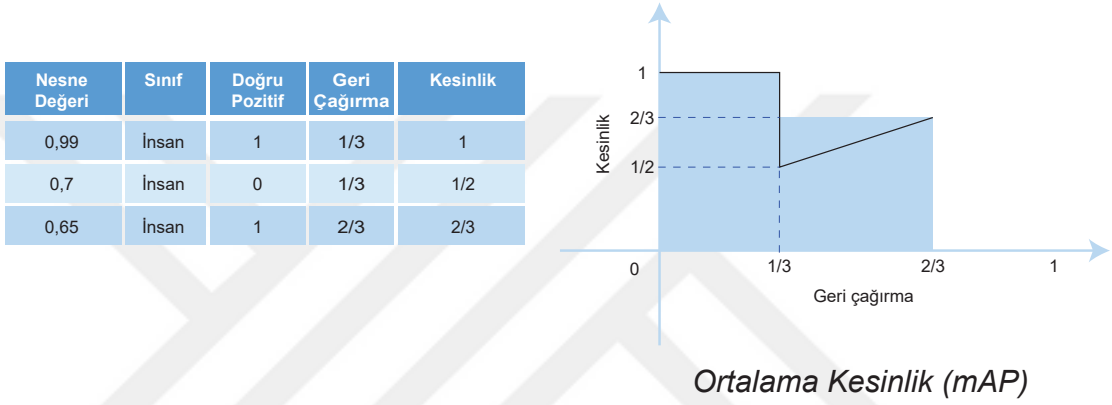
$$\text{Kesinlik} = \frac{\text{Doğru pozitiflerin sayısı}}{\text{Kırmızı kutuların sayısı}} = \frac{2}{3} \tag{3.1}$$

$$\text{Geri Çağırma} = \frac{\text{Doğru pozitiflerin sayısı}}{\text{Mavi kutuların sayısı}} = \frac{2}{3} \tag{3.2}$$

Geri çağırma ve kesinlik hesaplamaları denklem 3.1 ve denklem 3.2 dikkate alınarak hesaplanır. Şekil 3.11'de görüldüğü gibi yanlış pozitif güven puanı yüksek olsa bile IOU değerinin 0,5'den küçük olduğu yerlerdir.

[22]https://tinyurl.com/y5pmw9mz, Erişim Tarihi: 06-04-2019, Saat: 14:30

mAP'in Hesaplanması: Yukarıdaki adımdan elde edilen çıktılar, mAP değerini hesaplamak için kullanılır. Tahminler Nesne olup olmadığını gösteren nesne değeri puan değerine göre azalan bir sırayla sıralanır. Kesinlik-Geri çağırma eğrisi çizilir. Eğrinin altındaki alan mAP'dir. Hesaplanan alan dikdörtgen şeklindedir, bu nedenle grafiğin üçgen kısmı dikkate alınmaz. Her bir sınıf için mAP hesaplanır ve ardından tüm sınıfların ortalaması alınır. Geri Çağırma ve Kesinlik eğrisinin grafiği Şekil 3.12 gösterilmiştir:



Nesne Değeri	Sınıf	Doğru Pozitif	Geri Çağırma	Kesinlik
0,99	İnsan	1	1/3	1
0,7	İnsan	0	1/3	1/2
0,65	İnsan	1	2/3	2/3

Şekil 3.12. NMS algoritması sonucu çıkan sonuç[23]

SSD ile ilgili yapılan çalışmalardan elde edilen gözlemler:

1. Daha fazla varsayılan kutu, hız üzerinde negatif bir etkiye sahip olmasına rağmen, daha doğru algılama sağlar.

2. Çoklu kutunun çoklu katmanlar üzerinde olması, çoklu çözünürlükteki özelliklerde çalışan detektör nedeniyle de daha iyi algılama sağlar VGG-16 şebekesinde zamanın %80'i harcanır: bu, daha hızlı ve eşit derecede doğru bir ağ ile değiştirilebilirse SSD'nin performansının daha iyi olabileceği anlamına gelir.

[23]https://tinyurl.com/y5pmw9mz, Erişim Tarihi: 06-04-2019, Saat: 14:30

SSD, benzer kategorilere sahip nesneleri karşılaştırır (örneğin hayvanlar). Bunun nedeni, konumların birden fazla sınıf için paylaşılmasıdır. SSD-500 (521×512 giriş görüntülerini kullanan en yüksek çözünürlük değişkeni) Pascal VOC2007'de %76,8 ile kare hızının 22 fps'ye düştüğü hız pahasına en iyi mAP değerini elde eder. SSD-300 bu nedenle 59 fps'de 74,3 mAP ile çok daha iyi bir denge oluşturuyor. SSD açılımı olan bu metot tek geçişte objenin, bir görüntüde ne olduğunu ve nerde olduğunu bulmayı amaçlayan iki metottan biridir diğeri de YOLO metodudur. Bunu amaçlamasının sebebi hızlı bir şekilde objenin tespit edilmesini gerçekleştirebilmektedir. Çünkü CNN katmanından tek geçişte ikisini de tespit eder. Bu metodun anlaşılması CNN'nin iyi anlaşılmasına çok bağlıdır.

Eğitim aşamasında konvulasyon katmanlarında bulunan filtreler, test aşamasında görüntülerin üzerine yine konvulasyon işlemi ile tekrar uygulanır ve bu filtreler görüntüde kendisi ile ilgili yerlerde yüksek değerler alır ve bu değerler kullanılarak nesneler bulunur. Örnek olarak bir görüntüde olan iki kedinin yerinin bulunması ve kedi olarak etiketlenmesi süreci ele alındığında SSD'de eğitim aşamasında bulunan kedi ile ilgili aynı filtreler kullanılır. Kediye ait olan filtreler örneğimizdeki görüntü üzerinde konvolusyon işlemine tabii tutularak uygulandığında aynı sobel örneğinde olduğu gibi kedinin olduğu yerlerde yüksek değerler verir. Burada geriye kedinin ebadını bulmak kalmıştır. Bunun için çok katmanlı yapıdan yararlanılır. Evrişim katmanları derinleştikçe özellik haritaları bir hücrenin, pikselin gerçek resimde temsil ettiği yer artar. Dolayısıyla daha derin katmanlardaki kediye ait filtrelerde hem kediye ait daha detaylı parçaların yerlerini bulurken aynı zamanda gerçek resimde daha büyük alanlarda kedinin nerede olduğunu buluruz. En son olarak bunları anlamlı şekilde birleştirmek gerekmektedir.

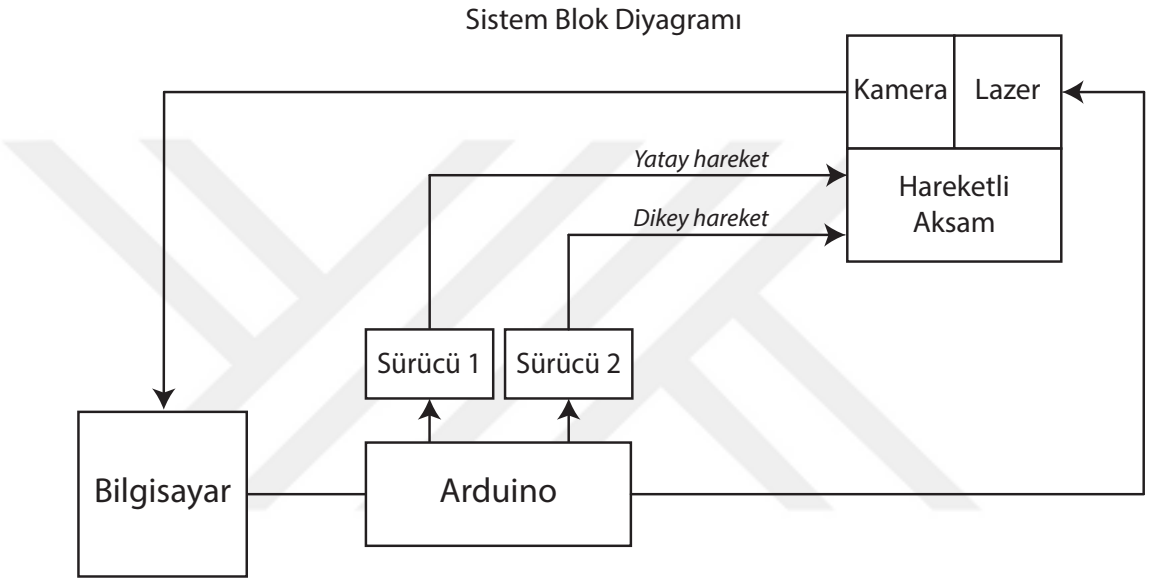
3.2. Takip Cihazının Tasarımı

3.2.1. Sistem Tanıtımı

Tezin amacı sadece SSD yöntemini kullanılarak herhangi bir nesne tanımanın yapılmasının yanı sıra gerçek bir platformda bu nesnenin takip edilmesine de yöneliktir. Sistem genel olarak SSD algoritması kullanılarak bulunan nesnelerin çevresine çizilen sınırlandırma

kutularının orta noktasına adım motorları vasıtasıyla ulaşmaya çalışır ve orta noktanın belirli bir çevresine geldiğinde lazer ışığıyla o noktaya ateş eder. Cisim hızlı ise daha çok adım atarak cisme yetişmeye çalışır. Ve geldiği nokta ile cisimin bulunduğu nihai nokta arasında sürekli olarak yön ve hız hesaplaması yapar.

