**T.C.**

**YOZGAT BOZOK ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**Yüksek Lisans Tezi**

# DERİN ÖĞRENME İLE ARAÇ PLAKA LOKASYONUNUN BELİRLENMESİ

**Talip ÇAY**

**Tez Danışmanı**

**Doç. Dr. Orhan ER**

**Yozgat 2019**

**T.C.**

**YOZGAT BOZOK ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI**

**Yüksek Lisans Tezi**

# DERİN ÖĞRENME İLE ARAÇ PLAKA LOKASYONUNUN BELİRLENMESİ

**Talip ÇAY**

**Tez Danışmanı**

**Doç. Dr. Orhan ER**

**Yozgat 2019**

# YOZGAT BOZOK ÜNİVERSİTESİ

# TEZ ONAY FORMU

T.C.
YOZGAT BOZOK ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ

Enstitümüzün Mekatronik Mühendisliği Anabilim Dalı Tezli Yüksek Lisans Programı 70111710005 numaralı öğrencisi Talip ÇAY 'ın hazırladığı "**Derin Öğrenme İle Araç Plaka Lokasyonunun Belirlenmesi**" başlıklı tezi ile ilgili tez savunma sınavı, Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri gereğince 21/06/2019 Cuma günü saat 15:00'da yapılmış, tezin onayına oy birliği/oy çokluğu ile karar verilmiştir.

**Danışman**
**Doç. Dr. Orhan ER** : .....

**Jüri Üyesi**
**Dr. Öğr. Üyesi Volkan AKDOĞAN** : .....

**Jüri Üyesi**
**Dr. Öğr. Üyesi Ahmet Nusret TOPRAK** : .....

**ONAY:**

Bu tezin kabulü, Enstitü Yönetim Kurulu'nun 04../07.../19. tarih ve 31.. sayılı Enstitü Yönetim Kurulu Kararı ile onaylanmıştır.

04.07.2019

**Müdür**

KYT-FRM-110/00

# İÇİNDEKİLER

**DERİN ÖĞRENME İLE ARAÇ PLAKA LOKASYONUNUN BELİRLENMESİ**

**Talip ÇAY**

**Bozok Üniversitesi**
**Fen Bilimleri Enstitüsü**
**Mekatronik Mühendisliği Anabilim Dalı**
**Yüksek Lisans Tezi**

**2019; Sayfa: 78**

**Tez Danışmanı: Doç. Dr. Orhan ER**

# ÖZET

Bu çalışmada, derin öğrenme yöntemlerini kullanarak araç plaka tanıma sistemi tasarlanmıştır. Tez çalışması iki aşamadan oluşmaktadır. Birinci aşamada evrişimli sinir ağı kullanarak plaka konumu tespit edilmiştir. İkinci aşamada ise tespit edilen plaka görüntü işleme teknikleriyle okuma işlemi gerçekleşmiştir. İlk aşama için sinir ağı eğitimi yapılmıştır. R-CNN eğitimi için 450 adet araç plaka görüntüsü kullanılmıştır. Ve modelin test işlemleri 100 adet araç plaka görüntüsü kullanılmıştır. İkinci aşamada tespit edilen plaka bölgesi üzerinden okuma işlemi gerçekleştirilmiştir. Okuma işlemi sırasında plaka görüntüsü gürültülerinden arındırıldıktan sonra karakterlerine ayrıştırılmıştır. Harf ve rakamlardan oluşan veri setimiz ile tespit edilen karakterler korelasyon işlemine tabi tutularak okunmuştur.

**Anahtar Kelimeler:** Araç Plaka Tanıma, Yapay Sinir Ağları, Derin Öğrenme, R-CNN, CNN

**DETERMINATION OF LICENSE PLATE LOCATION WITH DEEP LEARNING**

**Talip ÇAY**

**Bozok University**
**Graduate School of Natural and Applied Sciences**
**Department Of Mechatronics Engineering**
**Master of Science Thesis**

**2019; Page: 78**

**Thesis Supervisor: Assoc. Prof. Dr. Orhan ER**

# ABSTRACT

In this study we implemented license plate recognition using CNN. The study applied in two main steps. In the first step we detected license plate position with R-CNN. In the second step license plate reading operation was realized from detected license plate. License plate reading operation was performed with image processing. We used 450 license plate images for training R-CNN. For testing R-CNN model we used 100 license plates to images. In the second step we performed license plate reading from detected license plate. During the reading process noise of plate images removed and the image separated into characters.

**Keywords:** Vehicle License Plate Recognition, Artificial Neural Networks, Deep Learning, R-CNN, CNN

# TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde değerli bilgilerini, kıymetli zamanlarını benimle paylaşan tez danışmanım Doç. Dr. Orhan ER hocama ve Şube Müdürüm Emre ÖLMEZ'e teşekkürü borç bilirim.

Ayrıca beni bu günlere getiren haklarını hiçbir zaman ödeyemeyeceğim aileme, kardeşlerime ve gösterdikleri anlayıştan dolayı sevgili eşime, dünyalar tatlısı kızlarım Miray Begüm ve Elif Eylül'e sonsuz teşekkürler.

Talip ÇAY
YOZGAT 2019

# ŞEKİLLER LİSTESİ

# KISALTMALAR LİSTESİ

| | | |
|---|---|---|
| **SWM** | : | Destek Vektör Makineleri (Support Vector Machines) |
| **ReLU** | : | Rectified Linear Units |
| **CNN** | : | Evrişimli Sinir Ağları (Convolutional Neural Networks) |
| **R-CNN** | : | Bölgesel-Evrişimli Sinir Ağları |
| **RoI** | : | Region Of Interest |
| **YSA** | : | Yapay Sinir Ağları |
| **RGB** | : | Red – Green – Blue |
| **F (x,y)** | : | İki Boyutlu Resim matrisi |
| **CPU** | : | Merkezi İşlem Birimi |
| **GPU** | : | Grafik İşlem Birimi |
| **OCR** | : | Optical Character Recognition |
| **GAN** | : | Generative Adversarial Networks |

# 1. GİRİŞ

Yapay zekâ, insan davranışlarının makine tarafından taklit edilmesini sağlamaktadır. Bu sebeple otonom sistemlerin geliştirilmesinden yapay zekâ teknolojilerine ihtiyaç duyulmaktadır. (Fürnkranz, 1999). Yapay zekâda karar verme veya tahmin oluşturma süreçleri makine öğrenmesi ile gerçekleşmektedir. Makine öğrenmesi gözetimli ve gözetimsiz yöntemleri kullanarak sınıflandırma ve kümeleme işlemlerini sağlar. Makine öğrenmesinden insan faktörün devre dışı bırakılarak, karmaşık verilerin hızlı öğrenilmesi ve uygulanmasında derin öğrenme yöntemleri kullanılmaktadır (Le, Bengio, Hinton, 2015). Dünyadaki teknolojik gelişmeler ve hedefler doğrultusunda ülkemizde derin öğrenme uygulamaları artmaktadır. Gerçekleştirilen çalışmanın birinci bölümünde derin öğrenmenin tanımı, tarihçesi ve yapılan çalışmalar hakkında ilgili bilgiler verilmiştir. İkinci bölümde yapay sinir ağları, üçüncü bölümde optimizasyon ayarları, dördüncü bölümde evrişimli sinir ağları incelenmiştir. Son bölümde yapılan çalışma anlatılmış ve elde edilen başarı sonuçları verilmiştir.

## 1.1. Derin Öğrenme Nedir?

Derin öğrenme; bilgisayarların, deneyimlerden öğrenmelerini ve dünyayı kavramların hiyerarşisi açısından anlamalarını sağlayan bir makine öğrenimi olarak tanımlamıştır. (Gu, Zhang, Kim, 2016) Başka bir kaynakta derin öğrenme; denetimli veya denetimsiz özellik çıkarma, dönüştürme, desen analizi ve sınıflandırma için birçok doğrusal olmayan gizli katmandan yararlanan bir makine öğrenme teknikleri sınıfı olarak tanımlamıştır (Deng, Yu, 2014). Yine derin öğrenme; insan beyninin son derece karmaşık problemler için gözlemleme, analiz etme, öğrenme ve karar verme yeteneğini taklit etmeyi amaçlayan, büyük miktarda denetimsiz veri kullanan bir makine öğrenmesi olarak tanımlanmıştır (Najafabadi, Villanustre, Khoshgoftaar, Seliya, Wald, Muharemagic, 2015). Derin öğrenme konusunda farklı kaynaklardan alınan tanımlar bileştirilecek olursa; derin öğrenme insan beyninin karmaşık problemler için gözlemleme, analiz etme, öğrenme ve karar verme gibi yeteneklerini taklit eden, denetimli veya denetimsiz olarak özellik çıkarma, dönüştürme ve sınıflandırma gibi işlemleri büyük miktarlardaki verilerden yararlanarak yapabilen bir makine öğrenmesi tekniğidir.

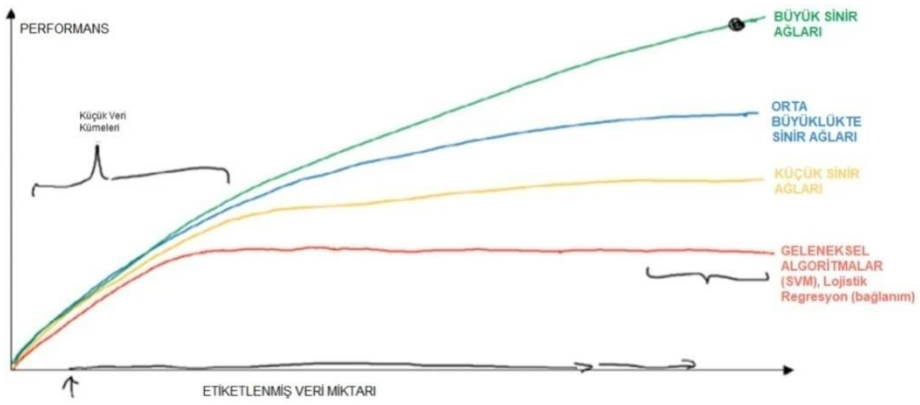
## 1.2. Tarihçe

Derin öğrenme makine öğrenmesi ve yapay sinir ağları gibi araştırmalar 1950 li yıllarda ilk kez ortaya çıkmıştır ve o yıllarda çalışmalar başlamıştır. 1950 yılında Mind dergisinde "Computing Machinery and Intelligence" makalesiyle dönemin bilgi kuramı uzmanı ve kriptoloji üzerine çalışmış olan Alan Turing "Makineler Düşünebilir mi" sorusuyla aslında bu çarpıcı konunun öncülerinden biri olmuştur. O yıllardan bu yıllara değişen en önemli şey makinelerin daha doğrusu bilgisayarların veri üretebilen ve tutabilen cihazların gelişmiş olmasıdır. Ve tüm bu verilerin fiziksel olarak değil artık internet üzerinden bunların hem tutulabilir hem transfer edilebilir olması önem taşımaktadır. 1950'li yıllarda Turing makineler düşünebilir mi dedikten sonra 1955 yılında yapay zekâ terimini ilk kez Jhon McCharthy kullandı. 1959 yılında ilk yapay zekâ laboratuvarı Marvin Lee Mainsky tarafından Massachusetts Teknoloji Enstitüsü'nde (MIT) kurulmuştur. 1957 yılında Frank Rosenblatt yapay sinir ağı modellerinin temelini oluşturan perseptron (algılayıcı) tanımı yapmış oldu. 1969 yılında Mainsky XOR probleminin tek katmanlı bir ağ yapısıyla çözülemeyeceğini ortaya koyarak aslında bugün çok katmanlı olarak adlandırdığımız derin öğrenmeyi de kapsayan yapay sinir ağlarının temelini atmış oldu. 1986 yılında sinir ağlarının matematiksel modelde edindiği bilgiyle yeni bilgileri, tepkileri, kararları güncelleyebildiği daha insansı hale gelmesini sağlayan geriye yayılım algoritması (backpropagation algoritm) ilk kez Geoffrey Hinton tarafından ortaya atıldı. Ardından 1997 yılında IBM'in Deep Blue yazılımı Dünya satranç şampiyonu olan Garry Kasparov'u yenince yapay zekâ konusu insanlara daha ilginç gelmeye başladı ve laboratuvarlardan dışarı çıkmış oldu. 1998 yılında şu an derin öğrenmenin ve bilgisayarlı görünün temeli olan evrişimli sinir ağı (CNN) modeliyle rakamların sınıflandırma işlemi şu an facebook yapay zekâ direktörü Yann LeCun tarafından gerçekleştirilmiştir. Google cloud'un yapay zekâ direktör olan, Stanford Üniversitesi profesörü Fei Li ve ekibi 167 ülkenin işbirliğiyle hazırlamış olduğu binlerce kategoriden oluşan milyonlarca görüntü etiketlenmiş ve "imagenet" ismiyle ücretsiz bir şekilde herkese açılmıştır. Ve 2010'lu yıllara kadar daha hızlı işlem yapan CPU ve GPU gibi donanımların gelişmesiyle yapay zekâ kaldığı yerden gelişimine tekrar devam etmeye başlamıştır. 2011 yılında "siri" sesli asistanıyla cep telefonuna girdi ve herkesin konuştuğu konu oldu. 2012 yılında, 1998 yılındaki LeNet modeli ImageNet veri setiyle Toronto Üniversitesinden Alex Krizhevskyi, Ilya Sutskever ve

Geoffrey Hinton'un oluşturduğu CNN (evrişimli sinir ağı) modeli AlexNet ismiyle ortaya atılmıştır. Bir önceki çalışmalara göre iki katı kadar daha yüksek başarı elde ettiğinde derin öğrenme popüler olmaya başladı. 2014 yılında Google; görüntü işleme, bilgisayarlı görü, nesne sınıflandırma ve takip etme gibi işlemler gerçekleştiren modeli olan GoogLeNet ile daha derin hale gelerek yüksek başarılar elde ettiler. 2014 yılında GAN çekişmeli ağlar Yoshua Bengio ve Ian Goodfellow tarafından ortaya atıldı. Bu yöntemle sentetik veri üretimi, gerçekte olmayan insan yüzü üretimi gibi ilginç çalışmalar gerçekleştirilmiştir. 2016 yılında Microsoft'un geliştirdiği TAY chat botu sürekli öğrenebilen ve insanlarla iletişim halindeyken öğrenmeye devam eden ve bu duruma göre cevap üreten yazılım geliştirdi. 2017 yılında Google Deep Mind şirketi Alphago yazılımı Go oyununda Dünya şampiyonunu farklı yenince yapay zekâ çok konuşulur bir konu haline geldi ve şu an gelişimine hızlı bir şekilde devam etmektedir. [1]

### 1.3. Neden Yapay Sinir Ağlarında Derin Sinir Ağları Popüler Oldu?

Dünya'daki verilerin yüzde doksanı son birkaç yılda üretilmiştir. Özellikle düzenli ve etiketlenmiş kıymetli veri kümesi sınırlı olduğunda şekil 1.1.'de görüldüğü gibi geleneksel yöntemlerle derin sinir ağları arasında performans farkı çok fazla değildir. Eldeki veri az ise hangi yöntem kullanılırsa kullanılsın benzer şekilde sonuçlar elde edilecektir. Böyle durumlarda derin sinir ağlarının kullanılması işlem yükünden dolayı uygun değildir. Fakat etiketlenmiş veri arttığında derin sinir ağlarını kullanmak daha uygundur. Çünkü bu şekilde veri artıkça ağın derinleşmesi fayda sağlayacaktır. Bu yüzden derin sinir ağlarının yüksek başarı elde etmesi son yıllarda popüler olmasının sağlamıştır.




**Şekil 1.1.** Sinir Ağlarının Veri Seti Miktarına Göre Performansı [17]

### 1.4. Literatür Taraması

Literatürde yapay sinir ağlarıyla ve görüntü işleme teknikleriyle yapılmış plaka tanıma sistemleri mevcuttur. Bu uygulamaların bir kısmı aşağıda sıralanmıştır.

Christian Gerber ve Mokdong Chung'ın 2016 yılında yayınladıkları makalede evrişimli sinir ağı kullanarak mobil cihazlar için araç plaka tanıma sistemi geliştirmişlerdir. 88 resim üzerinden test sonuçlarında % 91,25 başarı elde etmişlerdir. [39]

M. M. Shaifur Rahman, çalışmasında evrişimli sinir ağı kullanarak plaka karakter tanıma işlemi gerçekleştirmiştir. 1725 karakter ile evrişimli sinir ağını eğitmiştir. İterasyon (epoch) sayısını değiştirerek nihai hata değerini gözlemlemiştir. [40]

Kai Tang, 2018 yılında klasik görüntü işleme teknikleriyle ve derin öğrenme modellerinden olan R-CNN, Fast R-CNN ile plaka tanıma işlemi gerçekleştirmiş ve elde edilen sonuçları karşılaştırmıştır. [41]

Lele Xie, Tasweer Ahmad ve Lianwen Jin 2018 yılında yayınladıkları makalede derin öğrenme modellerinden Faster R-CNN ve YOLO ile plaka tanıma işlemi gerçekleştirmişlerdir. Eğitim için 2049 görüntü kullanılmıştır. Faster R-CNN ile % 85 başarı, YOLO ile % 97 başarı elde edilmiştir. [42]

Emre ÇAMAŞIRCIOĞLU, YSA ve görüntü işleme teknikleri kullanarak araç plaka tanıma sistemi geliştirmiştir. Bu uygulamada plaka konumu klasik YSA kullanılarak bulunmuş, eğik düzensiz görseller için radon dönüşümü kullanılmıştır. Karakter tanıma işleminde aktivasyon fonksiyonu olarak sigmoid fonksiyonu kullanılmıştır. [2]

İsmail IRMAKÇI, çalışmasında öncelikle görüntüye önişlem uygulamış, görüntüden plaka çıkarımı için morfolojik görüntü işleme teknikleri uygulanarak köşelerin kapladığı en küçük dörtgen bölge görüntüden çıkarılmıştır. Çıkarılan bölgeye tekrar önişlemler uygulandıktan sonra bölge tekrar karakter ayrıştırma sisteminde yatay ve düşey taramaya sokulmuştur. Son olarak karakterlerin tanımasında yapay sinir ağı kullanılarak plakaların ASCII kodu tespit edilmiştir. [3]

Naga Surya Sandeep Angara, yapılan çalışmasında derin öğrenme yöntemleri kullanılmıştır. Kesilmiş plaka sinir ağına sokularak karakterleri bulunmuştur. Bulunan karakterler görüntü işleme teknikleri ile tanınmıştır. Sinir ağının eğitimi sırasında aktivasyon fonksiyonu olarak ReLU fonksiyonu kullanılmıştır. Ağ Eğitimi için 9324 veri, test için 2318 veri kullanılmıştır. [4]

Hogne Jongensen, çalışmasında R-CNN, CNN, ve YOLO ile araç plaka konumunu belirlemiş ve % 98,9 başarı elde etmiştir. Ağ eğitimi için 30 adet katmanla yapılmış ve eğitim için 410 adet, test için 100 araç plaka görüntüsü kullanmıştır. Aktivasyon fonksiyonu ReLU fonksiyonuyla çalışmıştır. [5]

Ali BAKKALOĞLU, yapılan çalışmasında görüntü işlem teknikleri kullanarak hareketli araçlardan alınan 640x480 boyutundaki görüntüler üzerinde araç plaka tanıma işlemi yapmıştır. Test için 1500 görüntü kullanılmıştır. Bazı kenar bulma filtreleri kullanılmış ve OTSU'nun metodu ile gri tonlu görüntüden siyah beyaz görüntüye dönüşüm için eşik değeri hesaplamaları yapılmıştır. [10]

### 1.5. Plaka Tanıma Sistemleri ve Derin Öğrenme

Otomatik plaka tanıma sistemleri problemi birçok varyasyon içeren karmaşık bir iştir. Görüntüler renkli, siyah beyaz veya kızılötesi kamerayla çekilebilir. Ayrıca, uluslar plakalar için farklı arka plan renklerine veya plaka üzerinde dikkat dağıtıcı desenlere izin veren farklı standartlara sahiptir. Harflerin sayısı, sayılar, boşluklar, konumları ve kullanılan yazı tipi uluslararasında farklılık gösteren diğer değişkenlerdir.

Bu projenin kapsamı Türkiye Cumhuriyeti yollarında görünen plakaları tespit edebilmektir. Türkiye Cumhuriyetine ait plaka örneği Şekil 1.2'de gösterilmiştir.


**Şekil 1.2.** TR Plaka Örneği

Plaka tanıma işlemi yapılırken ortaya bazı zorluklar çıkmaktadır. En önemli neden sistemin eğitilmesi için yeterli veri olmamasıdır. Ve tanıma işlemi yapılırken çeşitli fiziksel engellere neden olan varyasyonlar vardır.

1. Karakter kalıbı: Türkiye Cumhuriyeti plakaları için standart kalıp

YY_XXX_YYY

X bir harf, Y bir sayıdır ve _ bir boşluktur. İlk iki sayılar il plaka kodlarıdır. Diğer harf ve rakamlar karakter sayısı olarak farklılık gösterebilir. Bununla birlikte, römorklar, motosikletler ve diğer bazı özel araçlar için plakalar bu modelden farklıdır.

2. Arka plan rengi olarak en çok beyaz renk kullanılır. Ancak, özel araçların başka arka plan renkleri olabilir.

Plakanın görüntüsü mükemmel şekilde yakalanmadığından, plakayla ilgili bir dizi başka değişiklik görülebilir. Bunlar aşağıda listelenmiştir.

Konum, plaka görüntünün farklı kısımlarında yer alabilir.

Boyut, plakanın tam çekilen resimle karşılaştırıldığında boyutu, kamera mesafesine ve yakınlaştırmaya bağlı olarak değişir.

Rotasyon, plaka görüntünün çekildiği açıya bağlı olarak eğilebilir

Plaka kir veya kar gibi ek gürültüler tarafından gizlenebilir.

Bulanıklaştırma veya düşük kamera ekipmanı nedeniyle görüntü kalitesi düşük olabilir.

Bir resim hiç, bir veya daha fazla plaka yakalamış olabilir.

Plaka vida, çerçeve ve çıkartma içerebilir. Türkiye Cumhuriyeti araç plakalarında, yıllık araç muayene bandrolleri bulunmaktadır.

Son olarak, aşağıdaki liste tarafından verilen bir dizi çevresel değişiklik de zor olabilir.

Çekilen görüntünün arka planı, plakalara benzer desenleri içerebilir. Görüntüdeki herhangi bir yerdeki metin ve metin benzeri desenler plaka ile karıştırılabilir.

Aydınlık, çevresel ışık, kamera ışığı veya araç ışığına göre değişebilir.

# 2. YAPAY SİNİR AĞLARI

Bu bölümde tek katmanlı ve çok katmanlı sinir ağlarının yapısından, işleyişinden, geriye yayılım algoritması hakkında teorik bilgiler verilecektir.

## 2.1. Yapay Sinir Ağlarının Kullanım Alanları

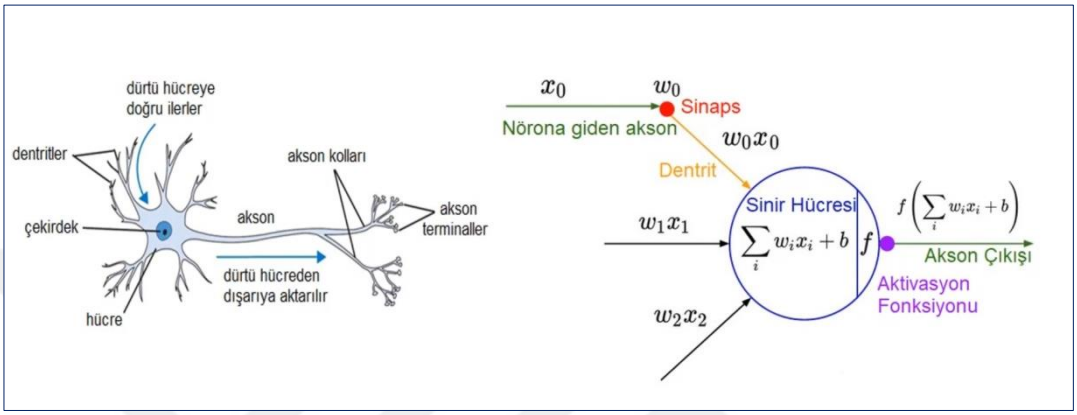
Yapay sinir ağları temel olarak kullanım alanları şunlardır; Finansal alanlarda, kredi skorlamasında, görüntü işlemede, yüz tanıma hareket algılamada, nesne tanımada, tıp alanında, enerji üretiminde fiyat yük tahmini, otomotiv, havacılık ve üretim alanında ve doğal dil işlemede kullanılmaktadır.