3.2.2. Donanım bileşenleri



Şekil 3.13. Takip makinesi blok diyagramı

Mekanik sistem şu bileşenlerden oluşmaktadır:

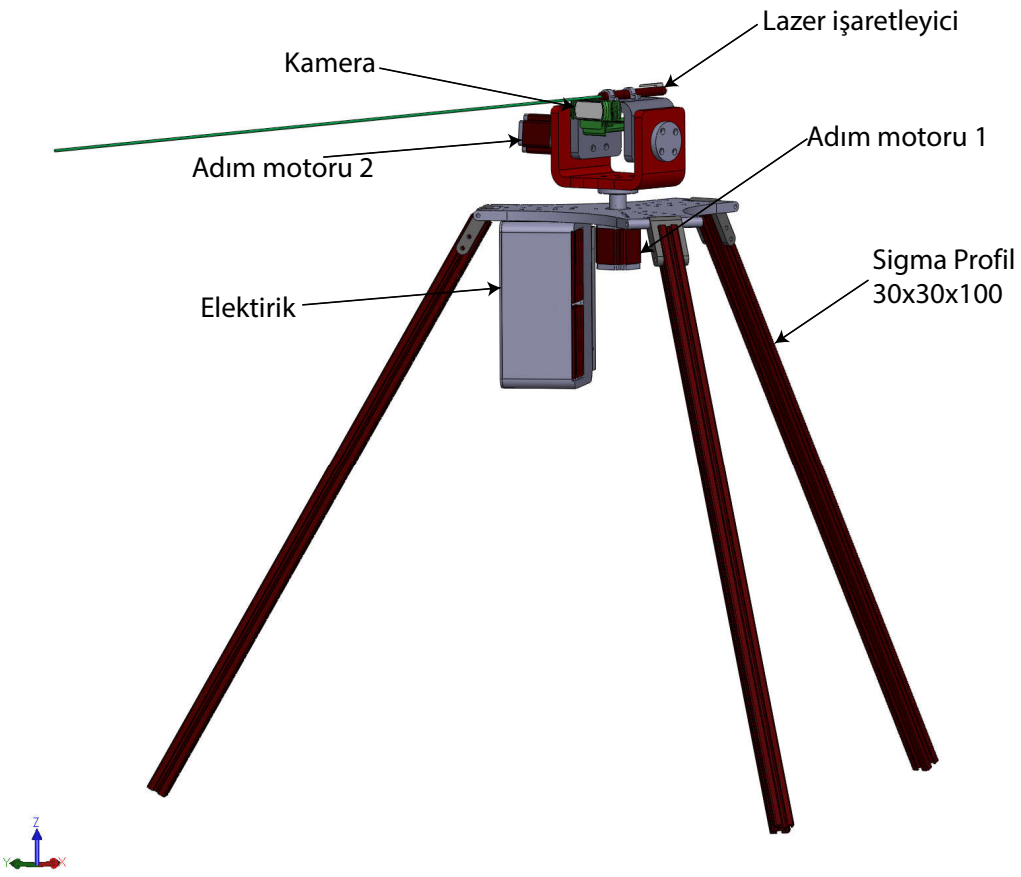
- 1 adet Bilgisayar Intel 7700HQ işlemci- 1050Ti Nvidia GPU- 16 GB ram
- 1 adet Kamera Logitech C920 HD
- 1 adet Arduino UNO
- 2 adet Güç Ünitesi MW LRS-200-48KO 48V-4,4Amp sabit güç ünitesi
- 2 adet Sürücü TB6600 5A Step Motor Sürücü
- 2 adet Adım Motoru Nema23 57×82 mm 3Amp 2,2NM

– 1 adet Acil stop Standart tip

– 1 adet Elektrik panosu Plastik

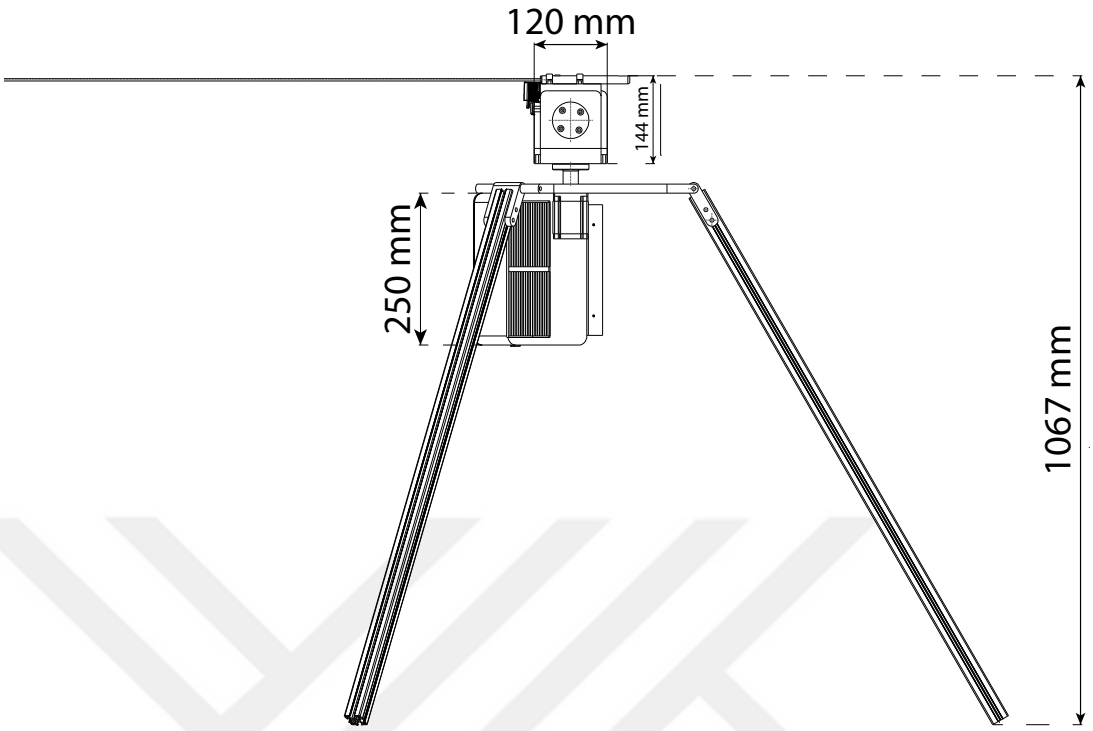
– 1 adet Gövde Alüminyum

– 3 adet Taşıyıcı bacak $30 \times 30 \times 100$ Sigma Profil



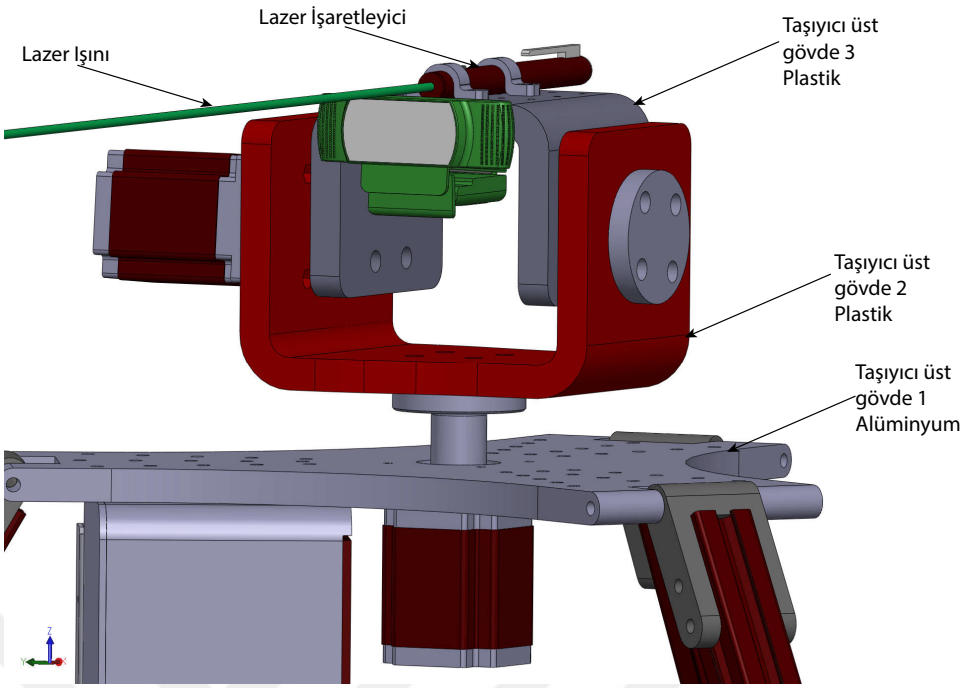
Şekil 3.14. Takip makinesi genel görüş

Şekil 3.14'de görüldüğü gibi sistem adım motoru 1 ile yatay eksendeki nesneleri takip edebilmektedir. Düşey eksendeki objeleri takip edebilmek için ise adım motoru 2 yi kullanır.



Şekil 3.15. Takip makinesi ebatlandırma görünüşü

Makinenin ebatları Şekil 3.15'de görüldüğü gibidir. Ölçülerin hepsi mm olarak verilmiştir. Sigma profil olan ayakların açısı istenildiği gibi ayarlanabilmektedir. Dolayısıyla yükseklik izafidir ve Sigma profil ayakların açısına göre değişir. Ayakları gövdeye bağlayan aparat özel olarak kolay ayar yapılabilmesi için kendinden vidalı olarak seçilmiştir. Elektrik panosu gövdeye cıvata bağlantısı ile sabitlenmiştir. Lazer işaretleyici ile kameranın paralel olmasını sağlamak için üst taşıyıcı gövdede delikler kullanılarak bağlantı yapılmıştır. Ayrıca sigma profillerin altında zeminden kaymasını önleyecek şekilde ek plastik parçalar konulmuştur.



Şekil 3.16. Takip makinesi ebatlandırma görünüşü

Yukarıdaki Şekil 3.16'de görüldüğü gibi üst gövde 2 ve 3, 3d baskı teknolojisiyle plastik olarak üretilmiştir. Bunun sebebi ağırlıktan kaçınmak içindir. Ağırlıktan kaçınılmak istenmesinin nedeni: Adım motorunun özelliklerinden kaynaklanmaktadır. Adım motorlarının frenleme kabiliyeti sarımlarında ürettikleri manyetik alan kuvveti kadardır. Makinenin yaptığı hızlı hareketler sonucu eğer ağırlık fazla olmuş olsaydı, durması gereken yerde durmayıp fiziksel atalet kuvvetlerinin etkisiyle kendiliğinden adım atacaktı bu sebeple 3d baskı teknolojisinin petek şeklinde üretim avantajı ve plastiğin hafiflik özelliği beraber kullanılmıştır. Ek olarak taşıyıcı gövde 1 in alüminyum seçilmesinin sebebi de taşınma kolaylığı sağlaması içindir. Üzerine delikler açılarak makinenin mümkün olduğunca hafifletilmesi istenmiştir. Buna rağmen ağırlığı 16,5 kg'dır.

3.2.3. Yazılım

Sistem yazılımsal olarak iki kısımdan oluşmaktadır. Bilgisayarda geçekleşen kısımda Tensorflow organizasyonunun standart olarak sunduğu SSD tensorflow uygulamasının üzerine koordinatları alarak arduinoya gönderen kısım eklenerek program gerçekleştirilmiştir. Yine aynı şekilde nesnelerin bulunması için gerekli algoritma ve ağırlıklar Google tarafından standart olarak sunulmuştur. Uygulamada "ssd_mobilnet_v1_coco_11_06_2017" versiyonuna göre bulunan yine standart ağırlıklar kullanılmıştır. Bu versiyonun kullanılmasının sebebi SSD'nin orjinal makalesinde[24] 79.6 mAP değeriyle en yüksek sonucu almasıdır. Ayrıca diğer verisyonlarla yapılan deneylerde takılmalar ve tespit eksiklikleri gözlemlenmiştir.Python yazılım dili kullanılarak yazılım gerçekleştirilmiştir. Arduino tarafında ise c dili kullanılmıştır.

1. Nesnelerin tespitini yapan bilgisayar modülü: Nesne tespitinin yapıldığı ve SSD algoritmasının işletildiği kısım bilgisayarda python dili vasıtasıyla yazılmıştır. Tensorflow-GPU versiyonu obje tespit kütüphanesi olarak kullanılmıştır. Yazılım genel olarak şu şekilde çalışır.

   (a) Sisteme ssd_mobilnet_v1_coco_11_06_2017 önceden eğitilmiş model yüklenir.

   (b) Kameradan gelen görüntüler(640×480, 25fps) hafızaya alınarak hem Pascal-VOC formatına hem de 300×300 ebatına dönüştürülür. Bunun sebebi SSD algoritmalarından 300×300 formatının yani SSD300 seçilmesi ve bu algoritmanın Pascal-VOC formatında çalışmasıdır. Eğitim için gerekli veriler ssd_mobilnet_v1_coco_11_06_2017 dosyasının içerisinde olduğundan ve bunu direk kullanılması sebebiyle bir eğitim yapılmasına gerek yoktur.

   (c) Eğitim modeli mevcıt olduğu için direk olarak nesne tespitine geçilebilir. Hafızadaki veriler sırasıyla okunarak daha önce SSD konusunda anlatıldığı

[24] https://tinyurl.com/yyv3tnke, Erişim Tarihi: 05-01-2019, Saat: 13:00

gibi önceden eğitilmiş modelden gelen filtrelerle evrişim işlemine tabi tutulur. Ve bulunan sonuçlardan 8732 tane her sınıf için tahmin üretilir.

(d) Bu tahminler hem güven hem de zemin kaybından uzaklık hatasını içeren lokalizasyon kayıplarıdır. Bu aşamadan sonra gereksiz birçok sayıda sınırlandırıcı kutu ile uğraşmamak için, güven puanının belirli bir eşiğin üstünde ve lokalizasyon kayıplarının belirli bir eşiğin altında olanların seçildiği ve sonrasında da ortalamalarının alınıp sınırlandırıcı kutunun nesnenin çevresine çizildiği NMS aşamasına gönderilir. Buraya kadarki işlemler standart Google yazılımıdır ve detaylı bilgilere[25] bağlantısından ulaşılabilir. Dikkat edilmesi gerek husus verilen linkin dinamik olduğu ve sürekli Google tarafından güncellendiğidir. Tensorflow versiyonu, protobuf versiyonu ve ssd versiyonları sürekli güncellenmektedir.

(e) Bulunan nesnenin takip edilmesi ve gerekli performans değerlerinin ölçülmesi için bazı eklemeler yapılmıştır.Bunlar:

- Konum bilgileri bir global değişkene atanarak Arduino kütüphanesinin kullanılabileceği hale getirilmiştir.

- Mevcut konumla sürekli SSD300'den gelen konum kıyaslanarak makinenin doğu-batı ve/veya kuzey-güney yönünde hareket edip etmeyeceği ve edecekse ne kadar hareket edeceğinin hesaplandığı bir algoritma yazılmıştır. Bu algoritma makine kamerasından gelen bilgi ile SSD300'den gelen veriyi sürekli olarak kıyaslayarak yönü ve atması gerek adım sayısını hesaplar.

- Gidilen yönde kameranın merkez noktasının belirli bir toleransındaki yere tespit edilen cismin merkez noktası çatışırsa program lazer işaretleyiciyi aktif ederek cismi işaretler. Cisim, en kolay hareket ettirerek ölçüm yapabilecek nesne olarak insan seçilmiştir. Dolayısıyla sadece insanı tespit

[25]https://tinyurl.com/yyndktuh, Erişim Tarihi: 07-10-2018, Saat: 09:00

edebilmektedir. Fakat duruma göre diğer sınıflarında açılmasıyla 99 adet çeşitli nesneyi takip edebilir.