**Şekil 2.1.** Makine Öğrenme Çeşitleri [17]

## 2.2. Yapay Sinir Ağlarının Biyolojik Temelleri

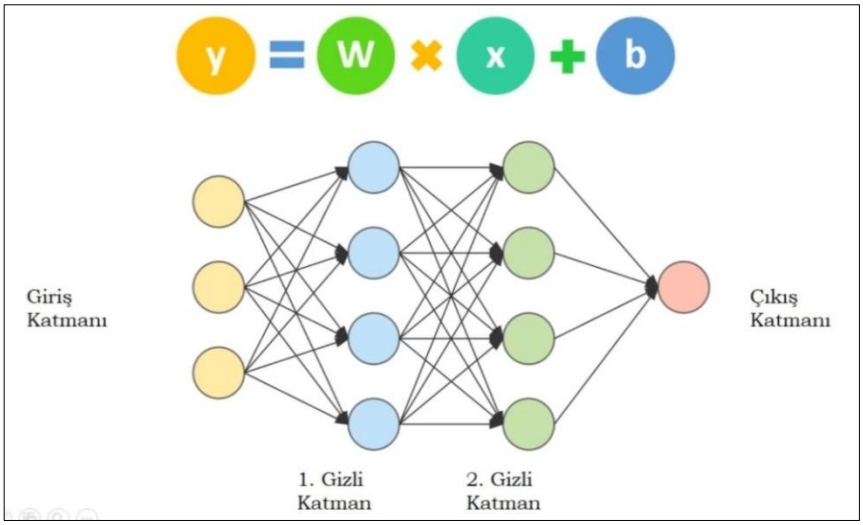
Canlıların davranışlarını inceleyip, matematiksel olarak modelleyerek benzer yapay modellerin üretilmesine sibernetik denir. Aslında varılmak istenen nokta eğitilebilir, adapte olan, kendi kendine organize olup öğrenebilen, değerlendirme yapabilen sinir ağları ile insan beyninin öğrenme ve uygulama yapısını modellemeye çalışmaktır. Bir işi bir bilgisayarda gerçekleştirmek için o işin algoritmasına hâkim olmak gerekir. Algoritma, girdileri çıktılara çevirmek için temel komut dizilerinin tamamıdır. Ancak bazı problemleri çözmek için bilinen bir algoritma olmayabilir. İstenen veya istenmeyen durumlarda zamanla değişiklik gösterebilir ya da kullanıcıya göre değişkenlik gösteren uygulamalar sabit bir algoritmaya sahip

olmayabilirler. Bilgimiz eksikse de verimiz bol olabilir. Kolayca hem istenen hem de istenmeyen binlerce örnekten sistemin öğrenmesini sağlayabiliriz. Bilgisayarların bir iş için kendi başına algoritmasını oluşturmasını isteyebiliriz. Günümüz teknolojisinde veri toplama cihazların sayısal olduğundan verilerin güvenli bir şekilde erişebilir, saklanabilir ve işlenebilir olması avantajımızadır. Canlılarda bulunan sinir hücresi ve sinir hücresinin matematiksel modeli aşağıda gösterilmiştir.




**Şekil 2.2.** Biyolojik Sinir Hücresi ve Matematiksel Modeli [15]

Şekil 2.2'de soldaki resimde görüldüğü gibi sinir hücresinde bir çekirdek bulunmaktadır. Akson boyunca iletim devam etmekte ve çıkış terminalleri (akson terminalleri) başka sinir hücrelerinin dentritlerine bağlanarak iletim tamamlanmış oluyor. Bu bağlantı noktalarına sinaps denilmektedir. Bunu matematiksel olarak modellemeye çalıştığımızda dentrit denilen yollar boyunca ağırlıklarımız mevcut, bu dentritlere giren bir başka nörondan da gelmiş olabilen giriş değeri var. Giriş değerimiz ve dentritdeki ağırlığımız çarpıldıktan sonra sinir hücresine iletiliyor. Diğerdentritlerden gelen giriş ve ağırlıkların çarpımları toplandıktan sonra bir bias değeri eklendikten sonra bir aktivasyon fonksiyonundan geçirilerek çıkışa aktarılıyor. Bu çıkış son çıkış olabilir veya başka bir hücrenin girişi olabilir.

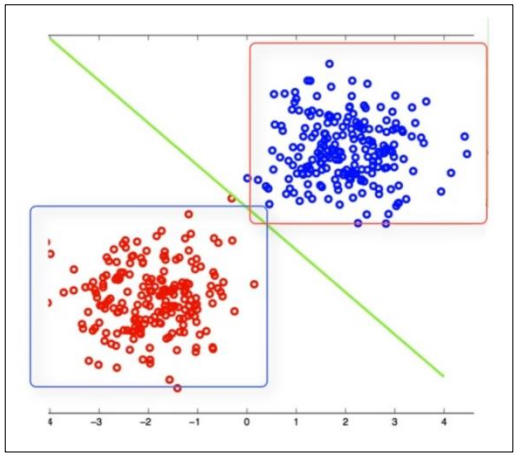



**Şekil 2.3.** Sinir Ağı Modeli [17]

Her bir sinir hücresi hesaplanır ve bunlar birbirlerine seri ve paralel bir şekilde bağlanır. Sinir ağı giriş katmanı, gizli katmanlar ve çıkış katmanlarından oluşur. Birden fazla gizli katmandan ve nörondan oluşuyorsa buna çok katmanlı sinir ağı diyoruz. Eğer tek bir katmandan oluşuyorsa buna tek katmanlı sinir ağı diyoruz. Her iki sinir ağının yetenekleri farklıdır.

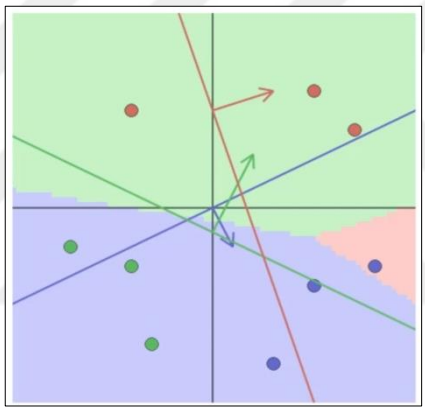
## 2.3. Tek Katmanlı Sinir Ağları

En basit yapay sinir ağı türüdür. Mantık devrelerinde kullandığımız kapılar gibi düşünülebilir. Girdisi ya doğrudan çıkışa veya bir sonraki mantık kapısına eklenebilir. İki giriş olur genellikle bir çıktısı olur. Girişler sinapslar boyunca uzanır ve ağırlıklarla çarpılarak bir eşik değerinden geçirilir. Aslında sınıfları ayıran biçimler için bir model ailesi seçmek gerekir. Viladimir Vapnik 1995'te sınıf dağılımlarını kestirmenin sınırları kestirmekten daha zor iş olduğunu ve kolay bir problemi çözmek için daha zor bir problemi çözmenin gereksiz olduğunu iddia etmiştir. Bu şu demektir şekil 2.4'deki örnekte görüldüğü gibi iki tane sınıfımız vardır bu sınıfların dış çevresini kestirmeye çalışmak yerine iki sınıfı birbirinden ayıran bir sınırı çizmeye çalışmak daha kolay ve doğrusal bir problem olarak tanımlamak daha doğrudur. Bunun geçerli olabilmesi için problemin sınırlarının kolayca belirleniyor olması gerekiyor. Şekil 2.4'de görüldüğü gibi kolaylıklar ayrılabilen iki sınıfı tek bir doğru ile kolaylıkla ayırabiliyoruz. Bu tek katmanlı sınır ağıyla gerçekleşebilir demektir.



**Şekil 2.4.** Sinir Ağı İle Sınıf Ayrımı Örneği [15]

Ya da birçok sınıf varsa tek katmanlı birkaç sinir ağı ile gerçekleştirilen diskriminant (ayırtaçlar) doğrusal diskriminant denklemleri bunları ayırabiliyoruz.



**Şekil 2.5.** Sinir Ağı İle Çoklu Sınıf Ayrımı [17]

$$g(x\backslash w_i, w_{i0}) = w_i^T + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0} \quad (1)$$

Diskriminantların doğrusal olduğunu ve x girdisine bağlı olduğunu varsayarsak hesaplayacağımız diskriminant ağırlıklar ve girişlere bağlı bir şekilde d kadar toplam yapılarak ağırlıklar w ve girdiler x şeklinde gösterilmiştir. Doğrusal diskriminant daha çok basitliği nedeniyle tercih edilen bir yöntemdir. Eğer $x_{ij}$ girdisinin j. ağırlığı büyükse o j. değer önemli olduğu anlamına gelir. Çıktımız en sonunda farklı özniteliklerin toplamı olarak yazılır. Bu modeli genelleştirmeye çalışırsak özetle $x^{T'}$nin (x vektörünün transpozu) w ile veya x ile çarpılmış halidir diyebiliriz. Birinci dereceden diskriminant yeterince esnek olmayabilir bunun için 2. veya daha yüksek dereceli diskriminantlar kullanılabiliriz. Fakat derecesi arttıkça daha büyük öğrenme kümesine ihtiyaç duyarız ve genel bir model haline gelir. Ama bu sefer ağımız daha

büyük öğrenme verisine ihtiyaç duyar. Eğer küçük öğrenme verisiyle yüksek dereceli diskriminantlarla tanımlamak için bir model kurmaya çalışırsak bu seferde ağ aşırı öğrenme (overfitting) riski ile karşı karşıya kalacaktır.. Diskriminantı Φ ile tanımlanacak olunursa genel diskriminant formülü ağırlıkların x'e bağlı diskriminantları şeklinde ifade edilir.

Girdiye çarpım terimleri de eklenebilir. Örneğin $x_1$ ve $x_2$girdisi için j değeri 1 ve 2 olarak alırsak $x_1$ ve $x_2$varsa bunu z'ler şeklinde tanımlamaya çalışalım. $z_1$=$x_1$,$z_2$=$x_2$,$z_3$=$x_1^2$, $z_4$=$x_2^2$, $z_5$=$x_1x_2$ olarak ifade edildiğinde z vektörü (z=[$z_1$,$z_2$,$z_3$,$z_4$,$z_5$]) beş boyutlu z uzayında yazacağımız doğrusal ayırtaç (diskriminant) iki boyutlu x uzayında doğrusal olmayan bir diskriminant olarak tanımlanacaktır. Ve aslında yaptığımız ilk uzayda doğrusal olmayan bir diskriminant ya da regresyon fonksiyonunu öğrenmek yerine doğrusal olmayan bir dönüşümle yeni bir uzaya geçip orada doğrusal bir fonksiyonlar tanımlanmaktadır. Yani doğrusal olmayan bir işlemi doğrusal bir fonksiyonla ayırmak mümkündür. Bunu genel gösterirsek ağırlıkların x'e bağlı diskriminant şeklinde açıklayabiliriz.

Genelleştirilirse;

$$g(x\backslash W_i w_i, w_{i0}) = x^T W_i x + w_i x + w_{io}$$

$$z_1=x_1, z_2=x_2, z_3=x_1^2, z_4=x_2^2, z_5=x_1x_2 \rightarrow z=[z_1,z_2,z_3,z_4,z_5] \quad (2)$$

Diskriminantı$\emptyset_{ij}$(x) ile gösterirsek;

$$g_i(x) = \sum_{j=1}^{k} w_j \emptyset_{ij}(x) \quad (3)$$

Doğrusal olarak ayrılan bilen iki sınıf varsa bunun için bir diskriminant denklemi kullanılması yeterlidir. İki sınıf için tanımlamış olan diskriminant denklemlerinden tek bir diskriminant denklemlerine geçiş yapılmaktadır. $g_1$ ve $g_2$ iki farklı ağırlık ve x girdilerine göre hesaplanan $w^T$ ve x birer vektör (vektörler birbirleri ile çarpılırken bir tanesinin transpozunun alınması gerekir) sonuç olarak elde edilen g(x) fonksiyonuna 0'dan büyük ve küçük olduğu durumlarda yani iki sınıf atıyoruz. Denkleme göre eğer g(x) ayırtacı 0'dan büyükse $c_1$ sınıfına, 0'dan küçükse veya eşitse $c_2$ sınıfına ait olmaktadır. Bunu yaparken y'yi yani diskriminant denklemi bir sigmoid fonksiyonundan geçirilmiştir.

$$g(x) = g_1(x) - g_2(x)$$

$=(w_1^T x+w_{10})-(w_2^T x+w_{20})$

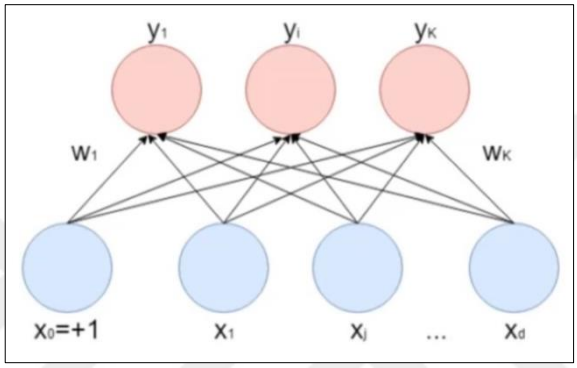
$=(w_1 x+w_2)^T x-(w_{10}+w_{20})$

$=w^T x+w_0 \leftarrow$ (Ayırtaç Denklemi) (4)

Seç $\begin{cases} c_1 \; eğer \; g(x)>0 \\ c_2 \qquad değilse \end{cases}$

y=sigmoid $(w^T x+w_0)$= (sigmoid(0)=0.5) → y > 0.5 ise $c_1$ (5)

Şekil 2.6'da basit bir ağı yapısı üzerinde x girdileri ve y çıktıları gösterilmektedir.




**Şekil 2.6.** Basit Sinir Ağı Modeli

Yukarıda bahsedilen y denklemi (çıkış denklemi) j'nin 1'den d'ye kadar değerleri için $w_j$ ağırlıkları ve $x_j$ girişlerinin çarpımları toplamı artı bias değeridir.

$$y=\sum_{j=1}^{d} w_j x_j + w_0 \quad (5)$$

Burada; w=$[w_1,w_2,\ldots,w_d]^T$ ve x=$[x_1,x_2,\ldots,x_d]^T$ değerleri birer vektördür.

Bunların bir tanesinin transpozu ile diğerini çarparak y elde edilmektedir. Ve $w_0$ bias değeriyle topluyoruz.

$y=w^T x$, d=1 tek $x$ girdisi için $y=wx+w_0$ (6)

Bu işlemi yaparken ağırlıkları en başta rastgele olarak tanımlamak uygun olabilir. Daha önceden ağırlıklar varsa onları da kullanılabilir. Diskriminant denklemini oluşturup çıkış ağırlıkları hesaplanarak sistem yakınsayana dek bu işlem tekrar edilmektedir. Buna algılayıcının eğitilmesi işlemi denilir. Eğitim işlemini modeller nasıl eğitilir başlığı altında daha detaylı incelenecektir. Temel olarak burada sonuca

nasıl erişildiğini anlatabilmek için istenen çıktı ile gerçek çıktının farkı (cost fonksiyonu) öğrenme oranı ile çarpılır ve sonuca güncelleme işlemi yapılır.

Tüm i = 1,…,K

Tüm j = 0,…,d

$w_{ij} \leftarrow$ rand (-0.01,0.01)

Yinele

Tüm $(x^t, r^t) \in$ x rastgele sırala

Tüm i = 1,…,K

$o_i \leftarrow 0$

Tüm j = 1,…,d

$o_i \leftarrow o_i + w_{ij} x_i^t$

Tüm i = 1,…,K

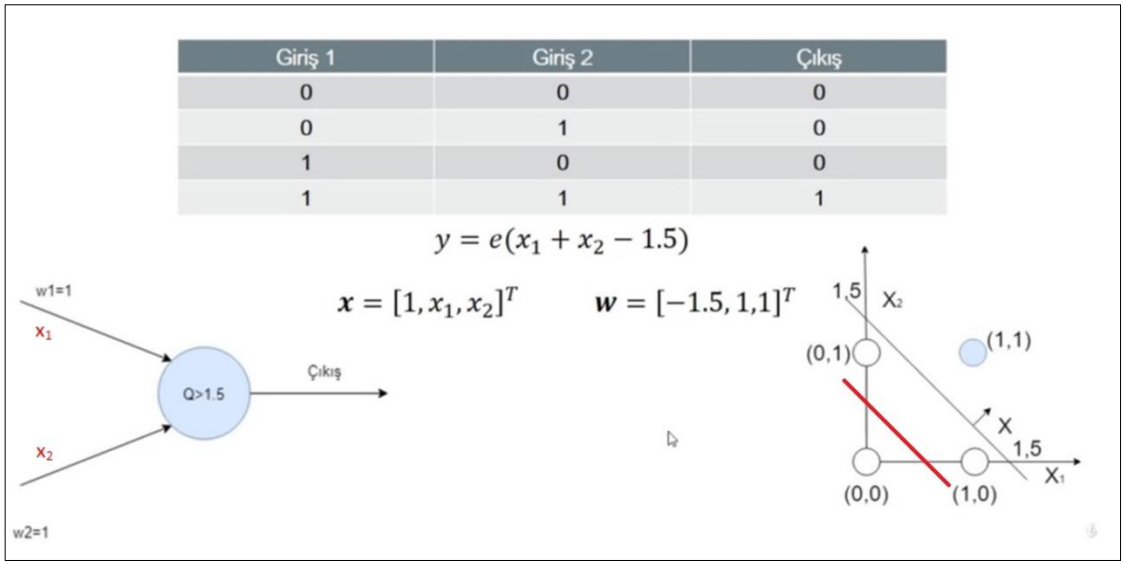
$$y_i \leftarrow \frac{\exp(o_i)}{\sum_k \exp(o_k)}$$

Tüm i = 1,…,K

Tüm j = 1,…,d

$$w_{ij} \leftarrow w_{ij} + \eta (r_i^t - y_i) x_j^t \quad (7)$$

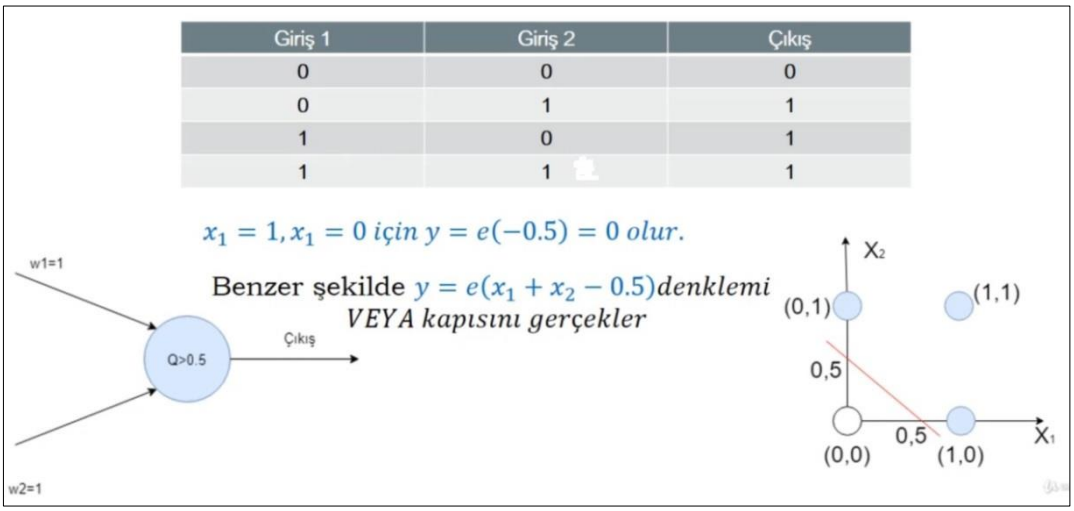
'VE' kapısı iki sınıflı çıkış üreten ve tek bir ayırtaçla simgelenebilen, tek katmanlı modelle ifade edilebilen bir problemdir. 'VE' kapısının çalışma şekli; iki giriş vardır girişteki değerlere göre çıkışta 0 ve 1 değeri gözlenir. Şekil 2.7'de görüldüğü gibi sinir hücresi eşik değerini 1,5 alınmıştır ve girişlerdeki değer 1,5'tan büyük olursa çıkış 1, değilse 0 olarak üretilmektedir. Ağırlıkları 1 seçilmiştir, 1 olarak seçilmesinin sebebi daha basite indirgemektir. İlk durumda girişler $x_1$ ve $x_2$ 0 olduğunda çarpımlar toplamı 0 edecektir dolayısıyla çıkış 1,5'tan (eşik değerden) küçük olduğu için sonuç 0'dır. Son satır hariç diğer değerlerde 1,5'tan küçük olacaktır. Bu uzayda gösterildiğinde $x_1$=0 $x_2$=0 olduğunda 0, $x_1$=0 $x_2$=1 olduğunda 0, $x_1$=1 $x_2$=0 olduğunda 0, $x_1$=1, $x_2$=1 olduğunda 1 üretmektedir. Bu 0'ların ve 1'lerin olduğu uzay doğrusal bir ayırtaçla (diskriminant) şekil 2.7'de gösterilmektedir. Burada gösterilen tek katmanlı bir sinir ağının çalışma şeklidir.




**Şekil 2.7.** VE Kapısı İle Sinir Ağı Modeli [17]

Bu ayırtacın eşik değerini ileriye geriye göre çektikçe modelin performansını artmaktadır. Örneğin şekil 2.7'deki modelde eşik değeri 1,5 yerine 0,5 olsaydı ayırtaç kırmızı ile çizilmiş doğru olacaktır ve sınıflandırma doğru şekilde yapılmayacaktı.

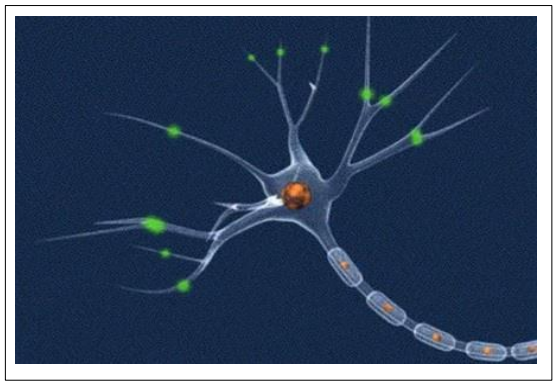
Şekil 2.8'deki örnekte benzer bir çalışma şekli olan 'VEYA' kapısı; çıkış yine iki sınıftan (0 ve 1) oluşuyor. Girişleri 'VE' kapısında olduğu gibidir fakat 'VEYA' kapısında çıkışlar farklıdır. Sadece girişlerin 0,0 olduğu durumlarda kapının çıkışı 0'dır. Diğer durumlarda kapı çıkışları 1'dir. Bu sefer ayırtacın (diskriminantın) eşik değerini 1,5 yerine 0,5 olarak alıyoruz 'VE' kapısında olduğu gibi aynı işlemi gerçekleştirildiğinde 0 olan değerlerle 1 olan değerleri ayırmış olacaktır. Bu basit tek katmanlı işlemler 'ÖZEL VEYA' problemi için gerçekleştirilmesi imkânsızdır.




**Şekil 2.8.** VEYA Kapısı İle Sinir Ağı Modeli [17]

## 2.4. Çok Katmanlı Sinir Ağları

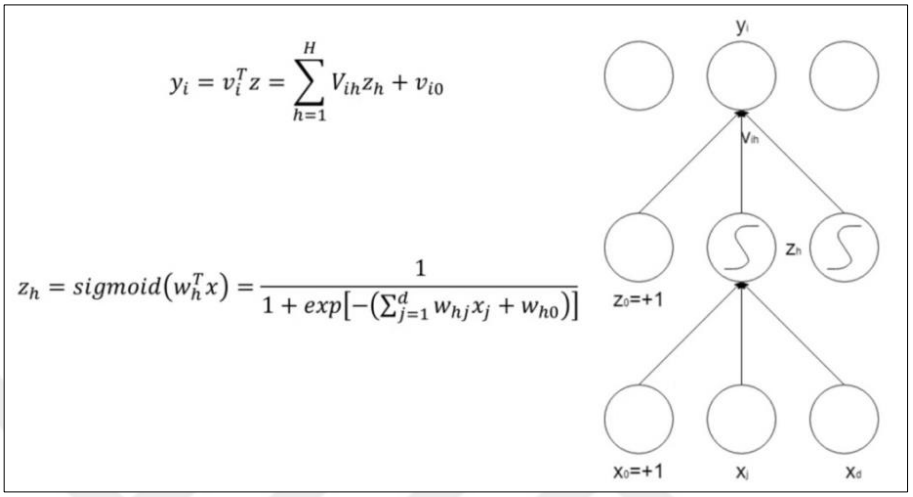
Çok katmanlı sinir ağı, algılayıcı ve sınıflandırma bağlamında kullanılan yapay sinir ağı yapısıdır. Hem model hem de eğitim türü bakımından tek katmanlı sinir ağıdan farklıdır. Çok katmanlı sinir ağlarındaki amaç beyindeki sinir ağlarının modellenmesini oluşturmak ve buna bağlı benzetimler yaparak birtakım problemlere çözümler üretmektir. Beyin görüntü, konuşma tanıma, öğrenme gibi yetenekleri açısından şu anki mühendislik ürünlerinden oldukça üstündür. Ve bu yetenekler bilgisayarlar üzerinde gerçekleştiğinde ekonomik açıdan önemli uygulamalar ortaya çıkmaktadır. Eğer beynin bu işlevleri nasıl yerine getirdiği anlaşılmış olunursa; bunları algoritmalar haline getirilip yapay olarak gerçekleştirebilir. İnsan beyni bilgisayardan oldukça farklıdır, bilgisayarların çoğunlukla tek bir işlemcisi olmasına rağmen beyin, sinir hücresi denen ve aynı anda çalışan çok sayıda örneğin $10^{11}$ kadar işlemci biriminden (nöron) oluşur. Her ne kadar ayrıntılı bilinmese de bu işlemci hücreleri bilgisayarların işlemcilerine göre daha basit ve yavaş olduğudur. Fakat beyni farklı kılan ve ona yüksek hesap gücü veren sinir hücreleri arasındaki bağlantılar olduğu düşünülmektedir. Beyindeki sinir hücresinin ortalama olarak yaklaşık $10^4$ farklı sinir hücresiyle bağlantısı vardır. Beyindeki işlem belleğin birlikte ağ üzerine dağıtıldığı düşünülmektedir. İşlemler ağ üzerindeki sinir hücrelerinde gerçekleştirilirken bellek bilgileri de aralarındaki bağlantılardır.



**Şekil 2.9.** Biyolojik Sinir Ağı [17]

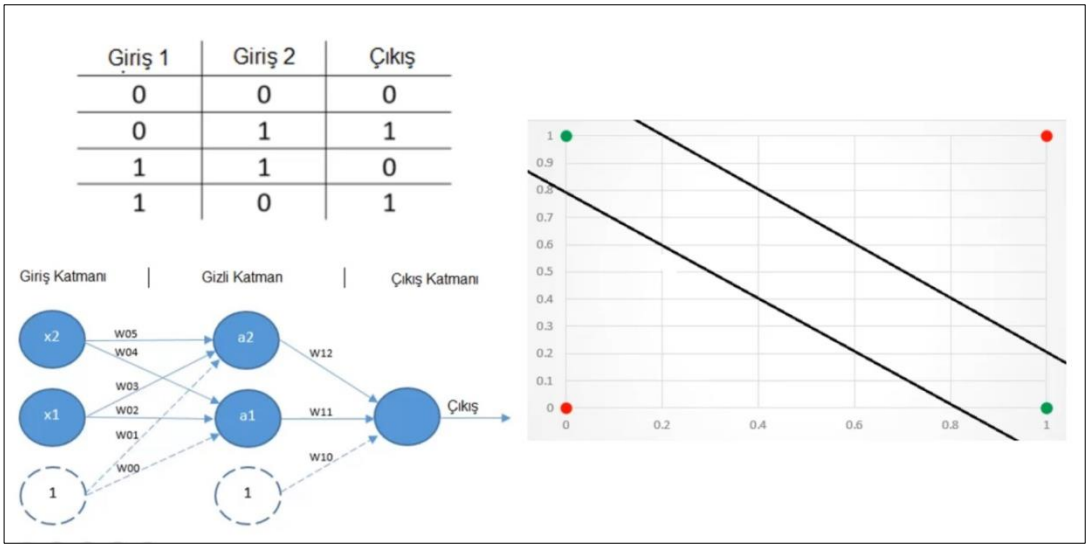
Aşağıdaki gibi çok katmanlı doğrusal olmayan bağlantılarda (non-linear regresyon) olabilir. x girdisi $x_0 = 1$ olarak girdi katmanına iletilir ve etkileşim ileriye doğru yayılarak gerçekleşir. $z_h$ ile gösterilen saklı katmanların değeridir. Burada yapılan işlem daha önce bahsedildiği gibi ağırlıklarla girişlerin çarpımlar toplamı yapılmaktadır. Elde edilen değere bir bias değeri eklenerek bir aktivasyon

fonksiyonuna sokulmaktadır. Aktivasyon fonksiyonu burada sigmoid fonksiyonudur. Saklı katmanlarda sigmoid fonksiyonu yerine doğrusal bir işlem yapılsaydı sonuç yine doğrusal olurdu ve gizli katman eklemenin bir anlamı olmazdı. Sigmoid fonksiyonu eşik işlemine benzer bir işlem ancak sürekli ve türevlenebilir bir işlemdir. Sigmoid fonksiyonu yerine başka aktivasyon fonksiyonları da kullanılabilir.




**Şekil 2.10.** Sinir Ağı İşlemleri

Daha önce bahsedildiği gibi ‘VE’ ‘VEYA’ kapıları gibi problemler tek katmanlı sinir ağı ile çözülebilmektedir. Fakat ‘ÖZEL VEYA’ problemi tek katmanlı sinir ağı ile çözülemediği bahsedilmişti. Şekil 2.11’de göründüğü gibi kırmızı noktalar arası veya yeşil noktalar arası birer doğru çizilirse sürekli hata ortaya çıkacaktır. Fakat iki katmanlı algılayıcı kullanıldığında iki tane ayırtaçla (diskriminant) sınıflar birbirinden ayrılmaktadır. Bu sınıf ayırma işlemi eklenen gizli bir katman sayesinde gerçekleşmektedir. Böyle bir yapıya gizli katmanı ve çıkış katmanı saymak şartıyla iki katmanlı sinir ağı denilmektedir. Giriş katmanı başka bir sinir ağının çıkış katmanı olabileceği için katman olarak sayılmamaktadır. Çıkış değerimiz x değerleri (giriş değerleri) ve w değerleri (ağırlıklar) çarpılıp toplamı bias değeriyle bir aktivasyon fonksiyonuna tabi tutulmaktadır. Hesaplanan değer bir sonraki katmana veya çıkışa aktarılmaktadır. Bu işlemin adına feed-forward (ileri besleme) denilmektedir..




**Şekil 2.11.** ÖZEL VEYA Kapısı İle Sinir Ağı Modeli [17]

## 2.5. Eğiticili Öğrenme (Supervised Learning)

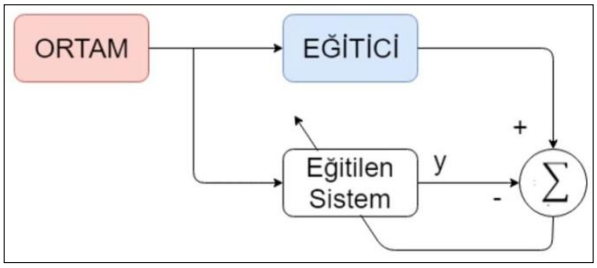
Sınıflandırma yöntemleri

- Destek Vektör Makineleri (Support Vector Machines)
- Ayrıştırma Analizi (Discriminant Analysis)
- Naif Bayes
- En Yakın Komşu (Nearest Neighbor)

Bağlanım (Regression) Yöntemleri

- Lineer Bağlanım
- Karar Araçları (Decision Trees)
- Yapay Sinir Ağları

Bir nesne sınıfı ayırmak gerekiyorsa sınıf sayısı ve örnek gruptaki her örneğin hangi sınıfa ait olduğunun bilindiği durumlardır. Ve eğitici ortam hakkında bilgiye sahip demektir. Eğitilen sistem ortam hakkında bilgiye sahip değildir. Eğitilen sistemdeki ağırlıklar eğitim kümesinde bulunan bilgiye ve hata aracılığıyla değiştirilmektedir. Sistem aslında geri dönen hataya göre ortamdan aldığı bilgiyle eğitilmektedir.



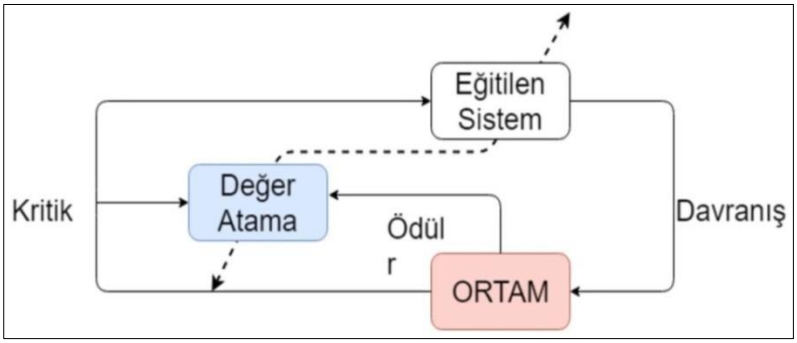

**Şekil 2.12.** Eğiticili Öğrenme [17]

## 2.6. Eğiticisiz Öğrenme (Unsupervised Learning)

Kümeleme Yöntemleri

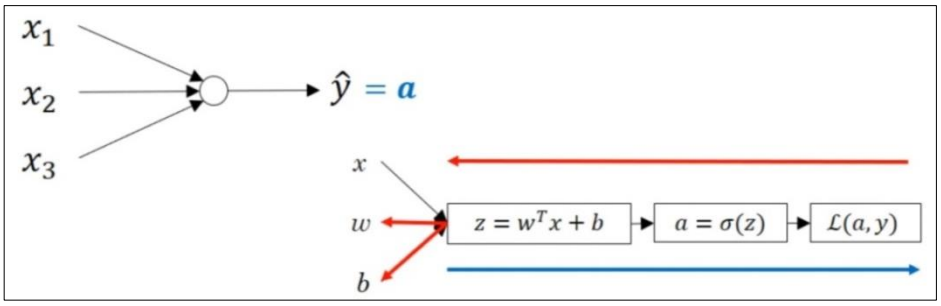
- K-Ortalama, Bulanık C-Ortalama
- Gauss Karışımı (Gaussian Mixture)
- Saklı Markov Modeli (Hidden Markov Model)
- Yapay Sinir Ağları

Burada aslında hangi nesnenin hangi sınıfa ait olduğunun, grup sayısının bilinmediği durumlarda gerçekleşir. Öğrenme işleminin her adımında istenilen yanıtı sağlayan bir eğitici yoktur. Eğitilen sistem sonuçta elde edilecek yanıta erişmek için gerekli davranışı, eleştiriyi göz önünde tutarak bulmak zorundadır. Eğitilen sistem tamamen ödül ceza ağırlığına göre bir değer ataması yapılır ve eğitilen sistem aslında ortamın davranışına göre eğitilmeye devam eder. Bunun için etiketli bir sınıf yoktur ve etiketsiz şekilde devam eder. Aslında buna pekiştirmeli öğrenmede denilebilir.

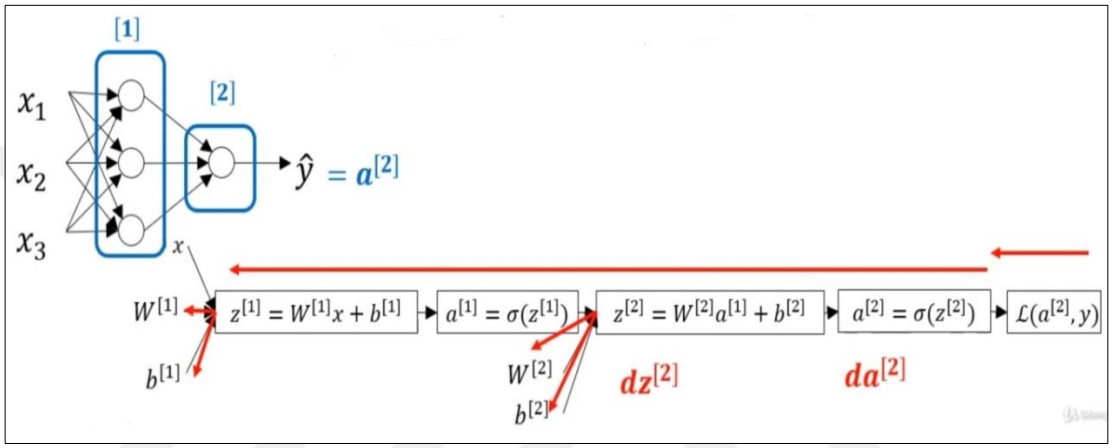



**Şekil 2.13.** Eğiticisiz Öğrenme

## 2.7. Öğrenme Nasıl Gerçekleşir?




**Şekil 2.14.** Sinir Ağı Öğrenme İşlemi [17]



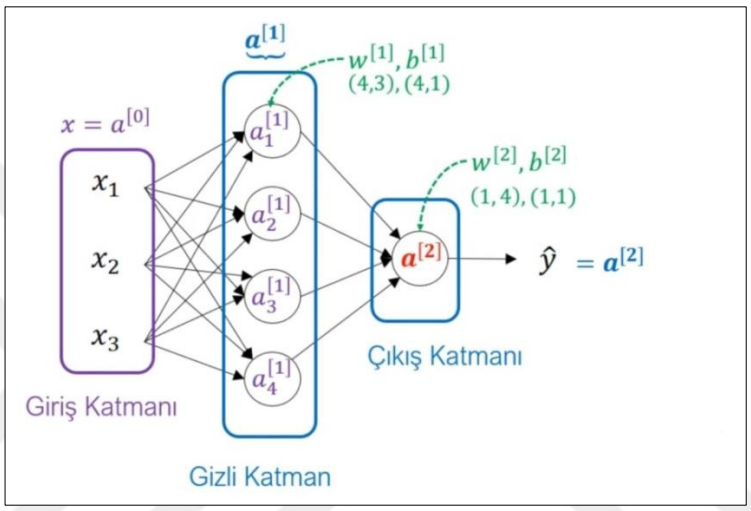

**Şekil 2.15.** Geriye Yayılım Algoritması [17]

Şekil 2.15'deki sinir ağında 1. katmandaki değerler tek tek hesaplanarak bir $z^{[1]}$ değerini bulunmuştur. Hesaplanan $z^{[1]}$ sigmoid fonksiyonuna (aktivasyon fonksiyonu) tabi tutularak $a^{[1]}$ değeri hesaplanmıştır. Her bir nöron için ayrı ayrı a değerleri hesaplama işlemi yapıldıktan sonra 2. katmana geçildiğinde giriş 1. katmandan gelen a değeri olmaktadır. Ağırlık değerleriyle ($w^{[2]}$) çarpılıp bir bias $b^{[2]}$ değeriyle toplandıktan sonra elde edilen $z^{[2]}$ değerini en sonunda da yine bir aktivasyon sokularak $a^{[2]}$ değeri elde edilmektedir. Yitim (loss) fonksiyonuna tabi tutulan çıkış değerleri loss değerine göre geriye yayılım (backpropagation) işlemine tabi tutulmaktadır. Geriye yayılım yaparken türev alarak gidilmekte ve yeni ağırlık değerler hesaplanmaktadır. Ardından tekrar ileri yayılımla yeni yitim değerimiz hesaplama işlemi yapılmaktadır. Daha detaylı inceleme yapılırsa; Şekil 2.16'da x'ler giriş katmanı, gizli katmanda 4 tane nöron bulunmaktadır ve $a^{[1]}$ olarak tanımlıdır. Her bir nöron girişe bağlı olarak ağırlık ve bias değeri içermektedir. Birinci katmanın birinci nöronu, birinci katma için olan ağırlıklar ($w^{[1]}$ (4,3)) ikinci katmandaki nöron sayısını, 3 ise o nörona gelen bilgileri kapsamaktadır. Çıkış katmanına $a^{[2]}$

denildiğinde elde bir nöron bulunmaktadır, bir önceki katmandan gelen 4 giriş ve 4 ağırlık var ($w^{[2]}$ (1,4)) ve bir tanede bias değeri ($b^{[2]}$ (1,1)) bulunmaktadır. Daha öncede bahsedildiği gibi elde edilen a değerleri aktivasyon fonksiyonlarının çıktısıdır. Sonra a değerleri en son çıkışa gelene kadar bir sonraki katmanlara aktarılmaktadır.

$$z = w^T x + b$$

$$a = \sigma(z) \tag{8}$$




**Şekil 2.16.** Sinir Ağı Katman İncelemesi

$a^{[1]}$ için söyleyecek olursak, $a^{[1]}$ bir vektör için dört tane nöronun değerleri var. Ve a değerleri her bir nöron çıktısının (z değerinin) aktivasyon fonksiyonun çıkışıdır. İleri yönde bu işlemler yapılarak yitim fonksiyonu elde edilmektedir.

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[2]} \\ a_3^{[3]} \\ a_4^{[4]} \end{bmatrix} = \sigma(z)$$

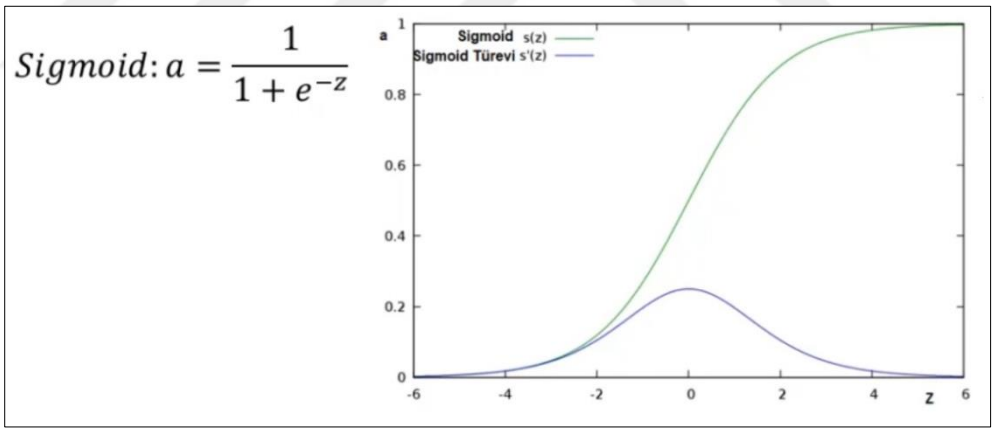
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]},\ a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_1^{[1]},\ a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]},\ a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]},\ a_4^{[1]} = \sigma(z_4^{[1]}) \tag{9}$$

İleri yöne besledikten sonra geri yöne yayılım yapmak önemlidir. Bunun için aktivasyon fonksiyonlarının türevlenebilir olması gerekir. Klasik yöntemlerde sigmoid fonksiyonu kullanılmaktadır ancak bugün farklı aktivasyon fonksiyonları da kullanılmaktadır. Sigmoid fonksiyonu Şekil 2.17'de gördüğünüz gibi sıfır ile bir arasında değişen üstel bir fonksiyondur. Aktivasyon fonksiyonu katmanlarda kullanılır ve türevlenebilir olması gerekir. Bunun nedeni ise ağ eğitiminde geriye yayılım algoritması sırasında ağırlıklar aktivasyon fonksiyonu sayesinde türev alınarak güncellenir ve yeni ağırlıklar ortaya çıkar yitim fonksiyonu yeni ağırlıklarla yeniden hesaplanır. Aktivasyon fonksiyonunun türev ortalaması 0 olduğunda daha iyi performans sağlamaktadır. Aktivasyon fonksiyonlarını ileride ki konularda detaylı olarak inceleyeceğiz. Ara katmanlarda hiperbolik tanjant fonksiyonu kullanılması daha avantajlıdır fakat çıkış katmanında bazen sigmoid fonksiyonu kullanmak zorunlu hale gelmektedir. Nedeni ise çıkışın 0-1 aralığında bir değer olması istenildiğinde ve sigmoid fonksiyonu da 0-1 aralığında bir değer ürettiği için sigmoid fonksiyonu çıkışta kullanılmaktadır. Buradan anlaşılacağı üzere aktivasyon fonksiyonu farklı katmanlarda farklı seçilebilir.




**Şekil 2.17.** Sigmoid Fonksiyonu

Son yıllarda derin öğrenme ile popüler olmaya başlayan aktivasyon fonksiyonu da ReLU (Rectified Linear Units)'dur. Çıkış değeri z'nin sıfırdan olan küçük değerlerinde sıfır olmaktadır. Çıkış değeri z'nin sıfırdan büyük değerlerinde artan (lineer) değerler almaktadır. Yani z'nin sıfırdan büyük olduğu değerlerde türevlenebilir anlamına gelmektedir. Sıfırdan küçük olduğu durumlarda türevi sıfır çıkar ve burada bir öğrenme gerçekleşmemiş olur. Aktivasyon fonksiyonunun türevi alındığında mümkün oldukça sıfırdan farklı olması istenilmektedir nedeni ise

öğrenme işlemi türevin sıfır olduğu durumlarda gerçekleşmemektedir. Çıkış fonksiyonu z'nin küçük olduğu ve büyük olduğu (0 değeri hariç) durumlarda da türevlenebilir fonksiyon olan Leaky ReLU fonksiyonu kullanılmaktadır. Ancak bu fonksiyonda işlem yükünden dolayı fazla kullanılmamaktadır. Bunun önüne geçmek için optimizasyon işlemlerinin yapılması gerekmektedir. Son yıllarda derin öğrenmede en çok ReLU fonksiyonu kullanılmaktadır. Sigmoid fonksiyonu neredeyse hiç kullanılmamaktadır. İşlem yükü az olan öğrenmelerde ise Leaky ReLU fonksiyonu tercih edilmektedir. Kısaca özetlemek gerekirse modelin, verinin ve alınmak istenilen sonuçlara bağlı olarak hangi aktivasyon fonksiyonunun kullanılacağı değişkenlik gösterir.