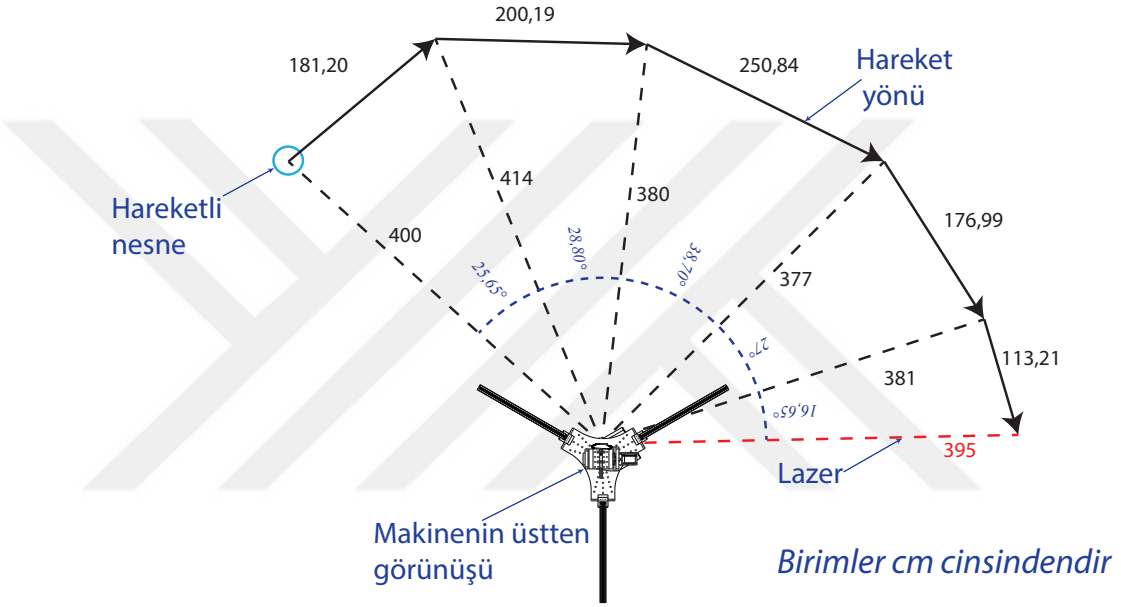
- Son olarak tez yazarının boyuna (1,72m) göre ölçüm yaparak kalibre edilen bir fonksiyon yardımıyla yazılım, kamera karşısına geçen insanların boyunu, hızını, kat edilen mesafeyi kayıt altına alarak resimlerini çeker ve kayıt eder.

2. Makineye hareket veren Arduino kısmı: Makine kısmında ise arduino ve içerisindeki program vardır. Ve bu program yardımıyla makine istenilen yönde hareket ettirilir. Program bilgisayardan sadece 'E', 'W', 'N', 'S' şeklinde yön bilgilerini alır. Python kısmında hesaplanan adım miktarı kadar arduino sürücüler yardımıyla adım motorlarını 300 milissaniye de bir hareket ettirir. Sürücü 4,2 amper ayarında ve $\frac{1}{8}$ adım atacak şekilde seçilmiştir ve adım motorunun adım sayısı 200'dür. Dolayısı ile her adımı 0,6 saniyede bir $\frac{360}{200\times8} = 0,225$ derecedir. Bu parametreler seçilen bileşenlere göre makinenin kararsızlığa düşmeden hareket edebildiği değerlerdir. Gelen yön bilgilerine göre sürücü adım motorlarını sürer bu şekilde cisim takip edilmiş olur. Bilgisayar kısmından yön verileri sıralı şekilde gelmez aynı anda çok izlekli olarak gönderilir. Böylece çapraz hareketler hipotenüsü izleyerek gerçekleştirilir.

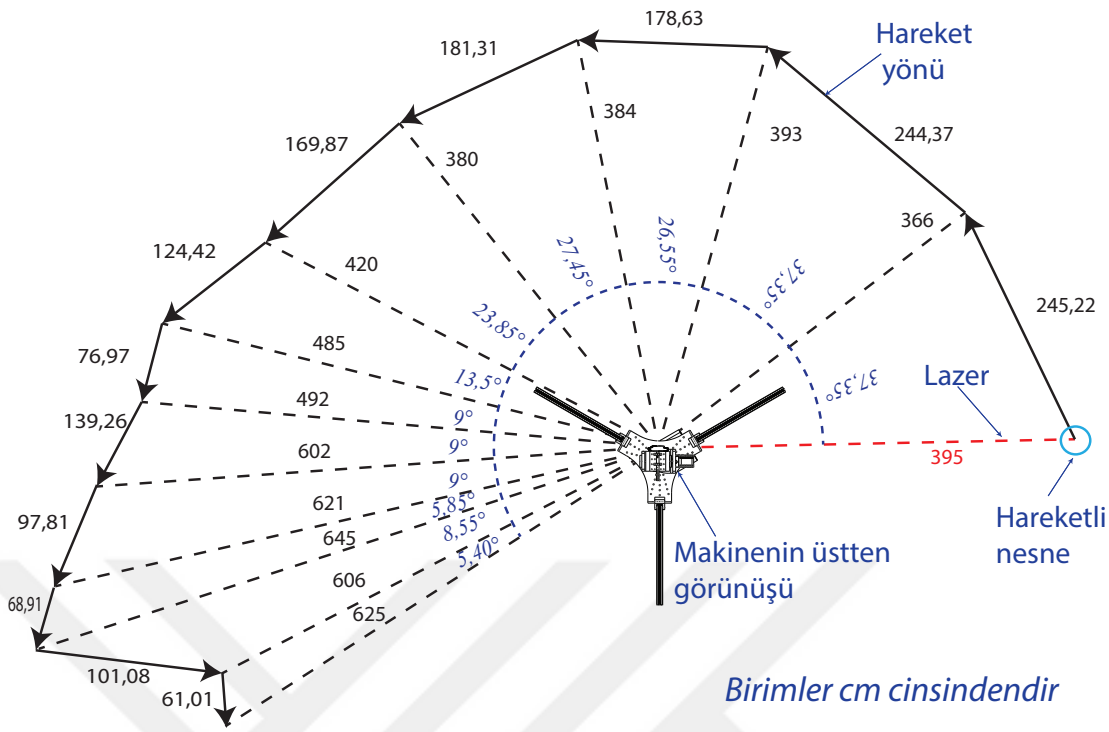
4. BULGULAR

4.1. Takip Sistemi Testleri

Sistemin hem program olarak hem de mekanik olarak başarısını ölçmek için yapılan deneyde mekanik sistem belirli bir noktaya konularak kişiyi takip etme kabiliyeti ölçülmüştür. Öncelikle istenilen nesneyi bulabilme sonrasında ise çeşitli hızlarda takip edebilme kabiliyetine bakılmıştır.

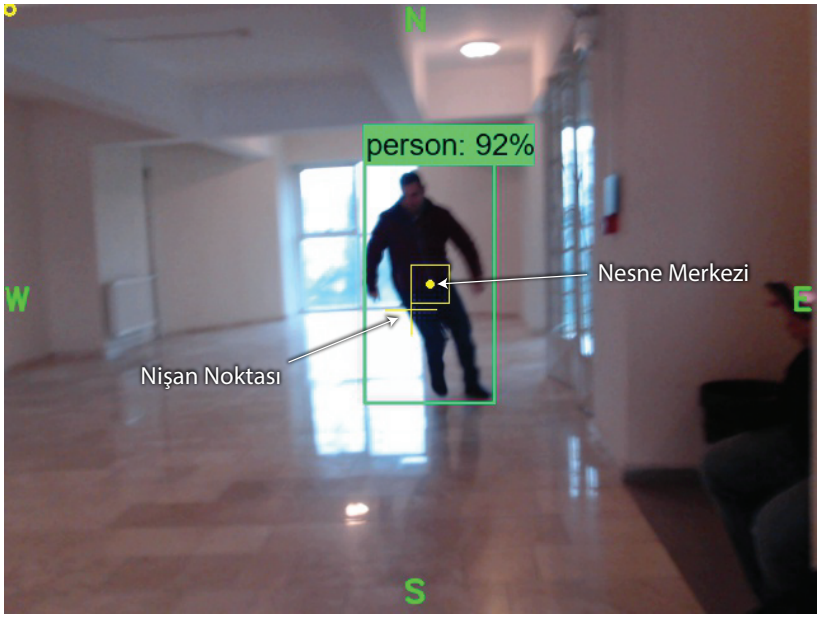


Şekil 4.1. Saat yönü hareket



Şekil 4.2. Saatin tersi hareket

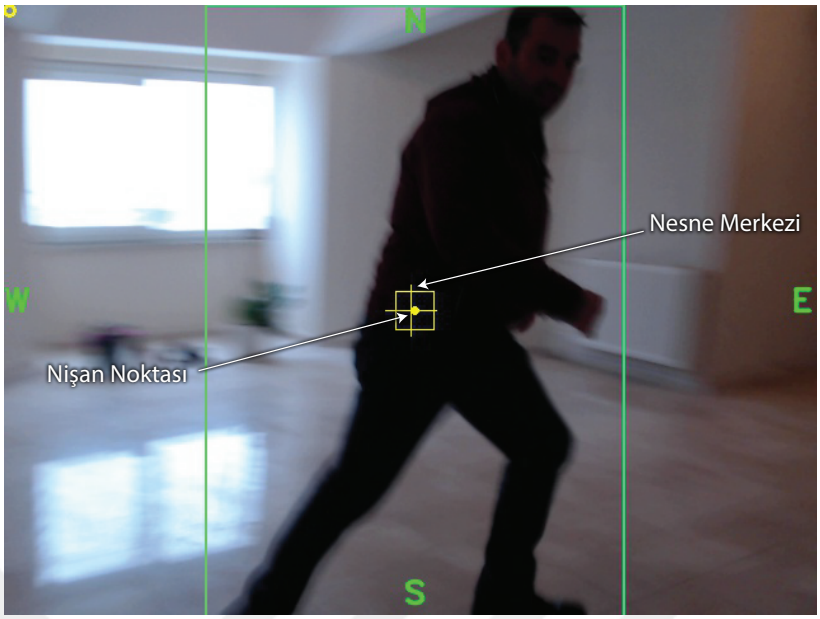
Şekil 4.1'de ve Şekil 4.2'de nesnenin tespit edildiği mesafeler görülmektedir.



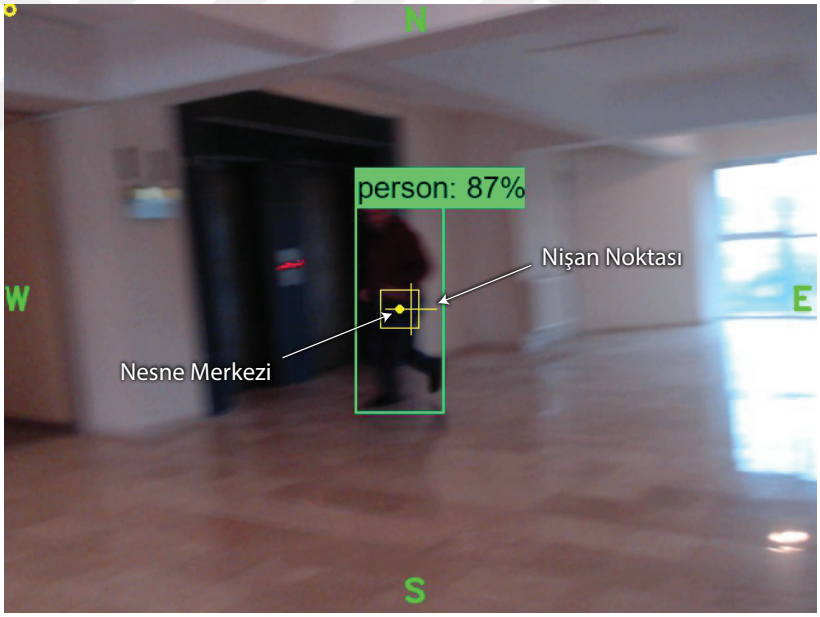
Şekil 4.3. İsabet görüntüsü, uzaklık=5,37 m, hız=6,95 km/saat

Şekil 4.4. İsabet görüntüsü, uzaklık=4,18 m, hız:7,6 km/saat

Şekil 4.5. İsabet görüntüsü, uzaklık=3,30 m, hız:5,5 km/saat



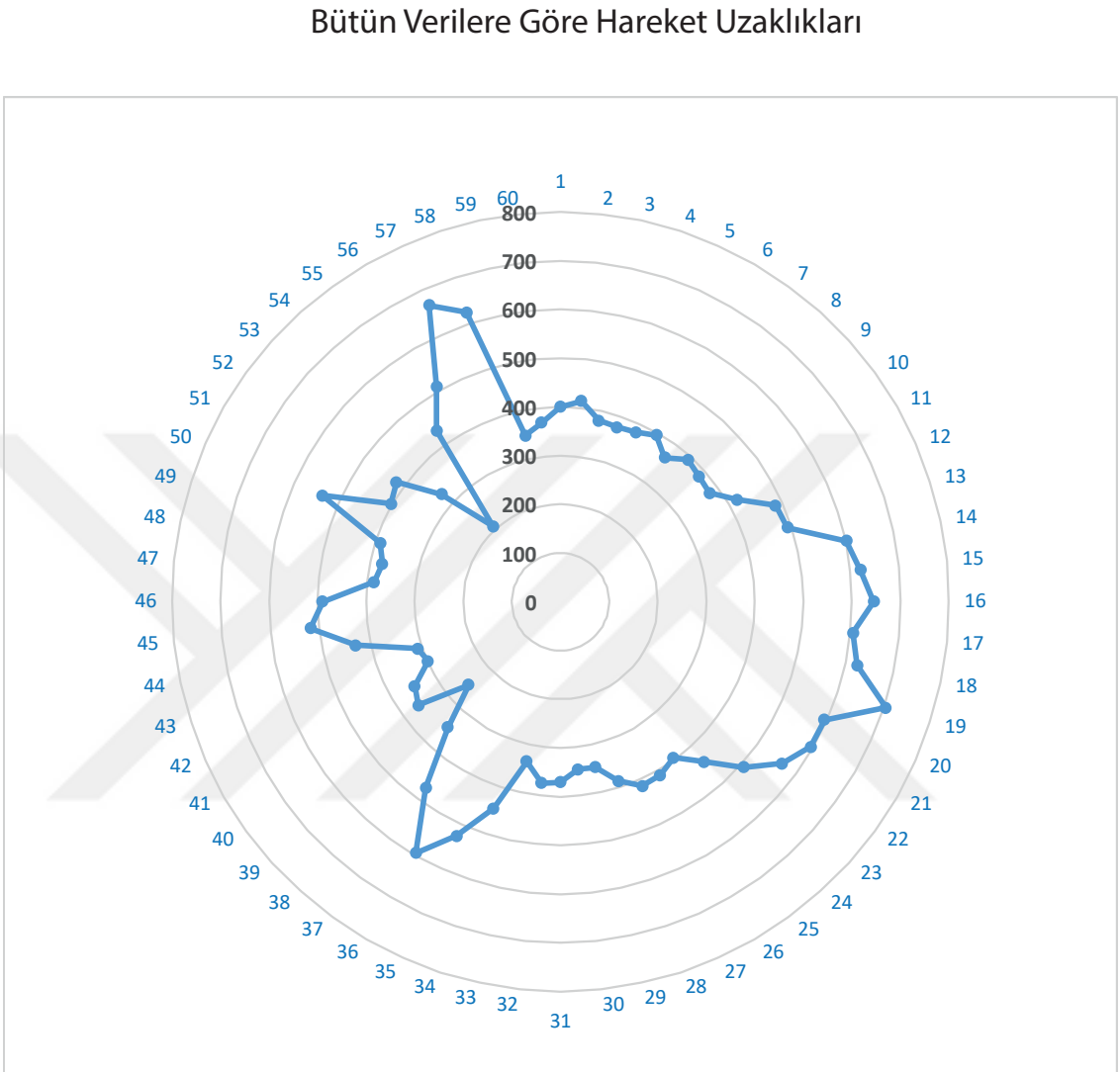
Şekil 4.6. İsabet görüntüsü, uzaklık=2,08 m, hız:6,83 km/saat



Şekil 4.7. İsabet görüntüsü, uzaklık=6,66 m, hız:10,34 km/saat

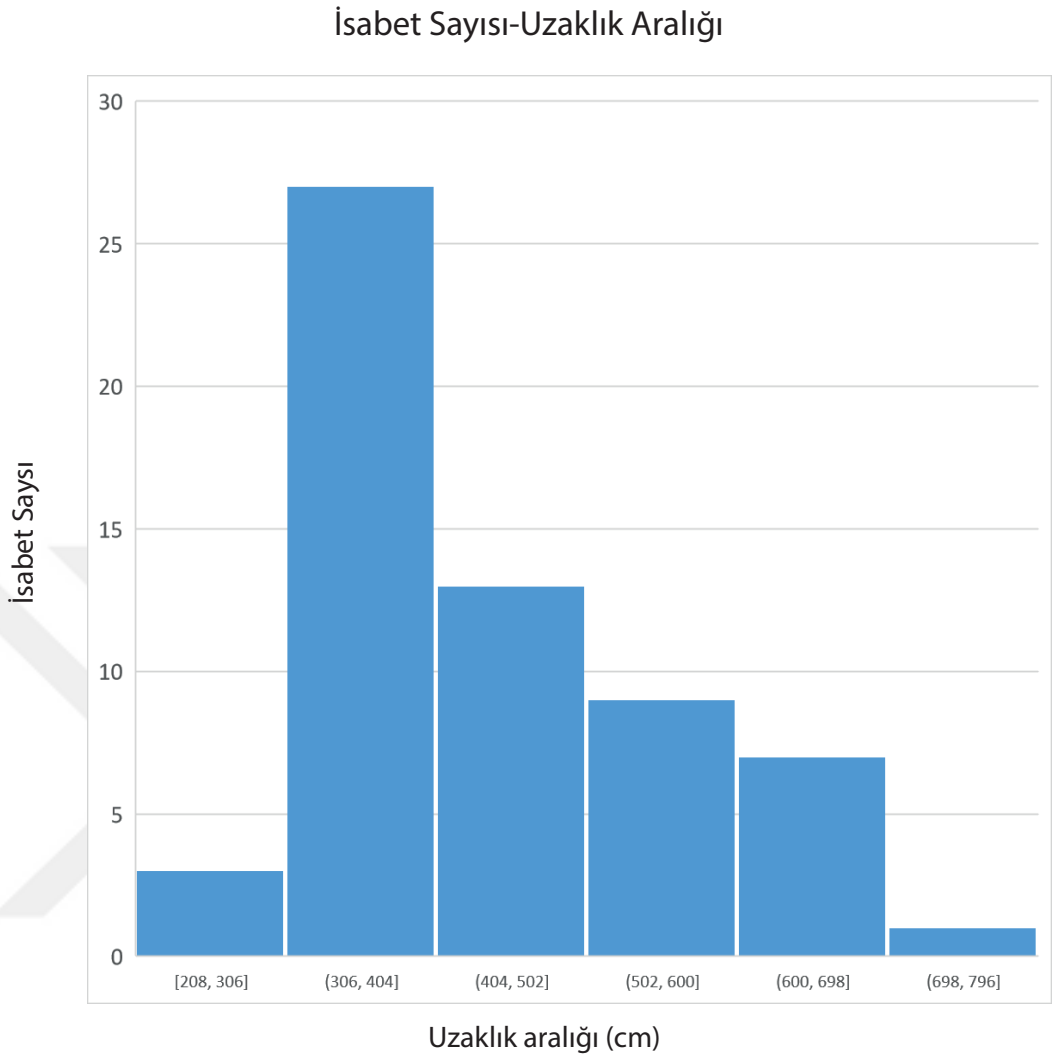
Test sırasında saat yönünde ve tersi yönünde çeşitli mesafelerde ve hızlarda koşularak hesaplamalar yapılmıştır. Her bir saat yönünde hareket bittiğinde tersi yönde harekete başlanmış ve sürekli koşularak toplam 135 m boyunca olan ölçümler kayıt edilmiştir. Ortalamalar bir yöndeki hareket ve isabet sayıları referans alınarak hesaplanmıştır.