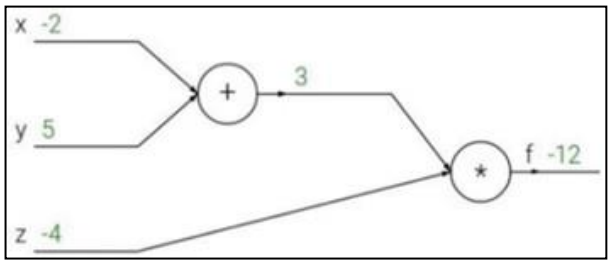
## 2.8. Geriye Yayılım Algoritması (Backpropagation Algorithm)

Bir yapay sinir ağı tasarlarken, başlangıçta ağırlıklar rastgele değerler veya herhangi bir değişken ile başlatırız. Dolayısıyla, seçilmiş olan ağırlık değerleri doğru olmalı ya da modelimize en uygunu olmalıdır. Fakat model çıktısı gerçek çıktıdan çok farklı, yani hata değeri çok büyük olabilir. Hatayı azaltmak yani minimize etmek için yapılması gereken ağırlıkların güncellenmesidir. Ağırlıkları güncellemek için geriye yayılım algoritmasını (backpropagation) kullanılması gerekmektedir. Basit bir yapıyla anlatılırsa;

Bir f fonksiyonunda x, y ve z girişleri bulunmaktadır;

$$f(x, y, z) = (x + y)\, z \tag{10}$$

(x + y) birinci düğümü oluşturmaktadır. (x + y) ile z'nin çarpımı ise fonksiyonun çıkışıdır. İleri yayılım algoritması bu şekilde gerçekleşecektir.




**Şekil 2.18.** İleri Yayılım Algoritması

Geriye yayılım algoritması ise; öncelikle f'den (çıkıştan) geriye türev alarak devam edecektir. $q = x + y$ denilirse;

$$q = x + y \quad \frac{dq}{dx} = 1 \quad \frac{dq}{dy} = 1 \tag{11}$$

q'nun x'e göre türevi ve q'nun y'ye göre türevi 1 çıkacaktır. Diğer fonksiyon ise;

$$f = q.z\,, \quad \frac{df}{dq} = z \quad \frac{df}{dz} = q \tag{12}$$

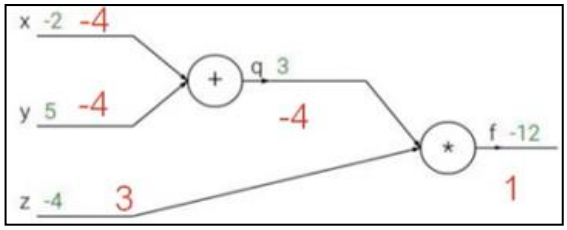
f'in q'ya göre z, f'in z'ye göre türevi q'yu verecektir.

Birinci adımda f'in f'e göre türevi 1 olduğu için değer 1'dir. Bir sonraki adımda f'in z'ye göre türevi q'ya eşit olduğu için (df/dz=q) değer 3'tür. İlerlendiğinde f'in q'ya göre türevi z'ye eşit olduğu için (df/dq=z) değer -4'tür. Bir sonraki adımda f'in y'ye göre türevini almak için zincir kuralını uyguluyoruz.

$$\frac{df}{dy} = \frac{df}{dq}\frac{dq}{dy} = -4.1 = -4 \tag{13}$$

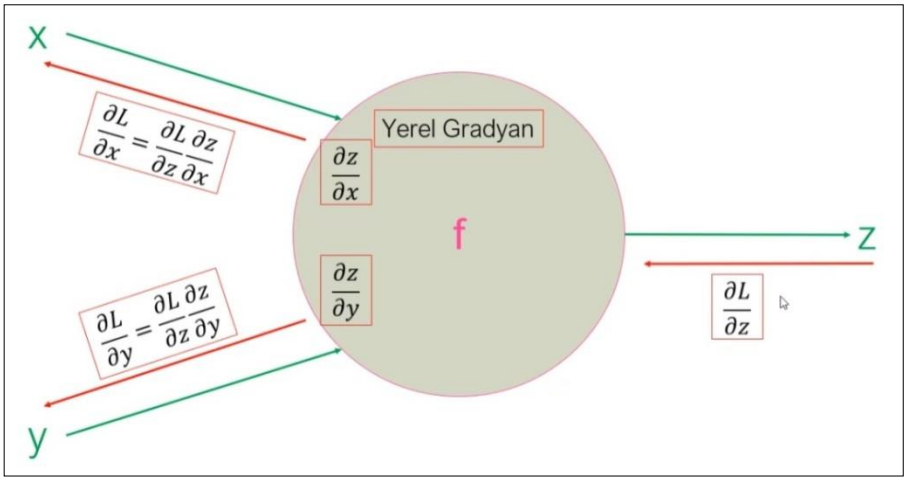
Diğer adımda f'in x'e göre türevi alırken aynı şekilde zincir kuralını uyguluyoruz.

$$\frac{df}{dz} = \frac{df}{dq}\frac{dq}{dx} = -4.1 = -4 \tag{14}$$




**Şekil 2.19.** Geriye Yayılım Algoritması

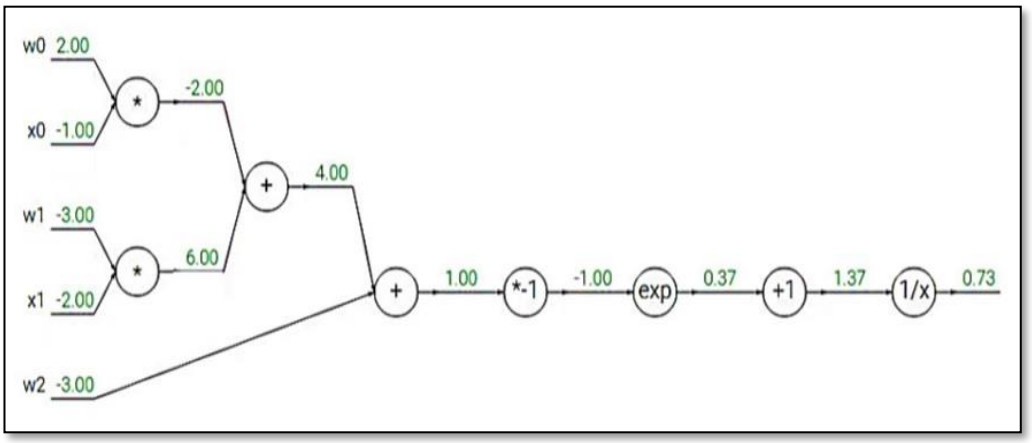
Sonuç olarak geriye yayılımda her bir hücreyi inceleyecek olursak; girişimiz, çıkışımız ve geriye yayılım yapacağımız zaman çıkışta bir yitim (loss) değeri olacaktır. Genel olarak türev alma işlemleri Şekil 2.20'deki gibi olacaktır. Girişler başka nöronlara da bağlı olabilir ve işlemler aynı şekilde devam eder, giriş bilgisine ulaşıldığında sistem tekrar ileri ve geri yönde yitim fonksiyonu minimize olana kadar devam eder. Bu işlemin tamamına öğrenme işlemi denir.




**Şekil 2.20.** Geriye Yayılım Türev Alma İşlemi [15]

Başka bir örnek devam edersek; Aşağıda verilen fonksiyonu bir grafik haline getirdiğimizde Şekil 2.21'deki değerler elde edilir.

$$f(x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}} \tag{15}$$




**Şekil 2.21** Geriye Yayılım Ağırlık Hesabı [15]

Bu grafikte geriye yayılım algoritması sırayla uygulanırsa; yukarıdaki örnekte hatırlarsak ilk işlemimiz f'in kendine göre türevini aldığımızda değerimiz 1 olacaktır. 2. adımımız 1/x'in x'e göre türevini alındıktan sonra 1 ile çarpılıyor ve yeni değerimizi atıyoruz.

$$f(x) = \frac{1}{x} \rightarrow \frac{df}{dx} = -\frac{1}{x^2}\left(\frac{-1}{1.37^2}\right).(1.00) = -0.53 \tag{16}$$

3. adımda sabit değerle toplama işlemimiz var. Sabit değerle toplama işleminin türevi 1'dir. Ve bir önceki değerle çarpıp yeni değeri atıyoruz.

$$f_c(x) = c + x \rightarrow \frac{df}{dx} = 1 \qquad 1.(-0.53) = -0.53 \tag{17}$$

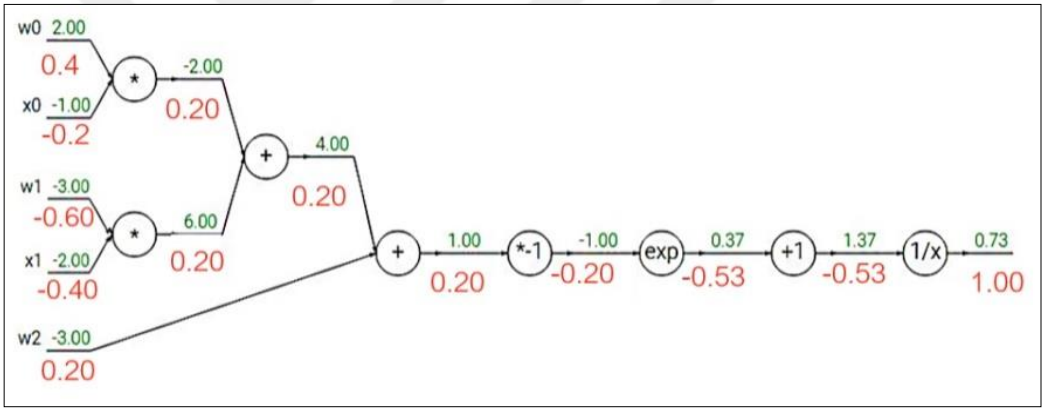
4. adımda eksponansiyel türevini almak gerekiyor. Ve bir sonraki adımın değeri elde ediliyor.

$$f(x) = e^x \rightarrow \frac{df}{dx} = e^x(e^{-1}).(-0.53) = -0.20 \tag{18}$$

5. adımda sabitle çarpma işlemi yapılmaktadır. Kendisiyle bir önceki değer çarptıktan sonra bir sonraki değer bulunuyor.

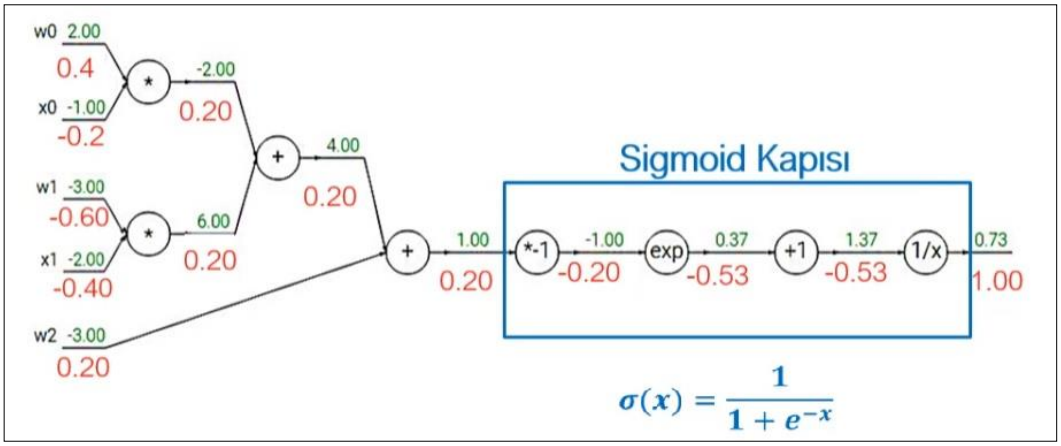
$$f_a(x) = ax \rightarrow \frac{df}{dx} = a \qquad (-1).(-0.20) = 0.20 \tag{19}$$

Diğer adımlarda türev alarak giderek ve tüm değerler (ağırlıklar) Şekil 2.22'de gösterildiği gibi güncellenir.




**Şekil 2.22.** Geriye Yayılım Ağırlıkların Güncellenmesi [15]

Pratikte grupları şekil 2.23'te gösterildiği gibi bir arada toplanılabilir. Mavi ile gösterilen bölge bir sigmoid yapısıdır. Birden fazla türev almak yerine sadece sigmoid fonksiyonunun türevini alarak aynı değerler elde edilebilir.



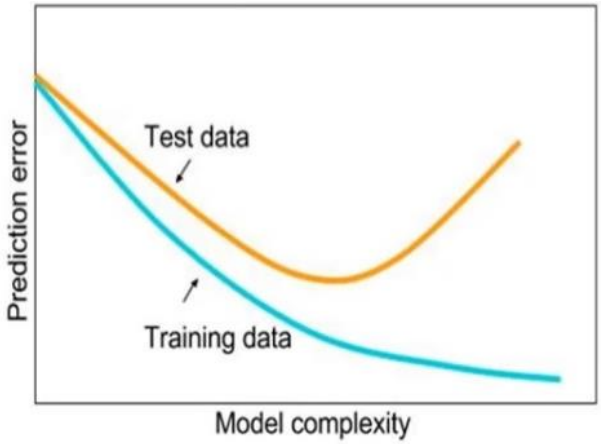

**Şekil 2.23.** Geriye Yayılım Sigmoid İşlemi [15]

İleri yayılım ve geri yayılım algoritmalarını kısaca özetleyecek olursak; ileri yayılım algoritmasında ağırlıkların (w) ve girişlerin (veya bir önceki katmandaki nöronların çıkışları) çarpımlar toplamı artı bir bias değerinin toplanmasıyla aktivasyon fonksiyonu önceki değeri (z) vermektedir. Daha sonra z bir aktivasyon fonksiyonuna (g) tabi tutulduktan sonra a değeri elde ediliyor. Bu a değeri bir sonraki katmandaki nöronun girişi veya nihai çıkışımızda olabilmektedir. Geriye yayılım algoritmasında son elde edilen değerden yola çıkarak türev alarak bir önceki katmanların ağırlıkları elde edilir ve ilk girişe kadar türev alma işlemi devam edecektir. Ve tekrar ileri yönde yayılım devam edecektir. Bu döngü yitim (loss) değeri minimize olana kadar devam eder ve tüm öğrenme işlemi bu şekilde devam eder.

# 3. OPTİMİZASYON VE REGULARİZASYON

## 3.1. Hiper Parametre Nedir? Problem Ve Çözümler

Makine öğrenmesinde ya da derin öğrenmede bir takım farklılıklar vardır. Makine öğrenmesinde öz nitelik çıkarmak ve seçmek çok ciddi bir iştir. Öz nitelik dediğimiz problemi anlatan özelliklerdir. Bu özelliklerin seçilmesi ve çıkarılması çok önemlidir. Daha sonra model bu verilerle eğitilir ve eğittikten sonra önemli gibi görülen öz niteliklerin ağ için bir şey ifade ettiği görülebilir. Modelin sonucunda elde edilen başarımlar sayesinde önemli, önemsiz öznitelikler görülür. İşlem yükünü azaltmak için önemsiz olanlar çıkarılır ve önemli olan özellikler eğitimlerde kullanılır. Bunun adına da öz nitelik seçme denilirdi. Bu önemli bir işti hatta öz nitelik mühendisliği dahi ortaya çıkmıştı. Fakat derin öğrenme çıktıktan sonra öz nitelik mühendisliği ortadan kalktı. Çünkü derin öğrenmede katmanlar boyunca öz nitelikler kullanıcıya gerek kalmadan seçilir ve çıkarılır. Kullanıcıya sadece mimariyi düzgün seçmek kalıyor. Burada katman sayısı, nöron sayısı hangi optmizasyon algoritmalarının kullanıldığı, öğrenme oranının ne seçildiği parametrelere karar vermek gerekiyor. Derin öğrenme konusunda tecrübe çok bile olsa ilk anda hiperparametreleri en efektif şekilde seçmek çok zordur çünkü seçilen parametreler veriler ve kullanılan donanımlar gibi birçok değişkeni aynı anda ağa etki etmektedir. O yüzden daha önce seçilen parametre bir sonraki modelde düzgün çalışmayabilir. Model aynı olsa da kullanılan donanım, çalışma alanınız, seçilen veri seti farklı olabilir. O yüzden belirli bir parametre en iyidir denilemez ama her modele uygun parametreler doğru şekilde seçebilmek için birtakım yollar ve yöntemler vardır. Örneğin şekil 3.1'de görüldüğü gibi veri seti sisteme verildiğinde eğitimin ve test hatasının sürekli düştüğü sırada bir anda test hatası artmaya başlar ve eğitim hatası düşmeye devam edebilir. Sürekli karşılaşılan bir durumdur. Bu durumu ortadan kaldırmak için hiper parametreleri doğru seçmek gerekir.



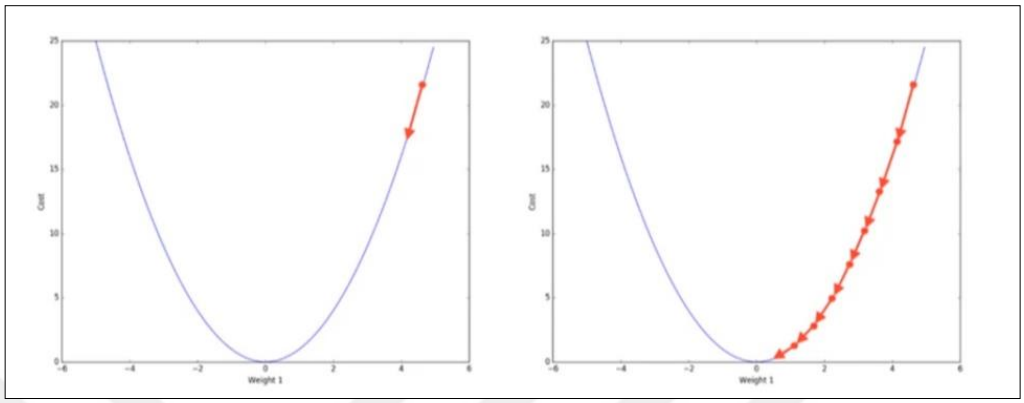

**Şekil 3.1.** Test hatası [15]

Veriden öğrenen makine öğrenmesi ya da derin öğrenme modelleri tasarlarken ne olması gerektiği çeşitli koşullar göz önüne alınarak modeli tasarlayan kişiye bırakılmış; probleme, veri setine ve donanıma da bağlı olarak değişkenlik gösteren parametrelere hiper-parametre denir. Yani tamamen değişken hiçbir zaman net bir şey söylenemeyecek bir konudur. Derin öğrenme modeli tasarlarken elde fazla veri olması gerekir. Ne kadar fazla veri olursa o kadar iyi denilemez ama yine elde yeteri kadar veri olması gerekir. Ayrıca verinin fazla olması yetmez o verinin gerçek dünyayı yansıtacak şekilde çok çeşitli olması gerekir. Eğitim süresini optimizasyon sırasında uzun tutmak gerekir. Veya farklı optimizasyon yöntemleri denemek gerekebilir. Her optimizasyon yöntemi hatayı minimize ederken aynı adım aralıklarıyla veya aynı sürede yaklaşmayabilir. Çeşitli kriterlere farklı yaklaşımları vardır. Bazen lokal minimumlarda takılabilir önemli olan global minimumları (maliyet fonksiyonundaki en düşük nokta) bulabilmektir. Önemli olan kaybın (lokal minimum) minimum olduğu noktayı bulmaktır. O yüzden farklı optimizasyon algoritmaları denemek faydalı olabilir. Ayrıca daha büyük veya daha küçük bir ağ kullanmak gerekebilir. Seyreltme uygulayabiliriz belli nöronların yok edilmesi bu unutma işlemini sebep olabilir. Düzenleme yöntemlerini kullanmak (regularizasyon) modelin mimarisinden bağımsız olarak daha başarılı sonuçlar edilebilir.

### 3.2. Öğrenmede Optimizasyon Problemi

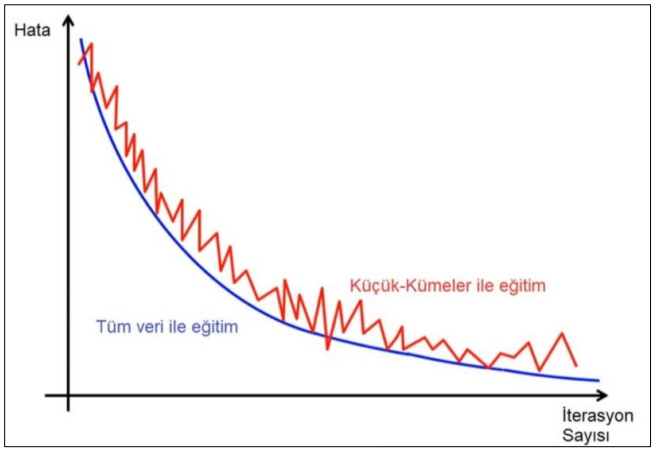
Maliyet hesabını minimize eden ağırlıkların bulunması için kullanılan optimizasyon doğrudan öğrenme yeteneğini ve hızını etkiler. Özellikle geriye yayılım algoritması işlem yükü bakımından oldukça karmaşıktır ve tüm veriyle eğitilen modellerde geriye yayılım süreci oldukça yavaş ilerler. Bu yüzden girdinin modele parçalar

halinde verilmesi mantıklı bir çözüm olabilir. Tüm veriyi bir anda vermektense veriyi parçalar halinde vermek modele iletmek gerekir. Buna da mini-bach (küçük-kümeleme) yöntemi denir. Mini-bach parametresi modelin aynı anda kaç veriyi işleyeceği anlama gelir. Giriş verisinden oluşan küçük paketler olarak düşünülebilir.




**Şekil 3.2.** Stokastik-Gradyan İniş Algoritması [15]

Seçilen kümeler üzerinden eğitim gerçekleşirken osilasyon meydana gelir. Bunun sebebi bir küme rastgele olarak seçilir. Bazıları için çok iyi eğitim kümesi olabilir. Yani genel verileri kapsayan altküme iken bazısında genel veriyi kapsamayan rastgele seçildiği için altkümelere denk gelmeyebilir. Bu tip durumlarda iyi veya daha kötü sonuçlar elde edildiğinden dolayı iterasyon sırasında osilasyonlar meydana gelir. Dikkat edilmesi gereken küme boyutunun ne kadar küçük veya büyük olması gerektiğidir. Geneli kapsayacak nitelikte olmasına dikkat edilmesi gerekir. Bu durumda öğrenme sürecinin, maliyet fonksiyonu yerel minimuma takılıp global minimumun bulanamaması ya da bunun çok uzun sürmesi olabilir bu da negatif etkilerinden biridir. Bu yüzden stokastik gradyan iniş optimizasyonu momentum ile birlikte kullanılıp daha tutarlı bir optimizasyon algoritması oluşturulabilir. Ya da karekök hata olasılığı, Rmsprop, Adagradi, Adadelta gibi bir takım optimizasyon yöntemleri denenip veya bazıları bir arada kullanılabilir. Böylelikle global minimuma yaklaşma daha hızlı olabilir.

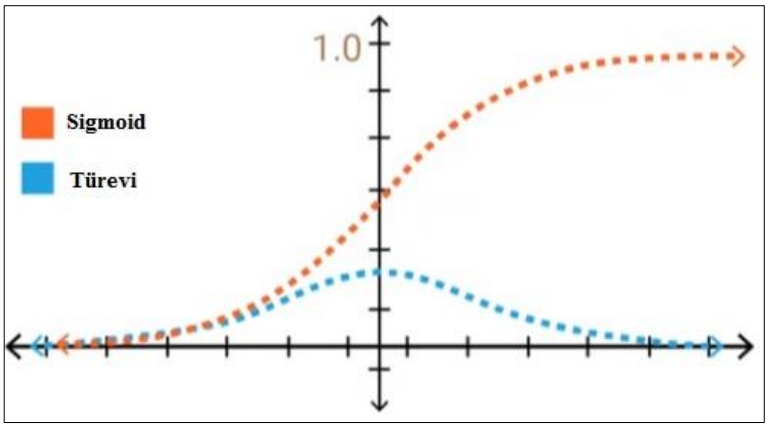



**Şekil 3.3.** Stokastik-Gradyan İniş Algoritması Hata Değerlendirmesi [15]

Derin öğrenme modellerinde genel varsayılan optimizasyon algoritması Stokastik gradyan iniş algoritmasıdır. Ama diğer yöntemlere göre biraz yavaştır.