4.2. Deney Sonuçları



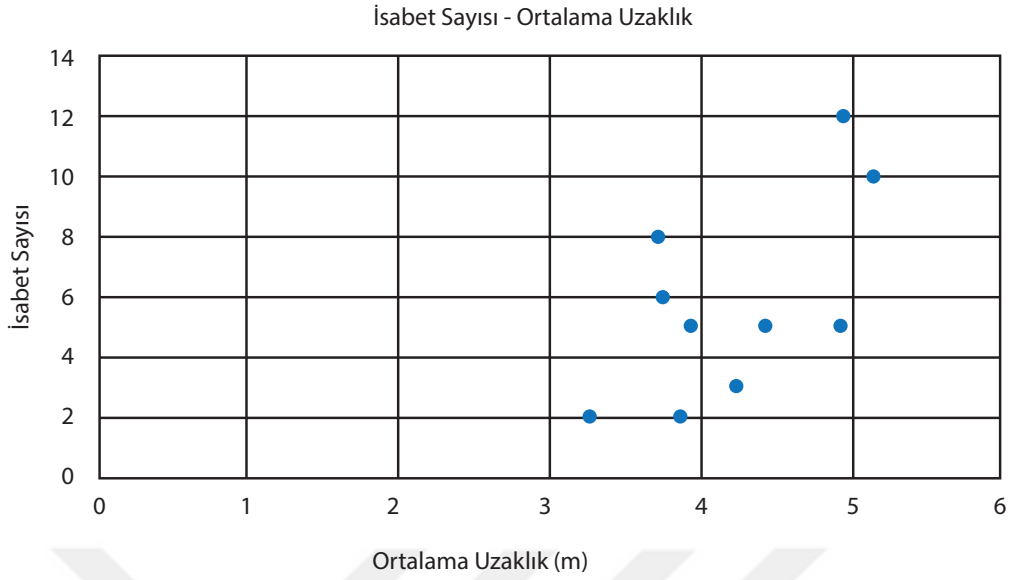
Şekil 4.8. İsabet yerlerine göre uzaklıklar (cm)

Deney sonuçlarında "uzaklık" makineye olan uzaklık anlamına gelirken, "mesafe" nesnenin kendi yönünde hareketi sırasında iki isabet arası uzaklık anlamına gelmektedir. Deneyde toplam 11 adet saat yönü ve tersine hareket edilmiştir. Şekil 4.8'de kişinin hareketlerine göre isabet alığı yerler gözükmektedir. Yani kişinin makineye göre uzaklıkları göstererek bütün koşu sırasında isabet aldığı yerlerdir. Merkezden en dış çembere olan sayılar uzaklıkları gösterirken dışardaki rakamlar isabet sayısılarıdır. Toplam hareket mesafesi 135,2 m kadardır. Ve bu mesafe miktarında nesne 60 kere isabet almıştır.



Şekil 4.9. İsabet Sayısı - Ortalama uzaklık X-Y grafiği

Şekil 4.9'de görüldüğü gibi deney sırasında nesnenin hareketlerine göre isabet aldığı uzaklıklar gösterilmektedir. En çok isabet sayısı 3 m ile 7 m arasında alınmıştır.

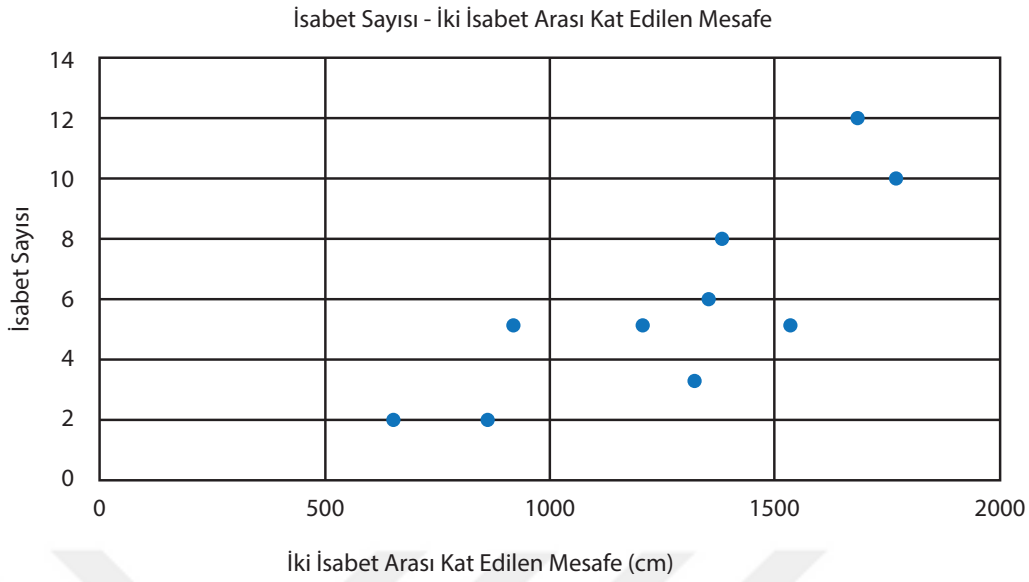


Şekil 4.10. İsabet Sayısı - Ortalama uzaklık X-Y grafiği

Çizelge 4.1. Hareketlerin ortalamasına göre değerlendirmeler

İsabet Sayısı	2	8	6	2	5	3	5	5	12	10
Ortalama Uzaklık	3,24	3,70	3,72	3,84	3,91	4,22	4,40	4,92	4,93	5,13
Ort. Uz. Ortalaması	4,2									
Ek küçük değer	3,24									
En büyük değer	5,13									
Standart Sapma	63									

Şekil 4.10 ve Çizelge 4.1 'deki veriler incelendiğinde görülmektedir ki: nesnenin hareketleri 3 m ile 5 m civarında yoğunlaşmaktadır. Bu mesafelerde ve Şekil 4.13 ve Çizelge 4.13 sonuçlarına göre makine 8 $\frac{km}{s}$ hızlarında çok efektif olarak hareketli nesneyi takip edebilmiştir.