### 3.3. Aktivasyon Algoritması Seçimi

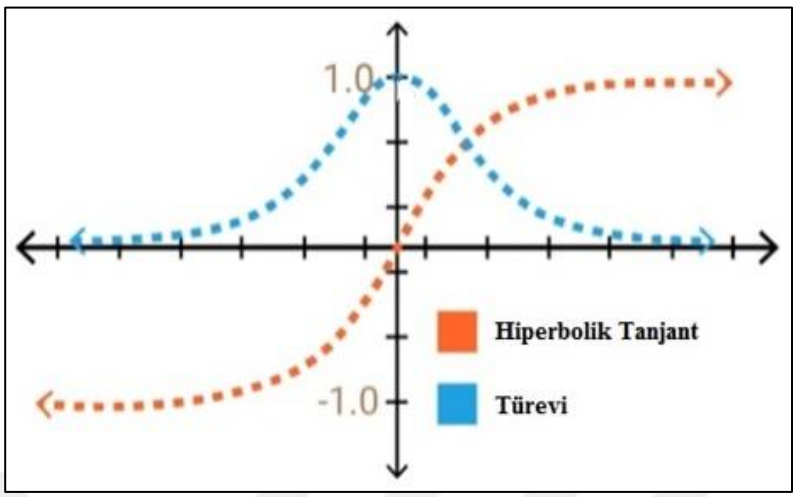
Geriye yayılım algoritmasında öğrenmeyi sağlan önemli bir fonksiyon seçimi yapılması gerekiyor. Çünkü türevin sıfır olduğu noktalarda işlem yükü hafifleyecektir. Bu istenilen bir şeydir fakat daha önemlisi türevin sıfır olduğu noktalarda öğrenme işlemi yapılmayacak bu da hiç istenilmez. Şekil 3.4'de görüldüğü gibi sigmoid aktivasyon fonksiyonu 0 ile 1 arasında değişen bir fonksiyonudur. Türevinin aldığı maksimum değer oldukça düşük ve sıfır bölgeleri mevcuttur. Bu yüzden çok iyi öğrenme gerçekleşmeyecektir.




**Şekil 3.4.** Sigmoid fonksiyonu ve Türevi [17]

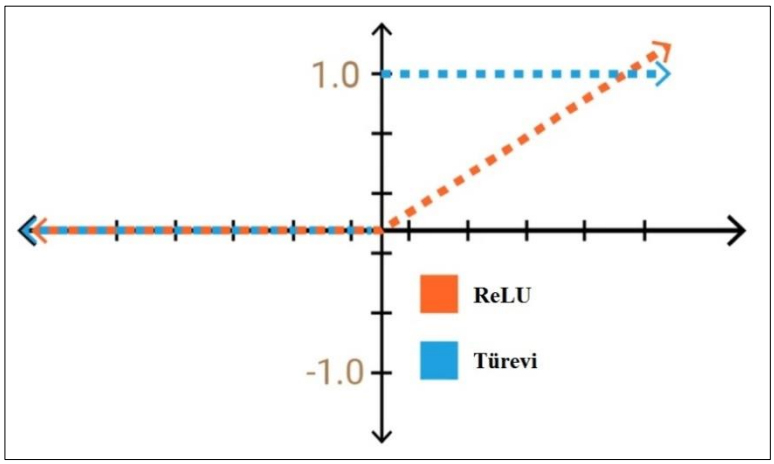
Daha çok hiperbolik tanjant tercih edilir ve hiperbolik tanjant -1 ile +1 arasında değişen sigmoid gibi üstel fonksiyondur. Hiperbolik tanjant türevinin maksimum noktası 1'dir ve bu yüzden 0 ile 1 arasında bir değer alır. Sıfır olduğu yerler vardır

bundan dolayı öğrenme işlemi çok verimli denilemez fakat sigmoid fonksiyonuna göre daha iyidir.



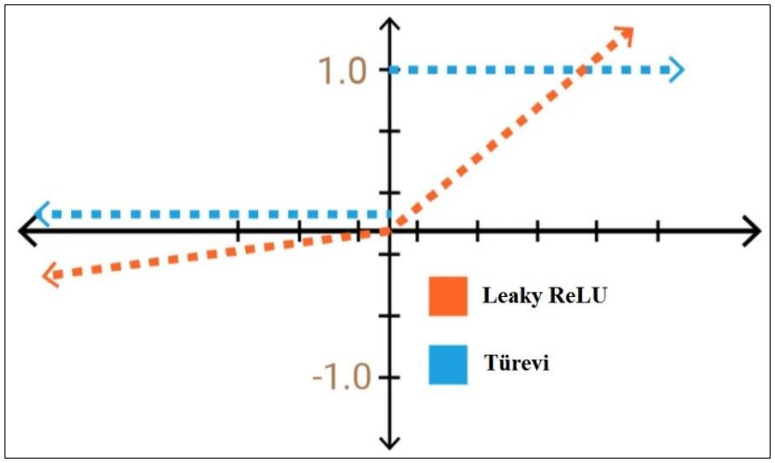

**Şekil 3.5.** Hiperbolik Tanjant Fonksiyonu ve Türevi [17]

ReLU (Rectified Linear Units) fonksiyonunda 0'dan 1'e kadar tamamen türevlenebilir ve türevlenebilir bir basamak oluşur. Ve derin öğrenmede en çok kullanılan aktivasyon fonksiyonudur.



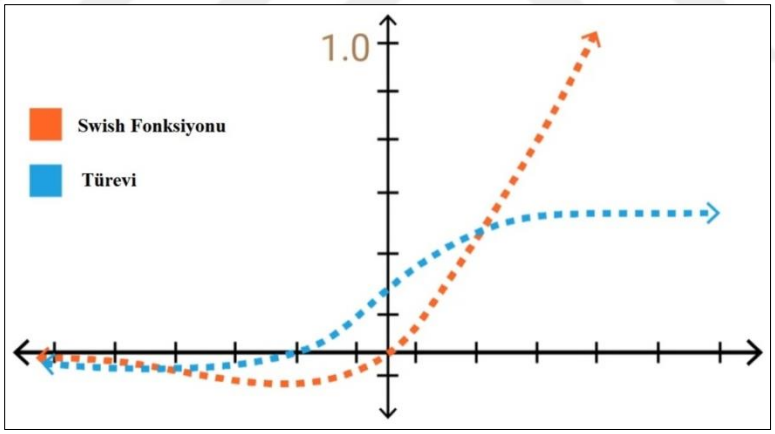

**Şekil 3.6.** ReLU Fonksiyonu [17]

İşlem yükü daha fazla olan uygulamalar Leaky ReLU kullanılmaktadır. Leaky ReLU'nun ReLU'dan farklı olarak 0'dan küçük değerlerinde de değer aldığı için türevide 0 değildir. Bu yüzden öğrenme oranı çok yüksektir ama işlem yükü çok fazladır.




**Şekil 3.7.** Leaky ReLU Fonksiyonu [17]

Google tarafından Swish fonksiyonunun başarımı arttırdığı görülmektedir. Swish fonksiyonu ReLU fonksiyonuna göre % 20 daha yavaş olduğu için özellikle akademide tercih edilmiyor. Ama öğrenme oranı ReLU'dan daha fazladır. Swish fonksiyonu yine bir noktada 0 görülür. Fakat Leaky ReLU gibi 0'dan küçük değerlerde bir değer almaktadır. Bu da öğrenme oranının yüksek olduğu anlamına gelmektedir.




**Şekil 3.8.** Swish Fonksiyonu [17]

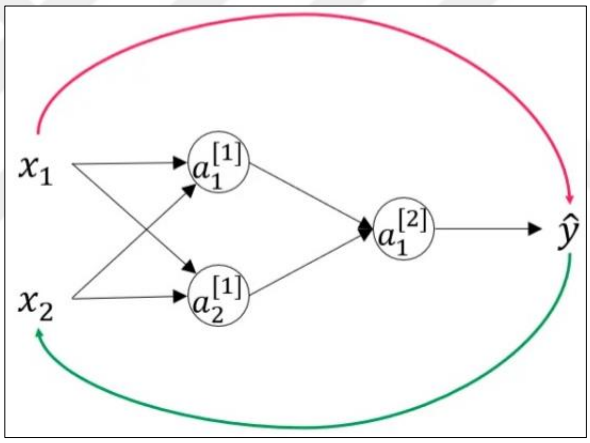
### 3.4. Eğitim Tur Sayısı (Epoch)

Modelin başarımı test edildikten sonra elde edilen başarıma göre geriye yayılım ile ağırlıklar güncellenir ve tekrar model eğitilir. Bu tekrar model eğitim işinde tekrar etme sayımız eğitim tur sayısını yani epoch değerini verir. Epoch sayısı arttıkça başarım bir süre devam edebilir. Başarımın artması azaldığında ya da durduğunda daha epoch işlem yapmaya gerek yoktur. Örneğin modelimizde 100 epoch çalıştırılmak istenildiğinde başarım 70. epoch'tan sonra artmaya devam etmiyor ya

da sabit kalıyordur o halde 100'e kadar devam etmeye gerek yoktur. Bu gibi hiper parametreler eğitim süresini optimize eder. Epoch sayısı problemin türüne aynı diğer parametrelerde olduğu değişkenlik gösterir.

### 3.5. Başlangıç Ağırlıklarının Belirlenmesi

Ağırlıklar 0 değerleri ile kesinlikle başlatılmamalıdır. Rastgele küçük sayılarda başlamakta fayda vardır. Çünkü öğrenme işlemini gerçekleştirmek için sinirdeki daha doğrusu modeldeki tek bir nöronun her birinin farklı değerler öğrenmesini istediğimizde göre rastgele değerler atanmalıdır. Şekil 3.9'daki örnekte girişlerin çıkışlara iletilirken ve daha sonra geriye yayılımda ağırlıklar güncellenirken aynı işlemler yapılması istenilmez ve her bir nöronda yeni değerler öğrenilsin. Bu yüzden 0 değil küçük bir sayıyla başlatılması gerekir. Eğer bir regülasyon yapılıyorsa yani tek bir nöron varsa 0'la başlatmakta sorun yoktur.



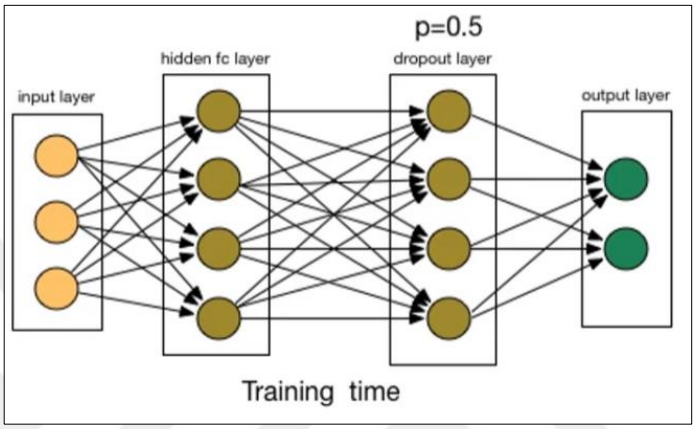

**Şekil 3.9.** Başlangıç Ağırlıklarının Belirlenmesi

Ağırlıkları güncellerken bir takım kriterleri göz önünde tutmak gerekiyor örneğin türev yüksek iken ağırlıkların küçük olması sağlanır. Örnekleme hatasına uyulmasını engeller veya daha esnek bir model olup olmadığını yine ağırlıkların nasıl güncellendiğiyle anlaşılır. Bu yüzden ağırlıkların mutlak değeri kullanılabilir.

### 3.6. Düğüm Seyreltme (Dropout)

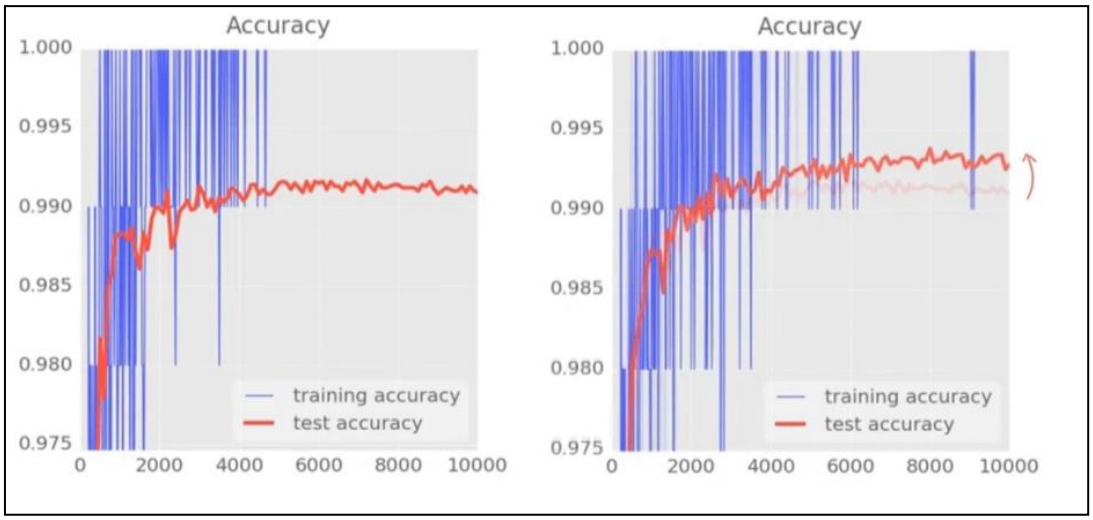
Tam bağlantılı katmanlarda eşik değerin altında kalan nöronların iptal edilmesi başarımı artırabilir. Örneğin şekil 3.10'da gizli katmanda (hidden layer) bazı değerler hesabı yapılmaktadır. Bir sonraki katmana geldiğimizde bazı nöronlarda ağırlık değerleri eşik seviyesinin altındadır. Şekil 3.10'daki örnekte eşik seviyesi 0 ile 1 arasında değişen noktada 0,5'in altında öğrenme oranı olan değerler kullanıcı için

önemli değilse öğrenilen özellikler iptal edilebilir. Bu da işlem yükünü azalmayı sağlamaktadır. Nöronlar iptal edildiğinde çıkışta performans azalması gerçekleşmez ama işlem yükü bakımında oldukça verimli hale gelmektedir. Ayrıca eşik değer istenildiği gibi veya rastgele olarak seçilebilir. Farklı epoch'larda farklı dropout'lar gerçekleştirilebilir. Böylelikle model değişik varyasyonları öğrenmiş olur. Dropout öğrenme modellerinde genellikle maksimum % 50 kullanılır. Yani en son eklenen dropout katmanında % 50'sini unut denilebilir ve böylelikle model genelleştirilir.

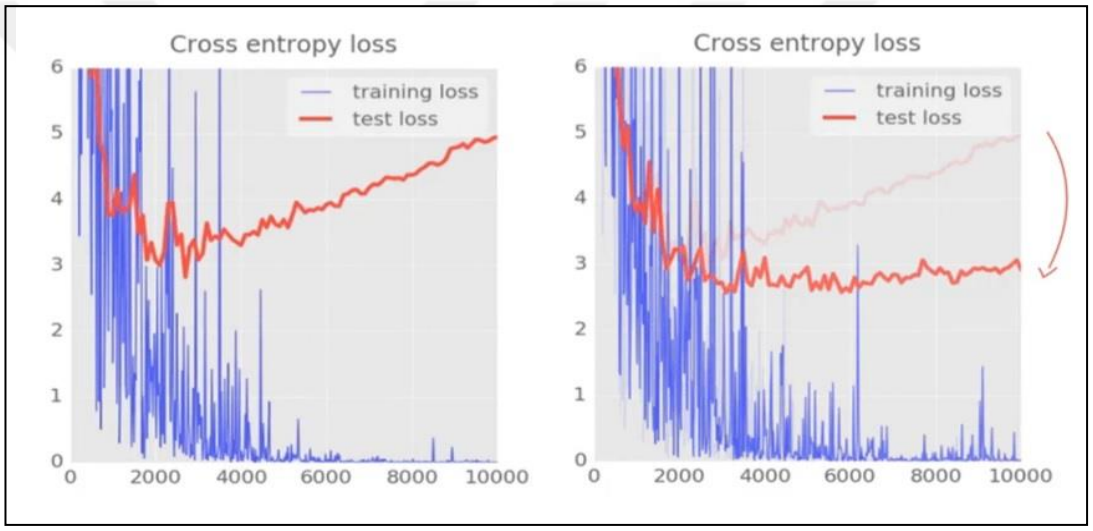



**Şekil 3.10.** Katman Seyreltme (Dropout) [5]

Şekil 3.11'de görüldüğü gibi bir eğitimin evrişimli sinir ağı (Convolutional neural network) modelinde düğüm seyreltme (dropout) uygulanmadan önce ve uygulandıktan sonraki başarımın artışı görülmektedir. Düğüm seyreltme uygulandıktan başarım artmaktadır. Veya şekil 3.12'deki yitime bakılırsa yitimin oldukça azaldığı görülmektedir. Çok nöron olması iyidir fakat unutulması gerekmektedir.




**Şekil 3.11.** Katman Seyreltmeden Sonra Doğruluk Değişimi [5]

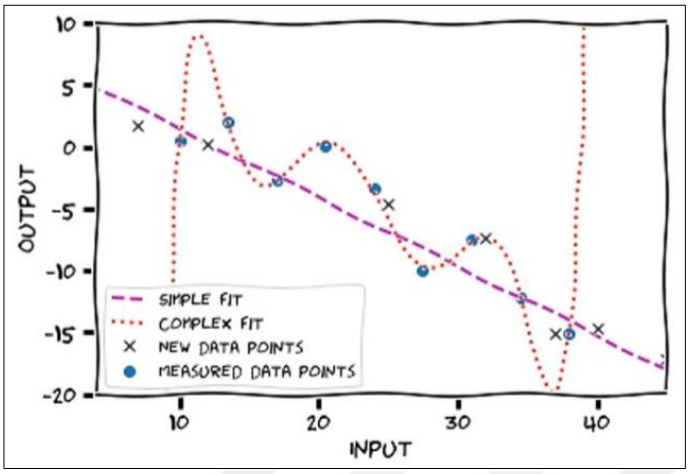



**Şekil 3.12.** Katman Seyreltmeden Sonra Yitim Hatası (Loss) Değişimi [5]

### 3.7. Düğüm Ekleme

Düğüm seyreltme (dropout) işlemine benzer bir işlem yapılmaktadır. Burada da modele gürültüler eklenilmektedir. Bu da nöronları olumsuz etkileyebilir. Olumsuz etkilemesi bazı noktalarda iyidir. Çünkü gerçek dünyada da gürültü vardır. Eğitim seti çok temizdir fakat teste gelecek olan veriler her zaman temiz veya eğitim seti gibi olmayabilir. Bu nedenle eğitim setine gürültüler eklemek gerekir. Örneğin şekil 3.13'de görüldüğü gibi verilen veriler mavi olarak gösterilen noktalardır. Mavi olarak gösterilen noktaları kırmızı eğri ile modellenirse ve bir başka veri eklendiğinde bu model şaşacaktır. Genelleştirilmemiş bir model haline gelecektir. Fakat ekstra yeni veriler eklendiğinde x şeklinde ve buna göre bir hesap yapıldığında daha basit bir fit oluşacaktır ve böylelikle durumu daha düzgün bir hale gelecektir.

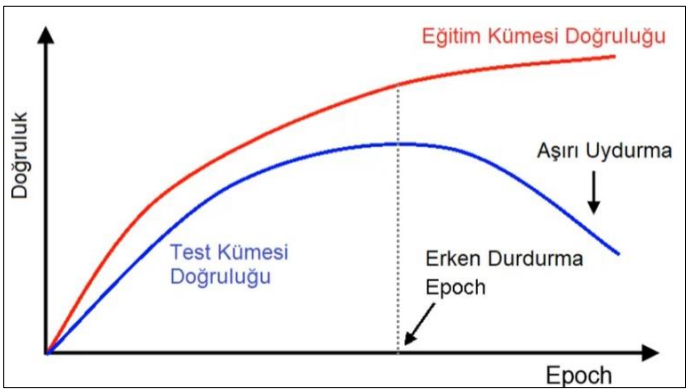
Yitim (loss) fonksiyonuna, modelin içindeki ağırlıklara ya da veriye gürültü eklenerek gerçekleştirilebilir.




**Şekil 3.13** Gürültü Ekleme [15]

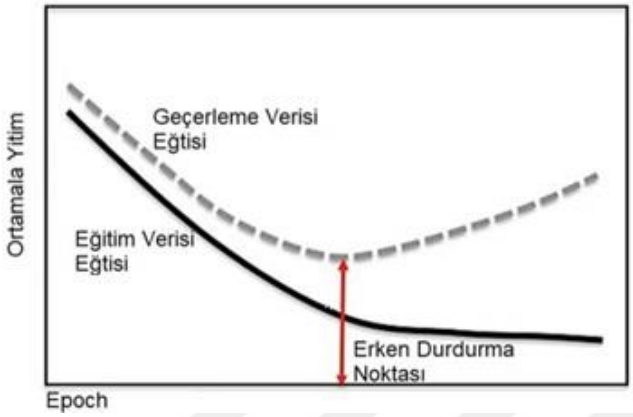
### 3.8. Aşırı ve Az Uydurma (Overfitting-Underfitting)

Bias-varyans ilişkisine çok benzemektedir. Performansın eni iyi olduğu noktayı bulmak gerekir. Bu nokta bulunduğunda çözüme ulaşılmış sayılır. Bunun için sadece eğitim hatasına bakmak en büyük yanlış olur. Bu yüzden geçerleme (test) kümesi kullanılır. Ve test kümesinin performansını eğitim süresince kontrol etmek gerekir. Bu problem erken durdurma (early stopping) işlemidir. Geçerleme verisiyle eğitim verisindeki hata oranına bakılır. Hata oranı düştüğü sürece eğitim devam eder. Belli bir süre arttığı anda eğitim işlemine epoch sayısı fazla verilmiş olsa da durdurmak gerekir. Bu sayede performansın yüksek olduğu nokta bulunur. Şekil 3.14'de görüldüğü gibi hata aralığı arttığı anda erken durdurma işlemi gerçekleşir. Hata aralığı arttıkça model ezberlemeye başlayacaktır ve test doğruluğu azalacaktır.




**Şekil 3.14.** Epoch Sayısının Doğruluğa Etkisi [15]

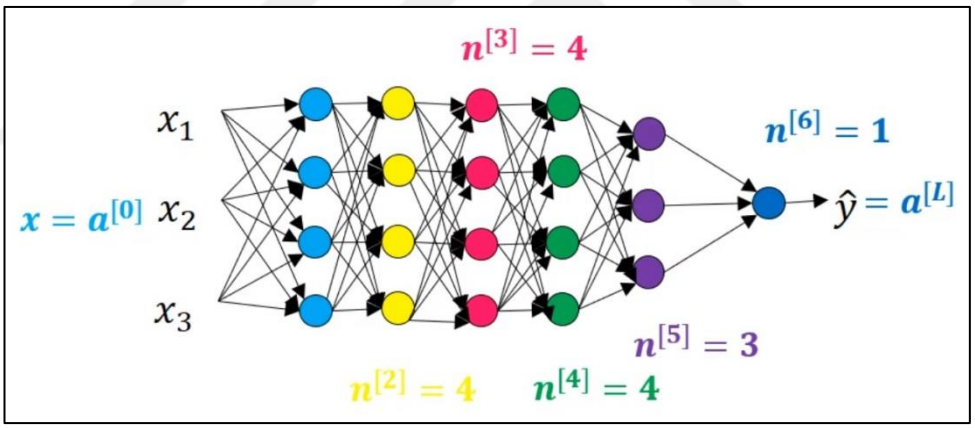
Şekil 3.15’de görüldüğü gibi burada da aynı durum geçerlidir. Ortalama yitim (loss) üzerinden de bakılabilir. Eğitim verisindeki yitim ile geçerleme verisi yitimi arasındaki farka bakılır. Geçerleme verisi eğitim sırasında kontrol edilebilir.




**Şekil 3.15.** Erken Durdurmanın Kayıp Fonksiyonuna Etkisi [15]

# 4. EVRİŞİMLİ SİNİR AĞLARI

Tek katmanlı ve çok katmanlı olmak üzere iki çeşit sinir ağı bulunmaktadır. Tek katmanlı sinir ağlarına sığ sinir ağı denilmektedir. Girdilerden ve tek çıkıştan oluşmaktadır. Yakın geçmişte çok katmanlı sinir ağlarının, az katmanı olan sinir ağlarının öğrenemediği birçok problemi öğrenebilen fonksiyonlara sahip olduğu anlaşılmıştır. Fakat çözüm için ne kadar derinlikte ağ yapısının olacağına karar vermek oldukça zordur. Katman sayısının artması bir takım farklılıklar doğurmaktadır. Her katmanda kenar, köşe ve ışık gibi bir takım öznitelikler öğrenilmektedir. Daha çok özellik öğrenilmesi demek nesne ayırmanın daha kolay olacağı anlamına gelmektedir. [16] Şekil 4.1'de derin sinir ağının yapısına bakılırsa L katman sayısını vermektedir. Girişler katman olarak sayılmamaktadır, $n^{[l]}$ L. katmandaki nöron sayısıdır. Girişlerle ağırlıkların çarpımlar toplamı artı bias değeri z değerini vermektedir. Ve z, aktivasyon fonksiyonuna (g) tabi tutularak a değeri bulunmaktadır.



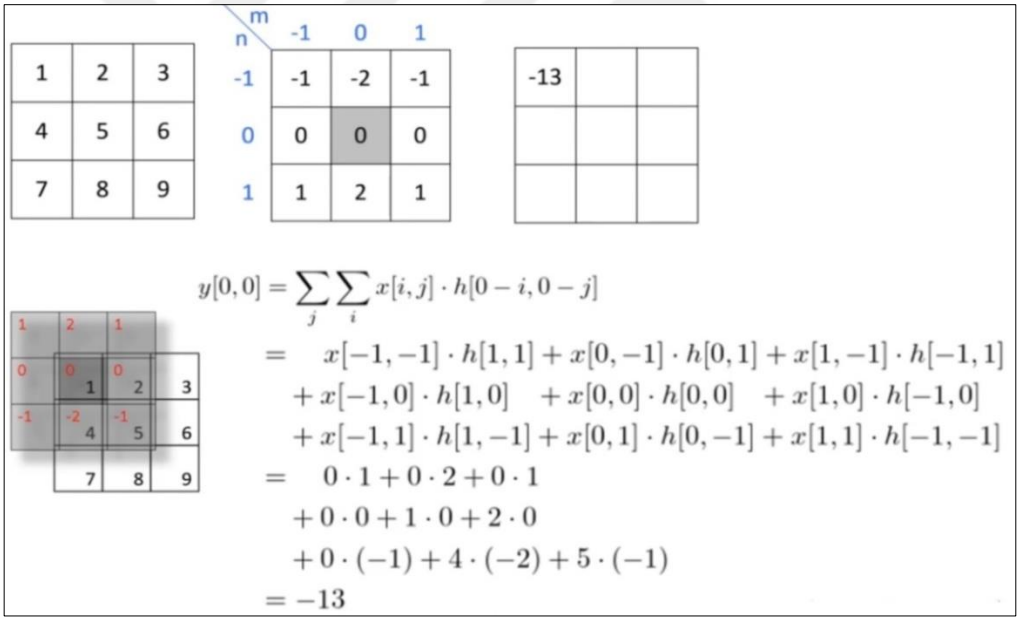

**Şekil 4.1.** Çok Katmanlı Sinir Ağı [18]

## 4.1. Evrişim İşlemi

Evrişim işlemi evrişimli sinir ağlarının temel işlemidir. Aslında evrişim işlemi işaretler sistemler gibi birçok mühendislik alanında gösterilen dersin içeriğinde yapılan evrişim işlemidir. Sadece bunu artık bilgisayarlı görü için 2 boyutlu yapılmaktadır. Çünkü görseller matris olarak ifade edilmektedir. Bunların üzerinde matrissel işlemlerin yapılabilmesi için evrişimin 2 boyutlu tanımlanabilmesi gerekmektedir. Ayrıca derin öğrenmede GPU'ların (grafik işlem birimi) bu kadar ön plana çıkmasının sebeplerinden biride budur. GPU'ların işlem gücünün yüksek

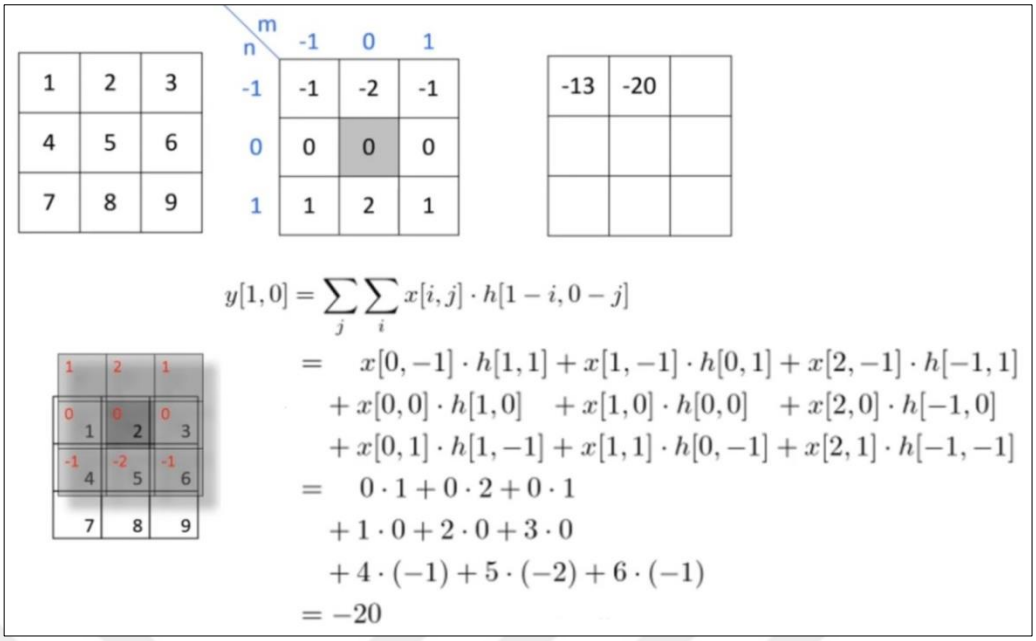
olmasından dolayı matrissel işlemleri daha kolay yapabilmektedirler. GPU'lar sayesinde modeller hızlı bir şekilde eğitilmektedir.

Şekil 4.2.'de 3x3'lük bir matris ve 3x3'lük yatay kenar belirleme filtresi görülmektedir. Bu filtreyle kenar bulma işlemi yapabilmek için gri olarak belirtilen orta noktayı çıkış matrisinde hangi eleman bulunmak isteniliyorsa, giriş matrisinde o elemana çakışacak şekilde işlem yapılmalıdır. Evrişim işleminin formülü de şekil 4.2'de görülmektedir. Giriş x, h ise filtre matrisinin iki eksende (x ve y ekseninde) simetriğinin alınmış halidir. Evrişim işlemine başlayabilmek için filtre matrisinin simetriğinin alınması gerekmektedir. Filtrenin orta elemanını giriş matrisinin ilk elemanı ile çakışacak şekilde çarpma işlemi yapılır. Bu esnada filtre matrisinin 5 elemanı giriş matrisinin hiçbir elemanı ile çakışmadığından boş kalan kısımlar 0 ile çarpılır. Formül tüm elemanlara uygulandığında çıkış matrisinin elde edilen ilk değeri -13'tür.




**Şekil 4.2.** Evrişim İşlemi

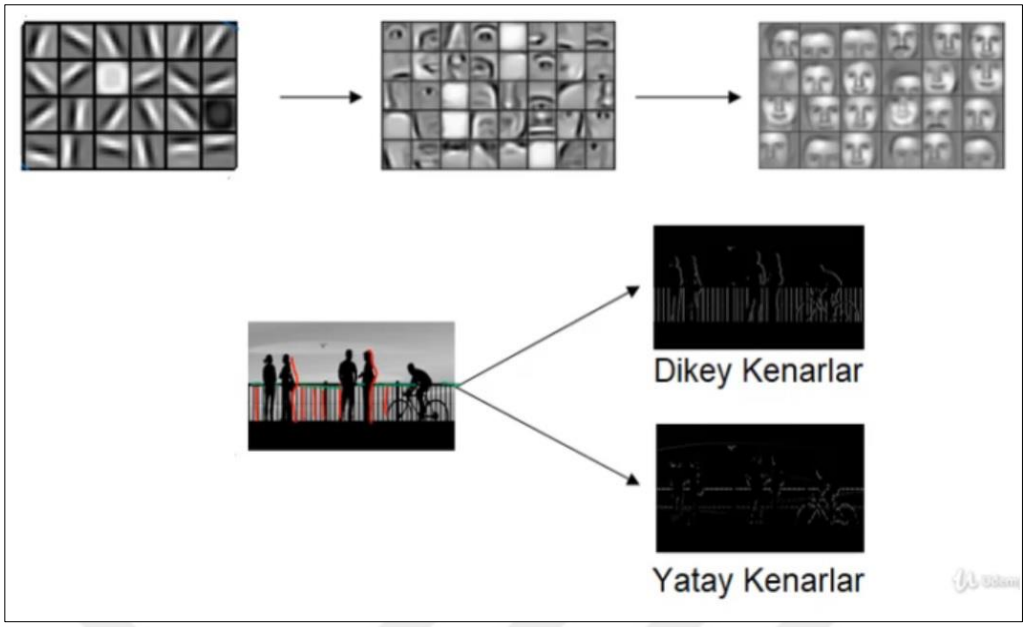
Filtre matrisi şekil 4.3'te görüldüğü gibi adım değeri 1 alarak sağa kaydırıldığında ve formül uygulandığında çıkış matrisinin 2. elemanı -20 çıkacaktır. Böylelikle bir giriş matrisi üzerine filtre matrisi 1'er adım kaydırılarak bütün değerler hesaplama işlemi yapılmaktadır. Evrişim işleminde bu bir yöntemdir. Farklı yöntemler kullanılarak örneğin filtre yerleştirme yöntemi ve adım kaydırmayı değiştirerek evrişim işlemleri yapılmaktadır.



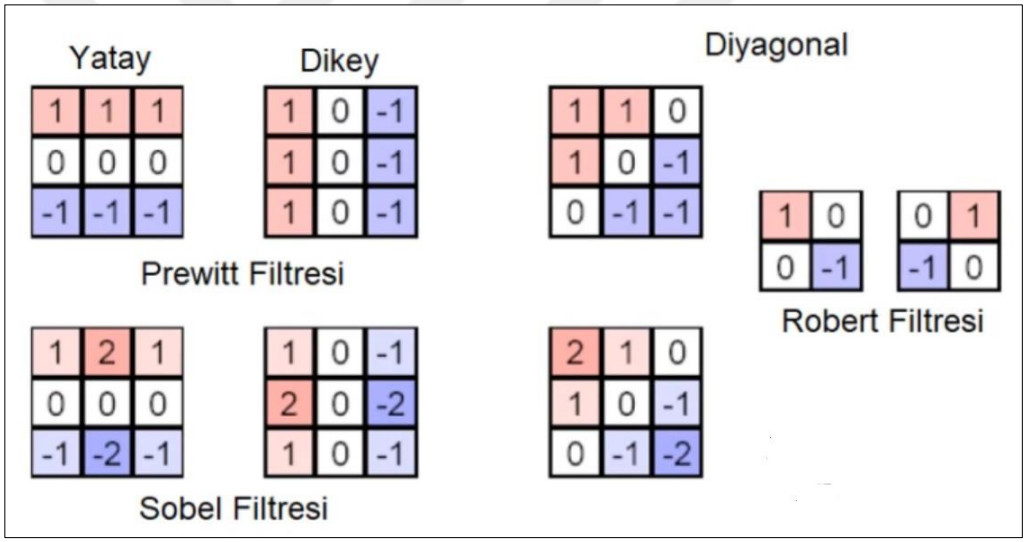

**Şekil 4.3.** Evrişim İşleminde Padding

### 4.1.1. Kenar Bulma Yöntemi

Görüntü üzerinde temel bilgileri elde etmek için kullanılır. Kenar bilgileri görüntülerden elde edilen öznitelikler içerisinde en çok ihtiyaç duyulanlardandır. Giriş bilgisinin yüksek frekanslı bölgelerini simgelemektedir. Bu bilgileri elde etmek için dikey yatay ya da farklı rotasyonlarda birtakım filtre görüntü üzerinde işleme alınmaktadır. Bu işlem sonrasında yüksek frekanslı yeni görüntünün kenar bilgileri belirlenmiş olur. Geleneksel yöntemlerde Sobel, Prewitt ve Robert gibi manuel olarak oluşturulan matrislerle yaparken, derin öğrenmede özellikle evrişimli sinir ağlarında bu işlemleri yapmaya gerek yoktur. Şekil 4.4'de görüldüğü gibi bu işlemleri yapmaya gerek yoktur çünkü ilk katmanlarda özellikle bu bilgiler evrişimli sinir ağında otomatik olarak çıkarılmaktadır. Şekil 4.5'de görüldüğü gibi geleneksel filtrelerin matrissel gösterimleri görülmektedir. Eski görüntü işleme teknikleriyle görüntünün bazı özelliklerini çıkarmaya çalışırken dikey-yatay kenarlar bulunduktan sonra birleştirilme işlemiyle nesnenin dış şekli parça parça elde edilmekteydi.



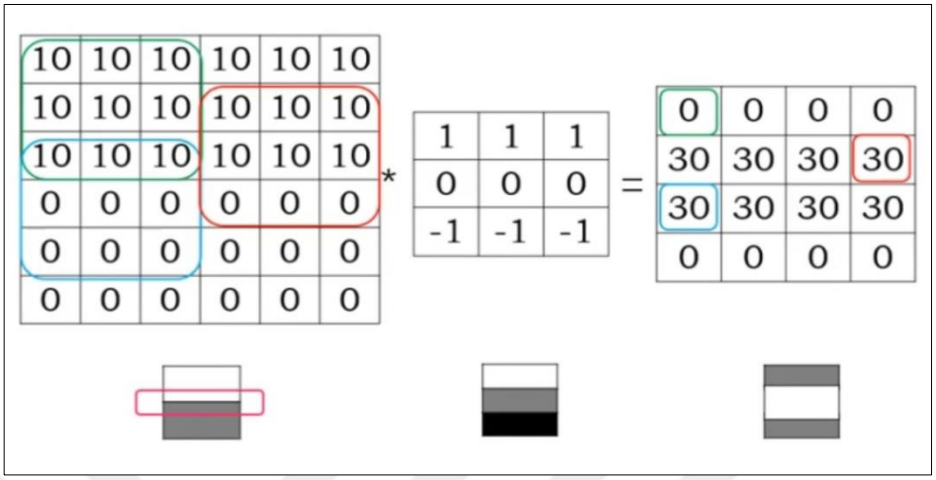

**Şekil 4.4.** Kenar Bulma İşlemleri [13]




**Şekil 4.5.** Kenar Bulma Filtreleri [13]

Örneğin şekil 4.6'daki özel bir matriste 10 ve 0'ın birleştiği nokta kenar bilgisi vermektedir. Çünkü yüksek frekanslı bölge demek hızlı geçişin olduğu bölge demektir. Şekil 4.6'da dörtgenlerde 10 ile ifade edilen beyaz olan bölgeyi, 0 ile ifade edilen daha koyu renkli bölgeyi göstermektedir. Ve ikisinin birleştiği çerçeveye alınan kısım kenar bölgesini ifade etmektedir. Kenar bulma filtresiyle giriş matrisi evrişim işlemine tabi tutulduğunda elde edilen çıkış matrisinde kenar yani yüksek frekanslı bölge görülmektedir. Burada dikkat edilirse filtre matrisi ile giriş matrisi birebir üst üste geldiğinden çıkış matrisi giriş matrisinden daha küçük boyutlu bir matris olmaktadır. Böylece evrişim işlemi ile görüntünün yüksek frekanslı

bölgelerini simgeleyen kenar bölgelerini hesaplamak mümkün olmaktadır. Buradaki filtre yatay kenar bulan bir filtredir. Dikey veya diagonal bir filtre olabilir fark etmez evrişim işlemine tabi tutulduğunda istenilen bilgiyi verecektir.




**Şekil 4.6.** Kenar Belirleme

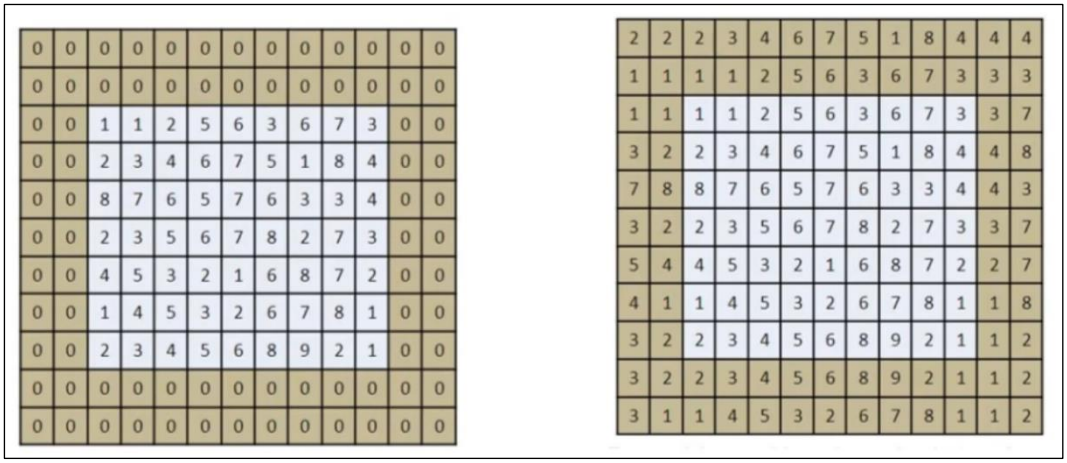
### 4.1.2 Piksel Ekleme (Padding)

Çıkış işaretiyle giriş işareti arasında oluşan boyut farkını yönetmek kullanıcın elindedir. Eğer giriş matrisi (görüntünün) ile çıkış matrisinin aynı boyutlarda olması isteniliyorsa giriş matrisine ekstra pikseller eklemek (padding) gerekmektedir. Literatürde piksel ekleme dolgulamak olarak da geçmektedir. Şekil 4.7'deki giriş matrisinde beyaz renkle gösterilen kısımdır. Giriş matrisinin çevresine hayali sıfırlar eklenerek yeni matris oluşturulur. Ne kadar 0 ekleneceğini 20 numaralı formülde hesaplanmıştır. Giriş matrisi ( n x n ) = ( 6 x 6 ), kullanılacak filtrede ( f x f ) = ( 3 x 3 )'lük bir matristir.

Piksel ekleme varsa → çıkış = ( n + 2p – f + 1) x ( n + 2p – f + 1 ) = ( 6 x 6 ) (20)

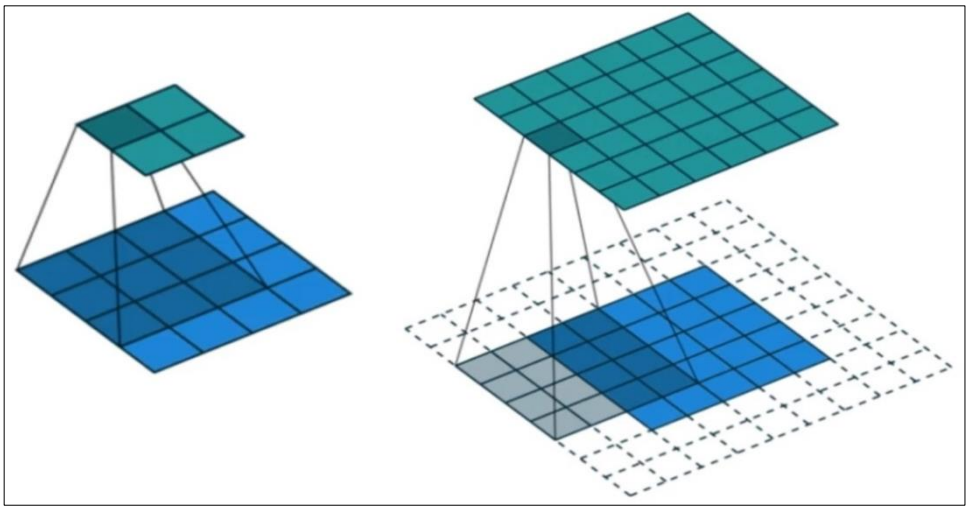
Piksel ekleme yoksa → çıkış = ( n – f + 1 ) x ( n – f + 1 ) = ( 4 x 4 )

$$p = \frac{(f-1)}{2}$$




**Şekil 4.7.** Piksel Ekleme (Padding) Sonucu Oluşan Çıkış Matrisi

Eğer piksel ekleme yapılacaksa; ekleme işlemi iki türlü yapılabilir. Padding adedi kadar 0 eklenebilir veya giriş matrisinin ilk iki satır-sütun değerlerinin simetrisi alınabilir. Ya da tamamı en dış çerçevedeki değerlerin kopyası olabilir. Evrişim işlemi yapılmaya başlandığında eğer köşeler sıfırlarla doluysa çıkış matrislerinin dış kısımlarındaki değerler çarpma işlemi yapıldığından sürekli aşağı çekilecektir. Ve sıfıra çekildiğinden dolayı elde edilen çıkış matrisinde ayrık değerler oluşacaktır. Ama görüntüdeki piksellerin dışa doğru kopyalanması işlemi kullanılırsa dışarıya eklenen pikseller görüntüye aitmiş gibi olduğu için yapılan evrişim işlemi çıkışında da mutlaka görüntüye dair bilgilerden oluşan bilgilere yakın çıkışlar elde edilecektir. Dolayısıyla daha hassas bir çıkış matrisi elde edilmiş olunacaktır. Bunun dezavantajı her adımda işlem yapılacağı için işlem yükü bakımından biraz daha yavaş çalışan evrişim işlemi olacaktır.



**Şekil 4.8.** Piksel Ekleme (Padding) Simülasyonu

Şekil 4.8'de görüldüğü gibi padding uygulanmadığında giriş matrisi 4x4 iken çıkış matrisi 2x2 olacaktır. Padding uygulandığında eklenen piksel sayısına bağlı olarak çıkış matrisi giriş matrisine eşit veya daha büyük olacaktır.

### 4.1.3. Adım Kaydırma (Stride)

Sinir ağına uygulanacak filtrenin kaydırılacak piksel boyutudur. Bu işlemde filtreyi görüntü matrisi üzerinde evrişim işlemine alırken filtreyi kaç piksel aralıklarla kaydırılacağının piksel değeridir. Aşağıdaki örnekte açıklamak gerekirse; 9x9'luk bir matrise 3x3'lük bir filtre uygularken normalde filtre 1'er adım aralığıyla uygulanır. Adım kaydırma aralığı 2 seçildiğinde oluşacak olan çıkış matrisinin boyutu da değişmektedir. Adım aralığı fazla olduğunda bilgi kaybı daha fazla olacak ve boyut biraz daha küçülecektir. Yani adım aralığıyla (stride) çıkış matrisinin boyutu birbirine ters orantılıdır. Şekil 4.9'daki 9x9'luk matrisin üzerine 2 adım aralığı olan 3x3'lük bir filtre uygulandığında sonuç 5x5'lik bir çıkış matrisi olacaktır.