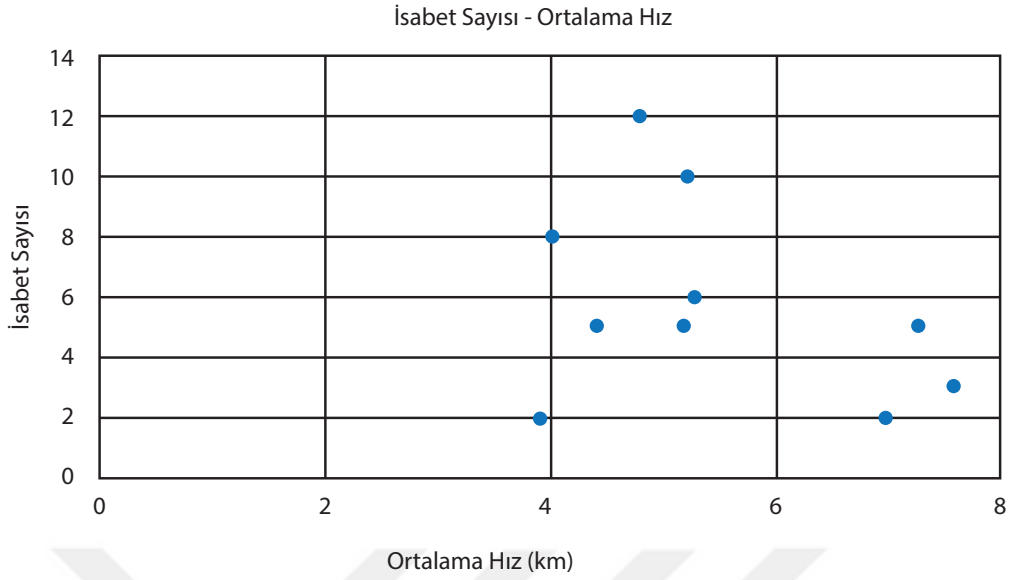


Şekil 4.11. İsabet sayısı - İki isabet arası kat edilen mesafe X-Y grafiği

Çizelge 4.2. İsabetler arası kat edilen masafe

İsabet Sayısı	2	8	6	2	5	3	5	5	12	10
Kat edilen mesafe	647	852	1315	922	1202	1533	1351	1388	1773	1689

Şekil 2.6'de Çizelge 4.2'de nesnenin kat ettiği iki ortalama mesafe arasındaki isabet sayısı görülmektedir. İki sayısı arasında 0,8 gibi kuvvetli bir korelasyon bulunmaktadır. Makinenin karşısında koştuğunuz mesafe artıkça makinenin sizi isabet ettirme sayısı da artmaktadır.

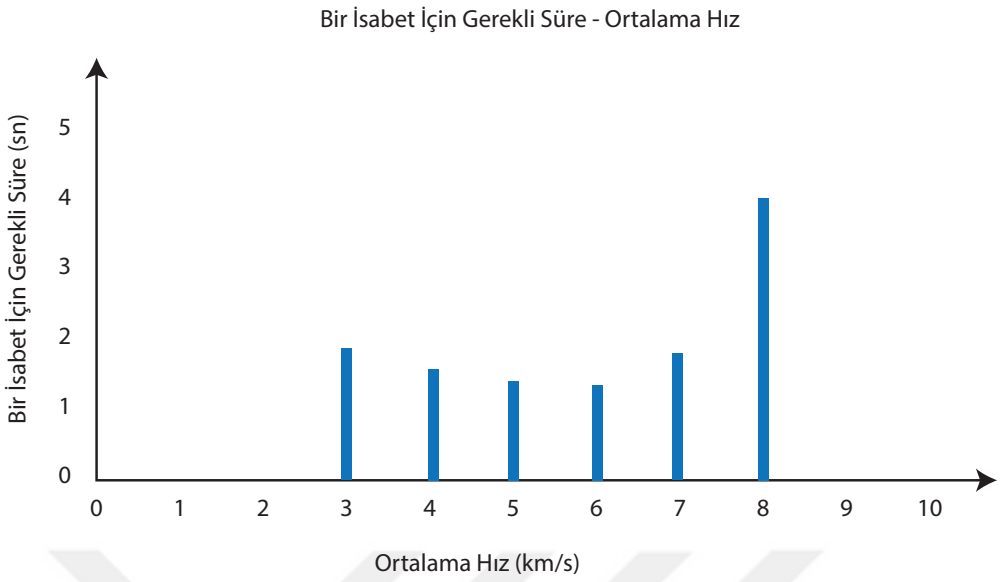


Şekil 4.12. İsabet Sayısı - Ortalama Hız X-Y grafiği

Çizelge 4.3. İsabetler arası kat edilen masafe

İsabet Sayısı	2	8	6	2	5	3	5	5	12	10
Ortalama Hız	3,90	6,95	7,54	5,17	4,41	7,23	5,24	3,97	5,17	4,81

Şekil 4.12 ve Çizelge 4.3'de çeşitli hız değerlerinde alınan isabet değerleri gözükmektedir. Aralarındaki korelasyon katsayısı -0,37'dır. Sayının eksi olması hız arttıkça isabet sayısının azaldığını göstermektedir. Bir önceki örnekte görüldüğü gibi mesafe ile ilişkiye göre hız ile olan ilişkisi çok kuvvetli değildir.



Şekil 4.13. Bir isabet için geçen süre - Ortalama Hız

Çizelge 4.4. Belirli hızlarda bir isabet için gerekli süre

Bir isabet için gerekli süre.	1,87	1,58	1,41	1,35	1,80	4,01
Ortalama Hız	3	4	5	6	7	8

Şekil 4.13 ve Çizelge 4.4 sonuçlarına göre koşulan hızlarda yapılan isabetler için geçmesi gereken hızlar gözükmektedir. Sistem 3 $\frac{km}{s}$, 4 $\frac{km}{s}$, 5 $\frac{km}{s}$, 6 $\frac{km}{s}$, 7 $\frac{km}{s}$'lik hızlarda yakın vuruş süreleri elde etmiş olamasına rağmen 8 $\frac{km}{s}$ hızında süreler uzamaya başlamıştır. Bu sistemin sınırlarına geldiğini göstermektedir. Bundan yüksek hızlarda da isabet yapmış olasına rağmen veriler güvenilir değildir. 8 $\frac{km}{s}$'lik hızda nesneleri kaybetmemekte ve kararsızlığa düşmemektedir.

5. TARTIŞMA VE SONUÇ

Makine test sonuçlarından görülebildiği gibi hızlı yürüme sınırları olan 8 km sınırına kadar çok stabil ve hiç kararsızlığa düşmeden çok kısa sürelerde isabet değerleri elde edebilmektedir. Bunun üzerindeki hızlarda yine isabet kaybetmektedir fakat nesnenin kameranın görüş açısından çıktığı durumlar oluştuğunda makine kendi sınırlarına eriştiği için nesnenin nerede olduğunu bilememekte ve kararsızlığa düşmektedir. Ve nereye gideceğini bilmeden belirsiz yönlerde hareket etmektedir. Fakat eğer belirsiz yöndeki bu hareket devam ederken tanıyabildiği bir nesne görürse tekrar kararlı duruma geçmekte ve nesneyi takip edebilmektedir. Tanıyabildiği nesne grupları ile aynı anda karşılaştığı durumlarda eğer nesneler sabit duruyor ise bir nesnenin ortasını bulup sabit kalmakta ve nesnelerin herhangi birinde hareket başlarsa o nesneyi takip etmeye başlamaktadır. Bu hareket tâ ki bir nesnenin diğer nesneyi kesişip kapatana kadar devam etmekte ve bazen takip ettiği nesneyi takibe devam etmekte bazense öbür nesneyi takip etmeye başlamaktadır. Bu durum gözlemlerimize göre tam belirgin değildir ve geliştirilmesi gereken noktalardan birisidir. Ayrıca algoritmada da bazı geliştirilmesi gereken noktalar olduğu açıktır. Kararsızlığa düştüğü anda algoritmaya kalman filtresi gibi bazı takip algoritmaları eklenerek takip ettiği cismin potansiyel olabileceği yerler hesaplatılabilir ve ona göre takip ettirilebilir. Bununla beraber yapay zekânın tanıma algoritmasına ek bir etiket tanıyabilme kabiliyeti eklenerek. İnsan-Ahmet veya İnsan-1, İnsan-2 gibi tanımlama ile gerçekte hangi nesneyi takip etmesi gerektiği tanımlanabilir. Hatta çektiği resimlerden eğer elimizden gerçekten çok hızlı bir bilgisayar yardımıyla gerçek zamanlı eğitim yaptırılarak gördüğü cismi tam olarak diğer cisimlerden ayırt edebilir hale getirilebilir. Yine aynı şekilde kararsızlığa düştüğü anda belirsiz hareketler engellenebilir veya belirli bir algoritma izleyebilir hale getirilebilir. Hareket algoritması adım motora göre ayarlanmıştır. Bu algoritma mevcut konumla gidilecek konum arasında bir oran-orantı yaparak adım miktarını bulur ve o miktar kadar adım atar ve bu süre içerisinde cismin hareketlerine adapte olamaz. O adım miktarı sonunda bilgisayardan gelen yeni yön ve adım miktarını alabilir. Bunu engellemek için adım miktarı hep 1 olarak belirlenmiş fakat bu seferde yapılan gözlemlerde hız olarak çok yetersiz olduğu görülmüştür. Bu sebeple tekrar hız kazanmak için adım miktarı oran-orantı