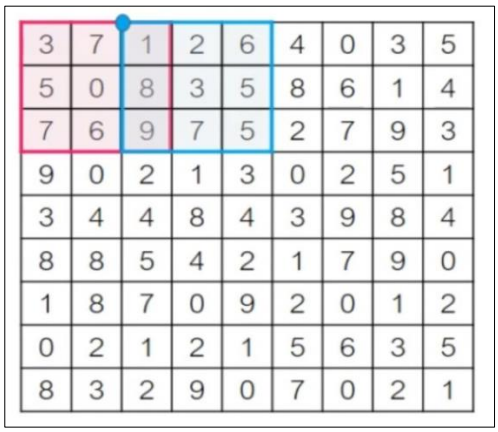
$s = 2 \rightarrow$ Adım kaydırma (stride)

$$(\frac{(n+2p-f)}{s} + 1)x(\frac{(n+2p-f)}{s} + 1) \quad (21)$$

$n = 9x9n = 3x3$

$$(\frac{(9+2.0-3)}{3} + 1)x(\frac{(9+2.0-3)}{2} + 1)$$

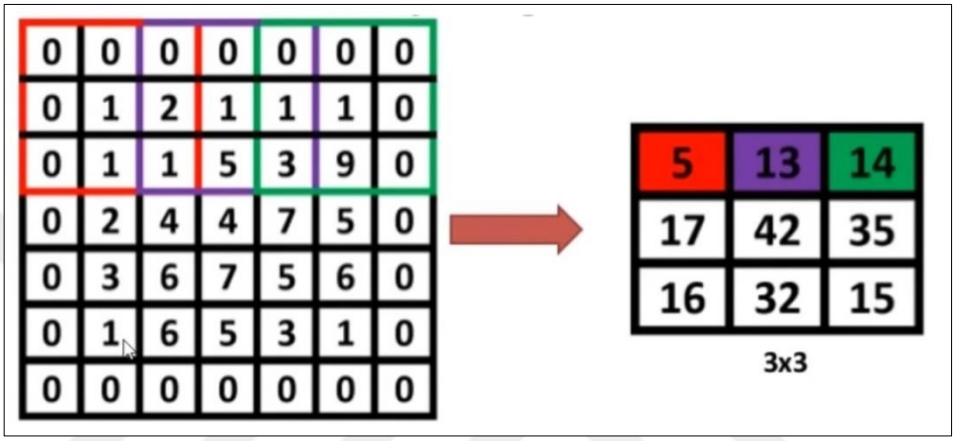
$= (5x5)$ Çıkış matrisin boyutu




**Şekil 4.9.** Adım Kaydırma (Stride) İşlemi

Şekil 4.10'daki örnekte giriş matrisi 5x5, filtre 3x3, adım kaydırma (stride) 2 ve bir çerçeve de sıfırlardan oluşan piksel ekleme işlemi (padding) uygulanmıştır. Bütün işlemler yapıldığında 3x3'lük bir çıkış matrisi oluşmaktadır. Görüldüğü gibi piksel ekleme (padding) yapılmış ve adım kaydırma 1 alınmış ve çıkış matrisinin boyutu giriş matrisinin boyutundan küçük çıkmıştır. O halde padding daha farklı seçilmelidir.

$$n = 9x9,\ f = 3x3,\ s = 2,\ p = 1 \tag{22}$$




**Şekil 4.10.** Stride Değeri 2 ve Padding Değeri 1 Olan Evrişim işlemi

$$\left(\frac{(n+2p-f)}{s}+1\right)x\left(\frac{(n+2p-f)}{s}+1\right) \tag{23}$$

$$=\left(\frac{(5+2-3)}{2}+1\right)x\left(\frac{(5+2-3)}{2}+1\right)=3x3$$

Aynı durum için padding 3 seçilirse; sonuç 5x5 çıkacaktır. Eğer adım kaydırma (stride) büyük seçilip giriş matrisi ile çıkış matrisi arasında boyut farkı olması istenilmiyorsa piksel doldurma da (padding) büyük seçilir.

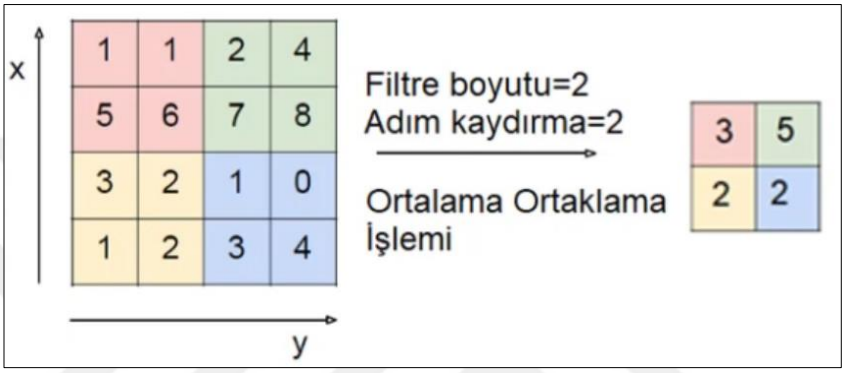
$$\left(\frac{(5+2.3-3)}{2}+1\right)x\left(\frac{(5+2.3-3)}{2}+1\right)=5x5 \tag{24}$$

Gerçek hayatta genellikle renkli görüntüler kullanılmaktadır. Renkli görüntülerde bu şekilde ifade edilmesi zordur. Çünkü kırmızı-yeşil-mavi olmak üzere 3 farklı kanal vardır.

### 4.1.4. Ortaklama İşlemi (Pooling Operation)

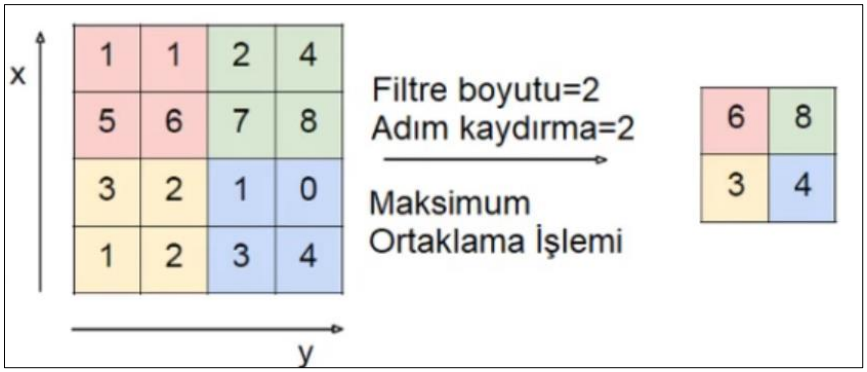
Ortaklama (Pooling) işlemini şekil 4.11 ve şekil 4.12'de açıklamak gerekirse; şekil 4.11'deki matrise 2x2 boyutlu bir filtrenin adım kaydırması 2 seçilmiştir. Daha sonra

bu matris üzerine filtre ve adım kaydırma boyutuna göre ortaklama (pooling) işlemi yapılmaktadır. Ortaklama (pooling) işlemi yaparken iki yöntem vardır. Bunlardan birisi ortalama ortaklamadır. İlgili adımda 2x2 alana denk gelen piksellerin değerlerinin toplamını alarak piksel sayısına bölünmektedir yani ortalama işlemi yapılmaktadır ve çıkan değer çıkış matrisine yansımaktadır. Filtre boyutu büyüdükçe ifade edilen alanda büyüyecektir ve görüntüye dair hassasiyet azalacaktır. Ve tüm matris boyunca adım kaydırma yapılarak diğer piksellerin ortalama değeri çıkış matrisine yansıtılmaktadır.




**Şekil 4.11.** Ortalama Ortaklama (Pooling) İşlemi [5]

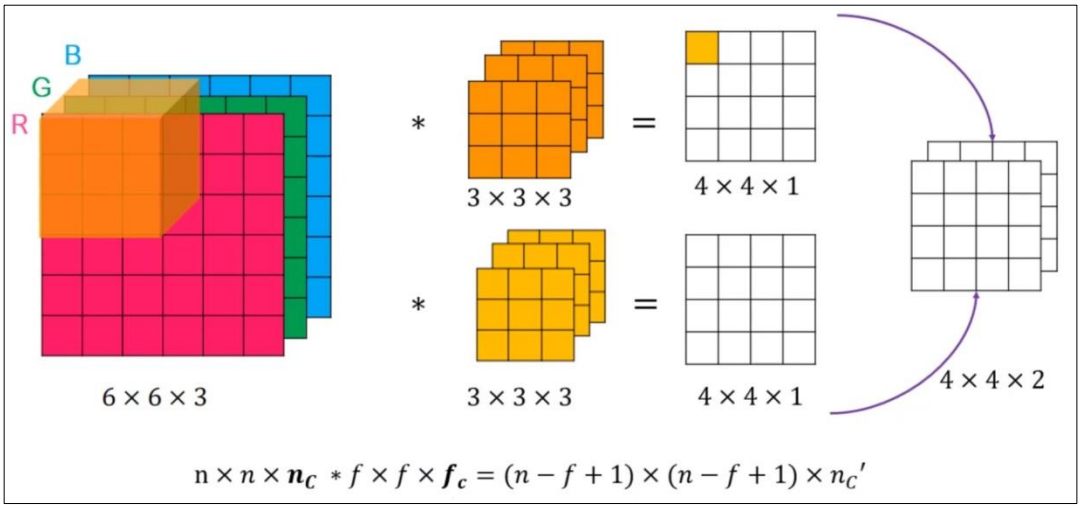
Bir başka ortaklama işlemi maksimum pooling denilen ortaklama işlemidir. Burada ortalama işlemi yerine filtre alanındaki en yüksek piksel değeri alınır. Dikkat edilirse ortalama ortaklama ile maksimum ortaklama arasındaki çıkış matrislerinde çok büyük fark yoktur. Burada esas amaç boyut azaltarak işlem yükünü düşürmektir. Bu da kullanılan ara katmanlardan biridir.




**Şekil 4.12.** Maksimum Ortaklama (Pooling) İşlemi [5]

### 4.1.5. Çok Kanallı Evrişim İşlemi (Multi Channel Convolution Operation)

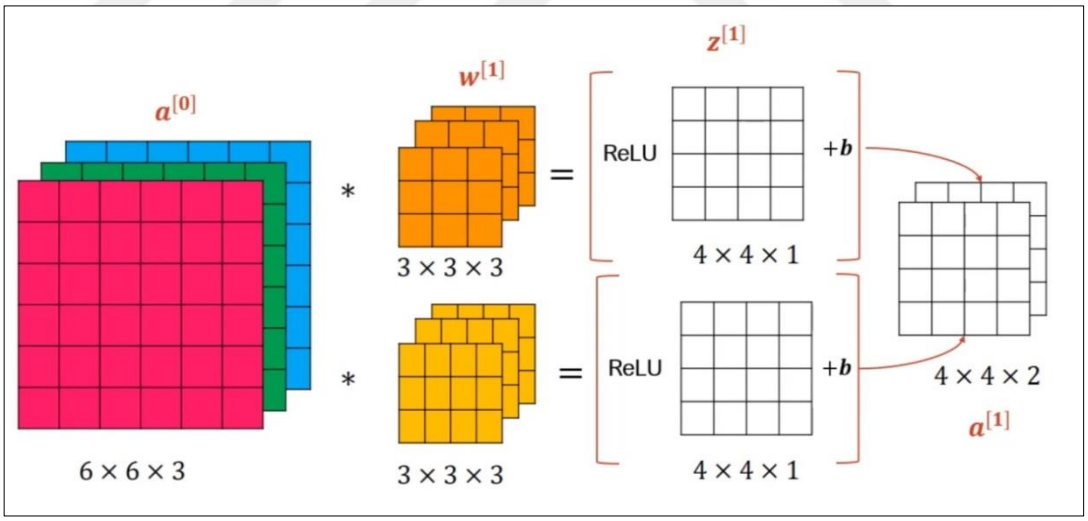
Evrişim işlemini yaparken dikkat edilmesi gereken adım kaydırma (stride), piksel ekleme (padding), ortaklama (pooling) işlemlerinin iki boyutlu (tek kanallı görüntüler) matrislerde nasıl yapıldığı önceki konularda değinildi. Fakat evrişim işlemleri genellikle çok kanallı görüntüler üzerinde yapılmaktadır. Çok kanallı görüntülerin en basit hali kırmızı, yeşil ve mavi renklerinden veya renk tonlarından (red-green-blue) oluşan görüntü tipidir. Eğer kullanılan görüntü örneğin 6x6 boyutunda, renkli görüntüyse (6x6x3) şeklinde kanal sayısını da sona eklenilmelidir. Böyle bir görüntüye bir evrişim işlemi uygulamak gerekirse filtrelerinde 3 kanallı olması gerekmektedir. Şekil 4.13'deki görüntüye evrişim işlemi uygulandığında her filtre katmanı kendi katmanı ile çarpılarak toplama işlemi yapılacaktır. Ve çıkışta tek bir değer ifade edilmektedir. Aynı işlemler alttaki filtre içinde yapılmaktadır. Burada ne kadar filtre kullanılıyorsa o kadar öz nitelik çıkarılacaktır. Örneğin birinci filtre yatay kenar, ikinci filtre dikey kenar bulmak için tasarlanmış filtreler olabilir. Dikkat edilirse filtre çıkışlarındaki elde edilen görüntü matrisi değeri 4x4x1'dir. Görüntü matrisinin 4x4x1 olmasının sebebi adım kaydırmanın (stride) 1 olduğundan ve piksel ekleme (padding) olmadığındandır. Bu işlemlerin sonucunda 4x4x2'lik bir çıkış matrisi elde edilmektedir. Çok kanallı matrise tensör denilmektedir. Her filtre birbirinden bağımsız işleme alınmaktadır ve sonra çıkışta birleştirilmektedir. Çıkışların birleştirilmiş hali bu işlemlerin tamamının sonucu olmaktadır. 4x4x2'lik çıkış tensörü bir görüntüyü simgelemiyor olabilir ama sonuçta bir öznitelik çıkarma işlemi yapılmaktadır.




**Şekil 4.13.** Çok Kanallı Evrişim İşlemi [18]

### 4.1.6. Tek Katmanlı Evrişimli Sinir Ağı

Bir sinir ağında çıkışın hesaplanması için giriş katmanı, ağırlıklar, aktivasyon fonksiyonu ve bir bias değerine ihtiyaç vardır. Temel olarak mantık aynıdır sadece giriş yani x değerleri bir vektör değil bir görüntü matrisini simgelemektedir. Ve bu görüntü matrisi çok kanallı da olabilmektedir. Bu yüzden de işlem yükü çok fazla artmaktadır. Şekil 4.14'de görüldüğü gibi 3 kanallı bir görüntüde 6x6x3 boyutlu ve 2 adet 3x3x3 boyutlu filtre bulunmaktadır. Bir evrişim işlemi yapıldığında 4x4x1 boyutlu birer matris elde edilmektedir. Elde edilen çıkış matrislerini bir aktivasyon fonksiyonuna tabi tutulduktan sonra bir bias değeri eklenmektedir. Sinir ağımızın denkleminde ağırlıkların (w) ve girişlerin (x) çarpımlar toplamına bir bias değeri eklenerek z değeri elde edilmektedir. Ve bu z değeri bir aktivasyon fonksiyonuna (g) sokularak sonuçta a değeri elde edilmektedir. Burada ki mantıkta aynı mantıktır. Burada filtrelere parametre bakımından bakılırsa 3x3x3'ten 27 tane ve 1 tanede bias değeriyle 28 parametre hesaplanmaktadır. Görüntü daha büyük olsa dahi filtre boyutu aynı olduğu sürece parametre sayısı aynı olacaktır. Yani parametre sayısı filtre boyutuna bağlıdır.

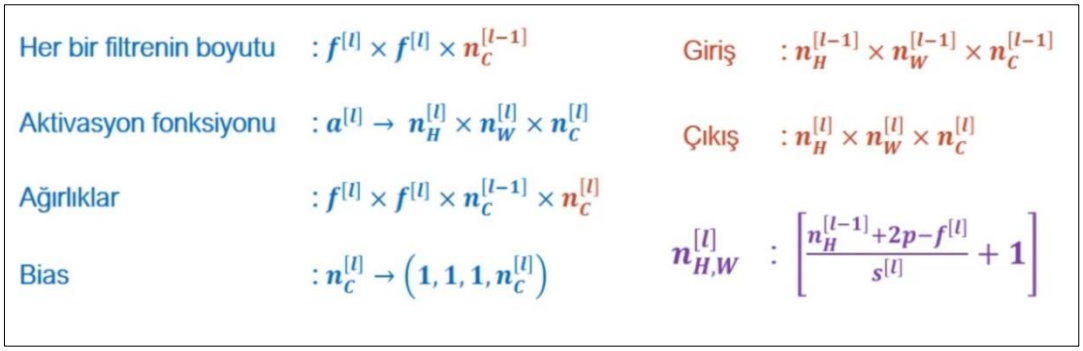



**Şekil 4.14.** Çok Kanallı Evrişim İşlemi ve Aktivasyon Fonksiyonunu [18]

$$z^{[1]}=w^{[1]}.a^{[0]}+b^{[1]}$$

$$a=g^{[1]}(z^{[1]})$$

Filtre boyutu f, piksel doldurma (padding) p, adım kaydırma (stride) s, filtre (kanal) sayısı $n_c$ olarak ifade edilmektedir. Filtre kanal sayısı ile giriş görüntüsü kanal sayısı eşit olmak zorundadır.




**Şekil 4.15.** Filtre Hesabı

Bir evrişimli sinir ağında, evrişim katmanı (aktivasyon fonksiyonu, bias değeri), ortaklama katmanı (maksimum ya da ortalama ortaklama), tam bağlantılı katman (klasik yapay sinir ağı bağlantıları) olmak üzere 3 adet katman veya katman grubu bulunmaktadır. Bu katmanları veya işlemleri görselleştirilirse; Şekil 4.16'da giriş 27x27x3 boyutunda bir görüntüden oluşmaktadır. Filtre boyutu 3x3x3, piksel doldurma (padding) 0, adım kaydırma (stride) 1'dir. Ve aynı boyutta 10 adet filtre bulunmaktadır. 25 numaralı formülde bu değerler uygulanırsa;

$$\left(\frac{(n+2p-f)}{s} + 1\right) x \left(\frac{(n+2p-f)}{s} + 1\right) \quad (25)$$

((n+2.p-f)/s)+1 → ((27+2.0-3)/1+1=25

1. katmandaki tensör (matris dizisi) 25x25x10 olacaktır. İkinci evrişim katmanına geçerken filtre boyutu 5x5x5, piksel ekleme (padding) 0, adım kaydırma (stride) 2 ve filtre sayısı 20 alınırsa;

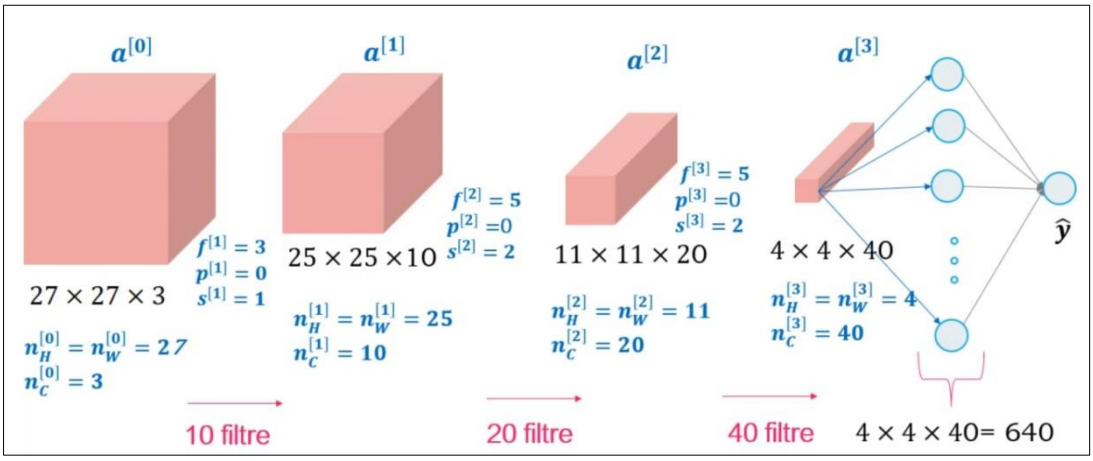
((n+2.p-f)/s)+1 → ((25+2.0-5)/2+1=11 (26)

2. katmandaki tensör (matris dizisi) 11x11x20 olacaktır. Burada dikkat edilmesi gereken nokta her bir matrisi elde ederken bir aktivasyon fonksiyonu ve bias değeri kullanılmaktadır. Bir sonraki katmana geçerken filtre boyutu 5x5x5, piksel ekleme (padding) 0, adım kaydırma 2 ve filtre sayısı 40 olarak alınırsa;

((n+2.p-f)/s)+1 → ((11+2.0-5)/2+1=4 (27)

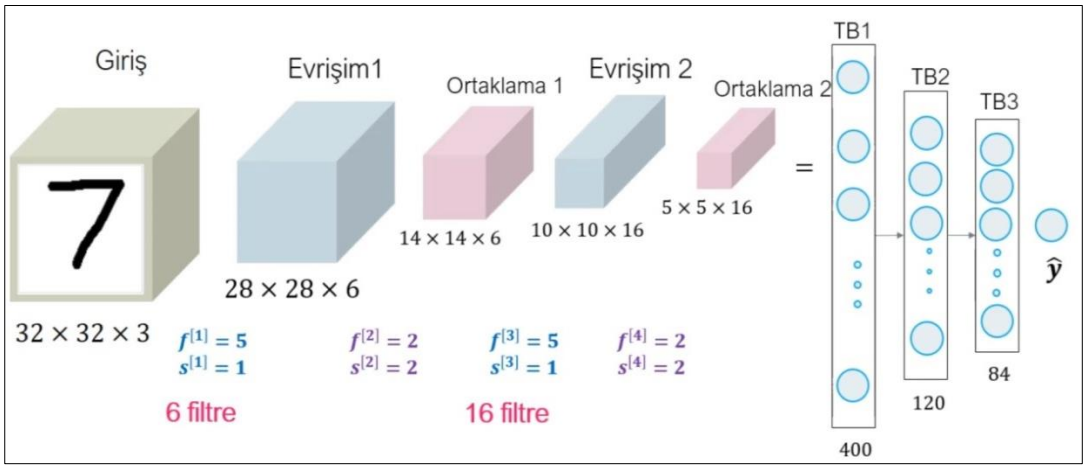
3. katmandaki tensör (matris dizisi) 4x4x40 olacaktır. Eğer bu katman sonuncu katmansa, katman vektörel hale getirilir. Çünkü çıkışta tam bağlantılı katmanı yapılmalıdır. Elde edilen vektör 640 satırlı bir vektör olacaktır. 640 satırlı vektörde

boyut azaltma işlemleri yaparak çıkışta bir kestirim (loss) değeri oluşacaktır. Örneğin bir nesne sınıflandırma yapılacaksa sınıflandırma yüzdesi olacaktır.




**Şekil 4.16.** Evrişim Hesap İşlemeleri

Şekil 4.17'de örnekte gibi 32x32x3 boyutlu bir giriş görüntüsü bulunmaktadır. Bu görüntüye 6 adet 5x5x5 boyutlu filtre, piksel ekleme (padding) 0 ve 1 adım kaydırma (stride) uygulanmaktadır. Şekil 4.15'deki formüle 1. katman 28x28x6 olarak çıkacaktır. Dikkat edilirse ortaklama işlemi yapılırken kanal sayısı azalmamaktadır. Ortaklama (pooling) işlemi yaparken sadece boyut azaltma işlemi yapılmaktadır. Son ortaklama (pooling) işleminden elde edilen 5x5x16'lık tensör, 400 satırlık tam bağlantılı katman (fully connected layer) haline gelmektedir. Kestirim değeri 10 sınıflı olarak softmax fonksiyonu ile çıkarılabilir. Genel olarak evrişimli sinir ağı bu şekilde katmanlardan oluşmaktadır. Evrişim ve ortaklama (pooling) işlemlerini çeşitli kombinasyonlarla modellemek tasarlamak mümkündür.