ile belirlenen hale getirildi. Bu algoritmada cismin hareketlerinden adım miktarı tamamlanıncaya kadar sistemi duyarsız hale getirmektedir. Adım motorunun bir başka dezavantajı da vardır. Adım motorları yapıları gereği fren tertibatına sahip değillerdir ve frenleme olayı manyetik olarak sağlanır. Eğer moment (hız $*$ ağırlık) belirli bir değeri geçerse adım motoru frenleme yapamaz ve ek olarak bir kaç adım daha atar. Sebeple makineye ince ayarlar yapılırken hız istenilerek belirli bir değerin üzerine çıkmaması sağlanmıştır. Görüldüğü gibi makinede geliştirilmesi gereken bileşen zayıflıklarda bulunmaktadır. Takip servo motorlar ile yapılabilir. Bu adım motorunun bahsedilen negatif etkilerinden kurtularak çok hızlı şekilde takip yapılabilir. Servo motorunun kullanılmamasının sebebi maliyet olarak çok fazla olmasıdır. Adım motorlar hızlı olması için direkt olarak sürülmüştür. Servo motorlar yapısı gereği çok hızlı olduklarından redüktör ile kullanılabilirler ve böylece makine hem çok hızlı hem de çok hassas bir takip yapabilir hale getirilebilir. Yapısal bir başka eksiklikte kameradan kaynaklanmaktadır. Sistemde kullanan kamera kaliteli olarak nitelendirilebilecek bir web kameradır. Fakat bu işe çok uygun değildir. Büyütme kabiliyeti kontrol edilebilecek bir kamera yardımıyla algoritmanın daha uzak nesneleri kontrol etmesi sağlanabilir. Hatta bir kaç kamera kullanılarak uzak ve yakındaki nesnelerin takibi hassas şekilde yapılabilir. Ayrıca gövdesi ve bağlantı şekilleri alüminyum ayaklar yüzünden hareket sırasında çok rijit değildir bu sebeple yeni malzemeler ve dizaynla daha rijit bir yapıya kavuşturularak geliştirilebilir. Yine de test sonuçlarından açıkça görülmektedir ki makine 15 metreden yakın ve hızı 8 km/saat hızından düşük nesnelerde çok başarılı isabet sonuçları vermektedir.

Sistem çok çeşitli alanlarda fonksiyonel olarak kullanılabilir. Sistem genel olarak takip ve tespit yaptığından dolayı okullarda, sınır güvenliğinde, hava alanlarında, mitinglerde, adliye saraylarında vb. gibi güvenlik ihtiyacının yüksek olduğu yerlerde kullanılabilir. Ayrıca müşteri takibi, işçilerin fabrika içerisindeki takibi ve bir müesseseye giren kişilerin takibi konularda da kullanılabilir. Bazı eklemeler yapılarak meyve toplama veya fabrikalarda kalitesiz parçaların ayıklanması gibi konularda da potansiyel vadetmektedir. Yine fabrikalarda üretime ürünün yığın parçalardan alınarak konveyör bantlarına konulması gibi işlerde de kullanılabilir. Bir cismin bir yerden alınıp belirli bir başka yere götürülmesi

gibi işlerde bazı eklemeler yapılarak kullanılabilir. Yine bazı ayarlar yapılarak futbol maçları veya filmlerde bazı sahnelerin çekiminde kullanılabilir.

Sonuç olarak yapılan bu çalışma bahsedildiği üzere bir çok konuda potansiyel vadetmekle birlikte birçok sebepten dolayı bazı eksiklikleri de bulunmaktadır. Bundan sonra makinedeki bu eksiklikleri giderilmesi yanında makinenin sabit durması yerine harekette ederek cisimleri takip etmesi planlanmaktadır. Ayrıca makine kendi kendine tek bir ünite değildir. Yani dışarıdan bir elektrik bağlantısına ihtiyaç duymaktadır. Ve bir dizüstü bilgisayar vasıtası ile kullanılmaktadır. Bu bileşenler değiştirilerek makinenin kendi başına duran ve hareket edebilen bir yapıya kavuşturulması bir sonraki çalışmada amaçlanmaktadır.

KAYNAKLAR

Barlow, H. B. 1989. Unsupervised learning, Neural computation 1.3, 295–311.

Byrd, R. H., Chin, G. M., Nocedal, J. ve Wu, Y. 2012. Sample Size Selection in Optimization Methods for Machine Learning, Mathematical Programming 134.1, 127–155.

Chapelle, O., Scholkopf, B. ve Zien, A. 2009. Semi-supervised learning (chapelle, o. etal., eds.; 2006), IEEE Transactions on Neural Networks 20.3, 542–542.

Dai, J., Li, Y., He, K. ve Sun, J. 2016. R-Fcn: Object detection via region-based fullyconvolutional networks, Advances in neural information processing systems, 379–387.

Girosi, F., Jones, M. ve Poggio, T. 1995. Regularization theory and neural networksarchitectures, Neural computation 7.2, 219–269.

Girshick, R. 2015. Fast R-CNN, Proceedings of the IEEE international conference oncomputer vision, 1440–1448.

Glorot, X. ve Bengio, Y. 2010. Understanding the difficulty of training deep feed forward neural networks, Proceedings of the thirteenth international conference on artificial intelligence and statistics, 249–256.

Hawkins, D. M. 2004. The problem of overfitting, Journal of Chemical Information and Computer Sciences 44.1, 1–12.

Hinton, G. E., Krizhevsky, A., Sutskever, I. ve Srivastva, N. 2016. System and method for addressing overfitting in a neural network. US Patent 9,406,017.

Ioffe, S. ve Szegedy, C. 2015. Batch normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv preprint arXiv:1502.03167.

Kaelbling, L. P., Littman, M. L. ve Moore, A. W. (1996), Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4, 237–285.

Kingma, D. P. ve Ba, J. 2014. Adam: A method for stochastic optimization, arXivpreprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I. ve Hinton, G. E. 2012. Imagenet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems, 1097–1105.

Liang, P ve Bose, N. 1996. Neural network fundamentals with graphs, algorithms andapplications, Mac Graw-Hill.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. ve Berg, A. C. 2016 Ssd: Single Shot Multibox Detector, European conference on computer vision. Springer, 21–37.

McCarthy, J., Minsky, M. L., Rochester, N. ve Shannon, C. E. 2006. A proposal forthe dartmouth summer research project on artificial intelligence, august 31, 1955, AImagazine27.4, 12–12.

Nair, V. ve Hinton, G. E. 2010. Rectified linear units improve restricted boltzmannmachines, Proceedings of the 27th international conference on machine learning(ICML-10), 807–814.

Nwankpa, C., Ijomah, W., Gachagan, A. ve Marshall, S. 2018. Activation functions: Comparison of Trends in Practice and Research for Deep Learning, arXiv preprintarXiv:1811.03378.

Oquab, M., Bottou, L., Laptev, I. ve Sivic, J. 2015. Is object localization for free?-weakly-supervised learning with convolutional neural networks,Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 685–694.

Pascanu, R., Mikolov, T. ve Bengio, Y. 2012. Understanding the Exploding Gradient Problem, CoRR, abs/1211.50632.

Perez, L. ve Wang, J. 2017. The effectiveness of data augmentation in image classification using deep learning, arXiv preprint arXiv:1712.04621.

Redmon, J., Divvala, S., Girshick, R. ve Farhadi, A. 2016. You Only Look Once: Unified, real-time object detection, Proceedings of the IEEE conference on computer visionand pattern recognition, 779–788.

Ren, S., He, K., Girshick, R. ve Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks, Advances in Neural Information Processing Systems, 91–99.

Rosenblatt, F. 1958. The Perceptron: a probabilistic model for information storage andorganization in the brain, Psychological review 65.6, 386.

Ruder, S. 2016. An Overview Of Gradient Descent Optimization Algorithms, arXiv preprintarXiv:1609.04747.

Rumelhart, D. E., Durbin, R., Golden, R. ve Chauvin, Y. 1995. Backpropagation: The basic theory, Backpropagation: Theory, architectures and applications, 1–34.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. ve Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research15.1, 1929–1958.

Tian, Y., Ma, J., Gong, Q., Sengupta, S., Chen, Z., Pinkerton, J. ve Zitnick, C. L.2019. Elf opengo: An Analysis and Open Reimplementation of Alpha-zero, arXiv preprintarXiv:1902.04522.

Vincent, O. R., Folorunso, O., ve ark. 2009. A descriptive algorithm for sobel image edge detection, Proceedings of Informing Science & IT Education Conference (InSITE).Vol. 40. Informing Science Institute California, 97–107.

ÖZGEÇMİŞ

Adı Soyadı	: Bilen BAŞARIR
Doğum Yeri ve Tarihi	: Ankara, 1975
Yabancı Dil	: İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise	: Eşrefpaşa Lisesi, 1992
Lisans	: Uludağ Universitesi Makine Mühendisliği Bölümü, 1999

İletişim : bilenbasarir@gmail.com

Yayınlar :