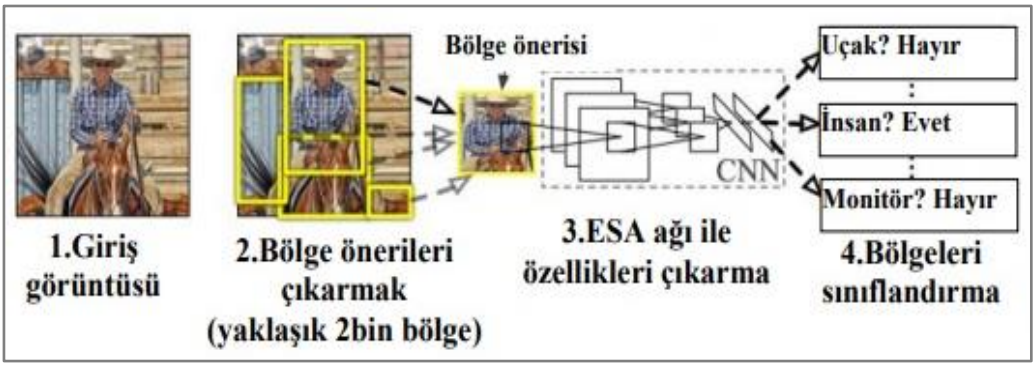

**Şekil 4.17.** Sinir Ağı Modeli ve Yapılan Evrişim İşlemi

### 4.2. R-CNN ve Fast R-CNN Modelleri

Görüntü sınıflandırma işlemi, genellikle bir görüntüdeki nesnenin tahmini üzerine çalışmaktadır. Nesnenin görüntünün neresinde olduğu ve kapladığı sınırların tespiti ise nesne tanımlama işlemidir. Nesne tanımlama için derin öğrenmede başlıca R-CNN, Fast R-CNN, Faster R-CNN gibi modeller bulunmaktadır.

#### 4.3.1. R-CNN (Region Convolutional Neural Network) Bölgesel Evrişimli Sinir Ağı Modeli

Bu mimari Şekil 4.18'de gösterildiği gibi başlıca 4 bölümden oluşmaktadır. Birinci bölümde görüntüler alınır. İkinci bölümde ise seçici arama ile bölge önerileri yapılır. Seçici arama, bir nesneyi ihtimali yüksek olan 2000 farklı bölge üretme işlevi gerçekleştirir. Üçüncü bölümde her bir bölge önerisi AlexNet'e benzer şekilde tasarlanmış evrişimli sinir ağı mimarisine verilir. Son olarak evrişimli sinir ağı çıktısı eğer bir nesne ise seçici arama ile belirlenen bölge üzerinde bir düzenleme yapılarak nihai sonuç üretilir. R-CNN bazı dezavantajları vardır. Öncelikle her bir görüntüdeki her bölge önerisi için evrişimli sinir ağından ileri besleme gerektirmekte (görüntü başına 2000 ileri besleme). Bu test aşamasında zaman kaybına sebep olmaktadır. İkinci olarak mimari ayrı ayrı üç farklı modeli eğitmek zorundadır. Bunlar, görüntü özelliklerini oluşturmak için evrişimli sinir ağı, nesnenin sınıfını tahmin eden sınıflandırıcı ve sınırlayıcı çerçeveleri sıkıştırmak için regresyon modeli. Bu işlem zordur ve ortalama görüntü başına 50 saniye gibi bir zaman almaktadır.

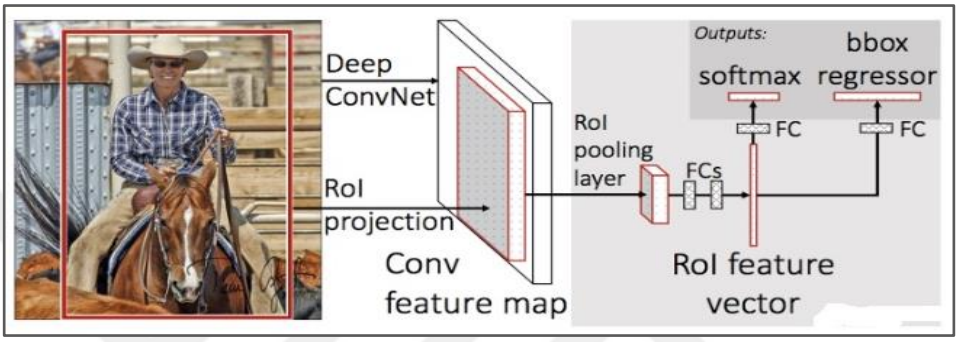



**Şekil 4.18.** R-CNN Mimarisi [9]

#### 4.3.2. Fast R-CNN (Fast Region Convolutional Neural Network)

R-CNN'deki sorunların üstesinden gelmek için Fast R-CNN (Girshick 2015) modeli geliştirilmiştir. Bu mimaride öncelikle her bir görüntü için yaklaşık 2000 kez sinir ağı çalıştırmak yerine tek bir kez sinir ağı çalıştırıp 2000 önerinin üst üste gelen

bölgeleri arasında hesaplama paylaşımı yapılmaktadır. Bundan dolayı R-CNN'ne göre hızı artmaktadır. RoI(region of interest) ortaklama katmanı tek tip olmayan boyutlardaki girdilerde maksimum ortaklamayı gerçekleştiren ve sabit boyutlu küçük bir özellik haritası üreten bir tür ortaklama katmanıdır. Bu sabit boyutun seçimi bir ağ hiperparametresidir ve önceden tanımlanmıştır. Bu tür bir ortaklama yapmanın temel amacı, eğitim ve test zamanını hızlandırmak Bu ortaklama katmanının kullanımı nedeniyle, eğitim ve test süresi orijinal R-CNN mimarisine göre daha hızlıdır.




**Şekil 4.19.** Fast R-CNN [9]

# 5. PLAKA TANIMA SİSTEMİ

Bu bölümde görüntü işlemede sık kullanılan derin öğrenme mimarilerinden bölgesel evrişimli sinir ağı (R-CNN) ile araç plaka lokasyonunun belirlenmesi için yapılan çalışma bulunmaktadır. Burada ilk aşamada araç plaka konumu belirleme işlemi yapılmaktadır, ikinci aşamada ise konumu belli olan plakadan karakterlerinin görüntü işleme teknikleri ve OCR ile ayrı ayrı okunmasıdır.

## 5.1. Evrişimli Sinir Ağı Tasarımı

İlk aşamada bölgesel evrişimli sinir ağı (R-CNN) kullanılmıştır. Görüntü üzerinde plakanın konumunun belirleme işlemi yapılmıştır. Şekil 1.30'da görüldüğü gibi sinir ağının katmanları gösterilmektedir.

Giriş katmanından sonra 2.katmanda 3x3'lük 8 adet filtre ile adım kaydırma (stride) ve piksel ekleme (padding) değeri 1 olacak şekilde evrişim işlemi parametreleri bulunmaktadır.

3. katmanda batch normalizasyon işlemi yapıldıktan sonra 4. katmanda aktivasyon işlemi için ReLu fonksiyonu bulunmaktadır.

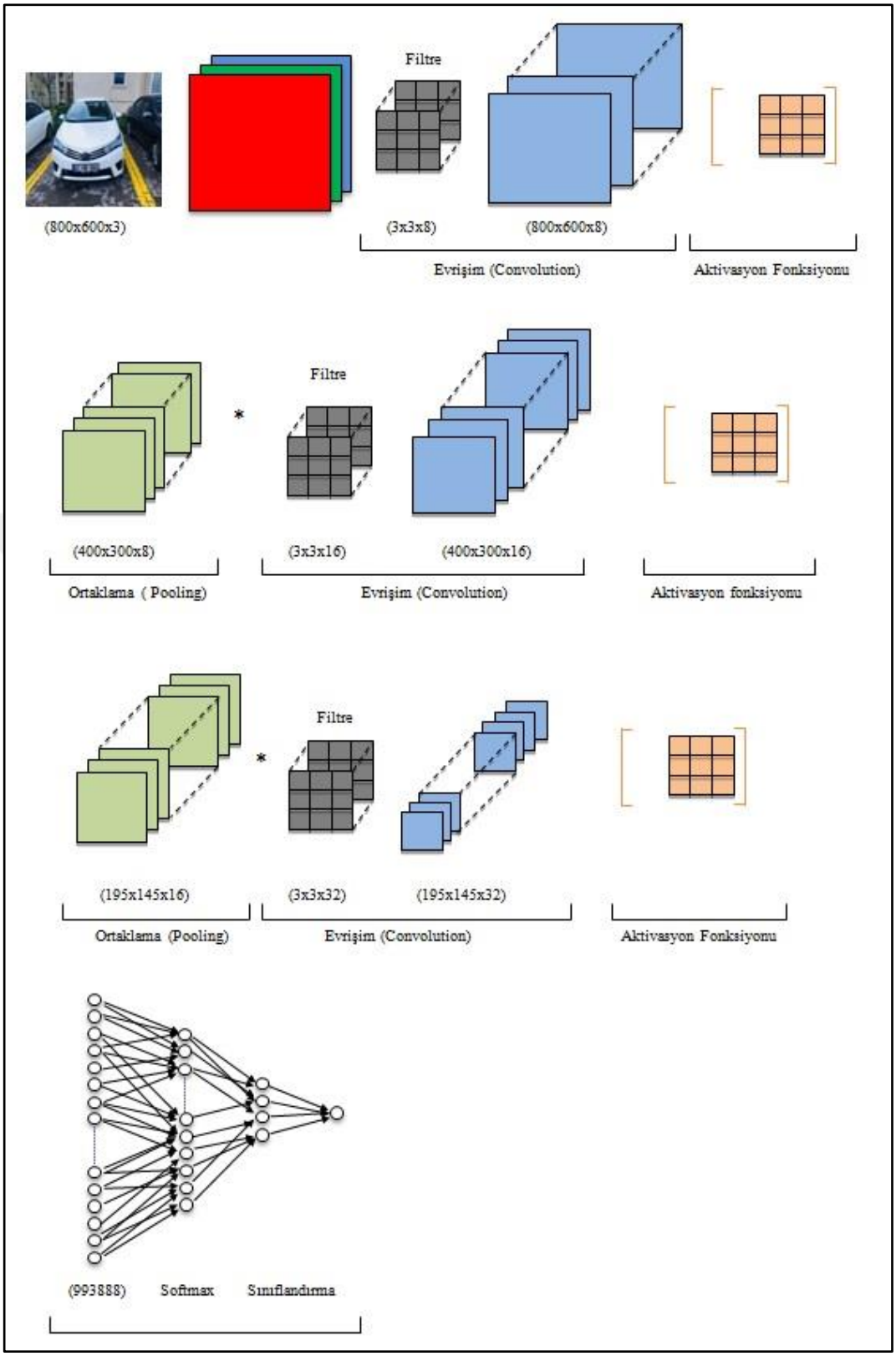
5. katmanda adım kaydırma (stride) ve piksel ekleme (padding) 2 olacak şekilde ortalama ortaklama (pooling) işlemi bulunmaktadır.

6. katmanda 3x3'lük 16 adet filtre ile adım kaydırma (stride) ve piksel ekleme (padding) değeri 1 olacak şekilde evrişim işlemi bulunmaktadır.

7. ve 8. katmanda batch normalizasyon ve aktivasyon işlemleri yapıldıktan sonra 9. katman için adım kaydırma (stride) ve piksel ekleme (padding) 2 olacak şekilde ortalama ortaklama (pooling) işlemi bulunmaktadır.

10. katman için 3x3'lük 32 adet filtre ile adım kaydırma (stride) ve piksel ekleme (padding) değeri 1 olacak şekilde evrişim işlemi bulunmaktadır.

11. ve 12. katmanda batch normalizasyon ve aktivasyon işlemlerinden sonra klasik sinir tam bağlantılı katman bulunmaktadır. Ve son katmanlarda ise softmax ile sınıflandırma işlemleri bulunmaktadır.




**Şekil 5.1.** Plaka Tanıma Eğitim Katmanları

### 5.2. Kod Mimarisi

Kod mimarisi iki aşamadan oluşmaktadır. Birinci aşamada R-CNN ile önce ağ eğitimi sonra ağ eğitim parametrelerine göre test verilerinden görüntü üzerinden plaka lokasyonu belirleme işlemi yapılmıştır. İkinci aşamada iki teknik kullanılmıştır. Birinci teknik lokasyonu bulunan plaka görüntüsüne görüntü işleme teknikleri uygulanarak karakter okuma işlemi yapılmıştır. Diğer teknikte ise OCR (Optical Character Recognition) ile karakter okuma işlemi yapılmıştır. Ve kullanılan iki tekniğin sonuçları karşılaştırılmıştır.

#### 5.2.1. R-CNN ile Plaka Konumunun Belirlemesi

```
load('D:\DERS\TEZ ÇALIŞMASI\plaka okuma birlestirilmis
22042019\plaka_bulma.mat');
```

Eğitilmiş ağın özellikleri plaka_bulma.mat matlab dosyası olarak kaydedilmiştir. Görüntü bu dosyada yüklü olan ağ özelliklerine göre evrişim işlemi yapılmaktadır. Bu komutla ağ parametreleri matlab çalışma alanına yüklenir. Ağın mimarisi, ağın katman sayısı, katmanların özellikleri, eğitilmiş ağ ağırlıkları burada yüklüdür. Bu parametrelere göre evrişim işlemi yapılır ve benzer ağırlıkların olduğu bölgeler bulunur.

```
load('D:\DERS\TEZ ÇALIŞMASI\plaka okuma birlestirilmis
22042019\Training\plaka_okuma.mat');
```

Birinci aşamada R-CNN ile bulunan plaka konumu, ikinci aşamada görüntü işleme teknikleriyle okunmaktadır. Bu kod ikinci aşama da kullanılacak eşleştirmelerde plaka_okuma.mat matlab dosyasındaki çeşitli plaka rakam ve harflerin matlab çalışma alanına yüklenmesini sağlamaktadır.

```
img = imread('D:\DERS\TEZ ÇALIŞMASI\plaka okuma test\0.jpg');
```

Bu kod ile ‘plaka okuma test’ dosyasında bulanan ‘0’ isimli, jpg uzantılı, renkli test görüntüsünün ‘img’ olarak matlab çalışma alanına yükleme işlemi yapılmaktadır.

```
[bbox, score, label] = detect(rcnn, img, 'MiniBatchSize', 32);
```

‘plaka_bulma’ matlab dosyası içindeki parametreleri kullanarak R-CNN ile ‘img’ görüntüsü içerisinde eğitilmiş ağırlıklara yakın ağırlıkların bulunduğu bölgeleri bulmaktadır. Bölgeler loss değeri en düşük, score fonksiyonu en yüksek olan bölgelerdir ve birden fazla olabilir. Bulunan bölgeler ‘bbox’ klasörü içerisine double değişkeni olarak atılır.

bbox

7x4 double

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1527 | 2302 | 51 | 25 |
| 2 | 1601 | 2074 | 1102 | 290 |
| 3 | 781 | 444 | 167 | 304 |
| 4 | 1229 | 1421 | 1009 | 363 |
| 5 | 908 | 499 | 274 | 182 |
| 6 | 1562 | 1872 | 335 | 460 |
| 7 | 2051 | 1596 | 547 | 821 |

**Şekil 5.2.** Bulunan Bölgelerin Görüntü Üzerindeki Konumları

score

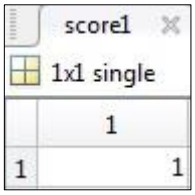
7x1 single

| | 1 |
|---|---|
| 1 | 0.9985 |
| 2 | 1 |
| 3 | 0.7159 |
| 4 | 0.9107 |
| 5 | 0.9998 |
| 6 | 0.9974 |
| 7 | 0.9988 |

**Şekil 5.3.** Bulunan Bölgelerin Skor Fonksiyonları

```
[score1, idx] = max(score);
```

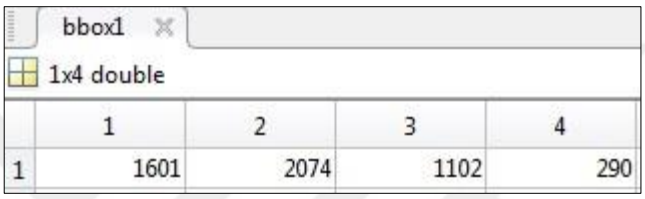
Bu kod ile maksimum score fonksiyonunu ‘score1’ değişkeni içerisine ve maksimum score fonksiyonunun satır değerini ‘idx’ değişkeni içerisine atama işlemi yapılmıştır.




**Şekil 5.4.** En Yüksek Skor Fonksiyonu

```
bbox1 = bbox(idx, :);
```

'bbox' klasörü içerisindeki maksimum score fonksiyonu en yüksek olan bölgeyi 'bbox1' klasörüne atama işlemi yapılmıştır.




**Şekil 5.5.** En Yüksek Skor Fonksiyonlu Plaka Konumu

```
aciklama = sprintf('(Doğruluk = %f)',score1)
```

'score1' fonksiyonunun değeri ondalık sayı olarak test görüntüsü olan 'img' görüntüsü üzerindeki en yüksek olasılıklı bölge etiketlenmektedir ve etiketleme işlemi yapıldıktan sonra 'aciklama' olarak çalışma alanına kayıt işlemi yapılmaktadır.


**Şekil 5.6.** Konumu İşaretlenmiş Plaka Görseli

```
bulunanbolge = insertObjectAnnotation(img, 'rectangle',bbox1,
aciklama,'LineWidth',8,'color','yellow','FontSize',18);
```

Bu kod ile en yüksek score fonksiyonuna sahip bölgenin ‘bbox’ değerlerine göre dikdörtgen içine alınmaktadır. Bir önceki kodda ‘aciklama’ değişkenine etiketlenen doğruluk oranı eklenmiş ve dikdörtgenin çizgi rengi, kalınlığı, yazı puntosu da girilmiştir.


**Şekil 5.7.** Plakanın Görüntü Üzerindeki Konumu

### 5.2.2. Görüntü İşleme Teknikleri İle Karakter Tanıma

```
picturex=imcrop(img,[bbox1(1),bbox1(2),bbox1(3),bbox1(4)]);
```

Birinci aşamada ‘img’ görüntüsünden R-CNN ile bulunan en yüksek score fonksiyonlu bölgenin kesilerek ‘picturex’ isimli görüntü olarak kaydedilmiştir. Burada ‘img’ görüntüsünden kesilen görüntü ‘bbox’ satır ve sütun değerlerinin lokasyonuna göre kesilmiştir.

**Şekil 5.8.** En Yüksek Skor Fonksiyonlu Kesilmiş Plaka Görüntüsü

```
[~,cc]=size(picturex);
```

Kesilen ‘picturex’ isimli rgb görüntüsünün matris olarak sütun değerleri toplamı ‘cc’ olarak kaydedilmiştir.

```
picture=imresize(picturex,[240 360]);
```

‘picturex’ isimli rgb görüntüsü yeniden boyutlandırma işlemi yapılarak 240x360 piksel olarak ‘picture’ ismiyle kaydedilmiştir.


**Şekil 5.9.** 240x360 Olarak Yeniden Boyutlandırılan Plaka Görüntüsü

```
if size(picture,3)==3
picture=rgb2gray(picture);
end
```

Burada ‘if’ döngüsüyle ‘picture’ isimli görüntü eğer renkli görüntü ise gri tonlu görüntü olarak değiştir ve ‘picture’ olarak tekrar kaydetme işlemi yapılmaktadır.


**Şekil 5.10.** Gri Tonlu Görüntüye Çevrilen Plaka Görüntüsü

```
threshold = graythresh(picture);
```

‘graythresh’ komutu ile gri tonlu ‘picture’ isimli görüntüye otomatik olarak threshold değeri atanmaktadır. Bu komut görüntünün parlaklık eşiğini otomatik olarak belirler ve sonuç olarak 0-1 arasında bir değer oluşturur.

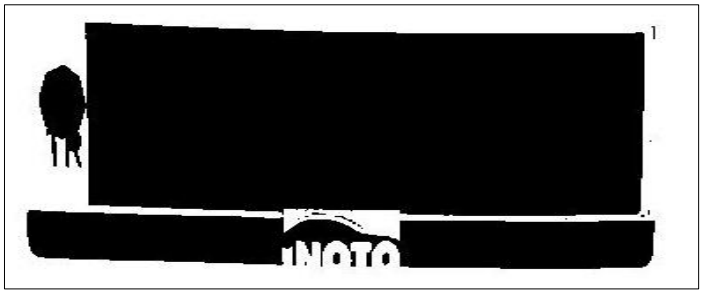
```
picture =~im2bw(picture, threshold);
```

Otomatik olarak belirlenen threshold eşik değerine göre ‘picture’ isimli gri tonlu görüntü siyah-beyaz görüntüye dönüştürülmekte ve ~(tilde) işaretiyle görüntünün tümleyeni alınmaktadır.


**Şekil 5.11.** Siyah-Beyaz Görüntüye Çevrilmiş Plaka Görüntüsü

```
if cc>2000
    picture1=bwareaopen(picture,5500);%
else
picture1=bwareaopen(picture,3978);
end
figure, imshow(picture1)
```

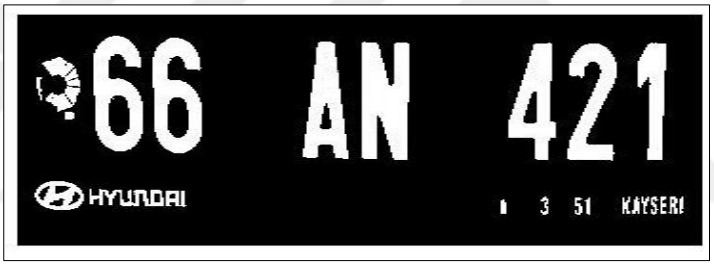
Kesilmiş görüntünün matris sütun değeri ‘cc’ eğer 2000’den büyükse ‘picture1’ isimli görüntüdeki karakterlerin sütun sayı değeri 5500 pikselden büyük olan parçaları sil, 5500 pikselden büyük karakter yoksa 3978 pikselden büyük olan parçaları sil denilerek görüntüden karakterler temizlenmektedir. Bu işlemler yapıldıktan sonra görüntü tekrar ‘picture1’ ismiyle kaydedilmektedir.



**Şekil 5.12.** Karakterlerden Arındırılmış Plaka Görüntüsü

```
picture2=picture-picture1;
```

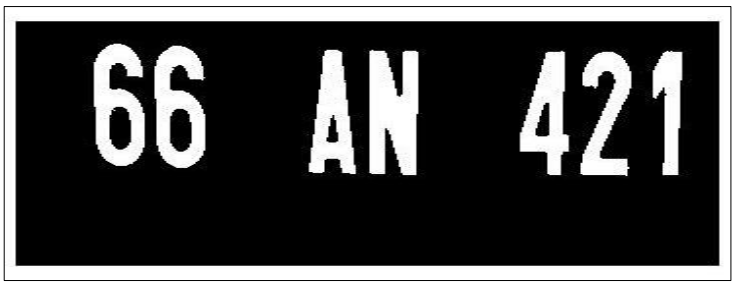
Siyah beyaz plaka görüntüsünden (‘picture’) karakterlerden temizlenen görüntünün (‘picture1) çıkarılmasıyla karakterlerin olduğu görüntü elde edilmektedir. Burada görüntülerin satır ve sütun sayıları eşit olduğundan çıkarma işlemi matrissel olmaktadır.




**Şekil 5.13.** Karakterlerin ve Gürültülerin Olduğu Plaka Görüntüsü

```
picture2=bwareaopen(picture2,550);
```

Bu komutta elde edilen karakter görüntüsündeki (‘picture2’) 550 pikselden fazla olan gürültülerin silinme işlemi yapılmıştır.



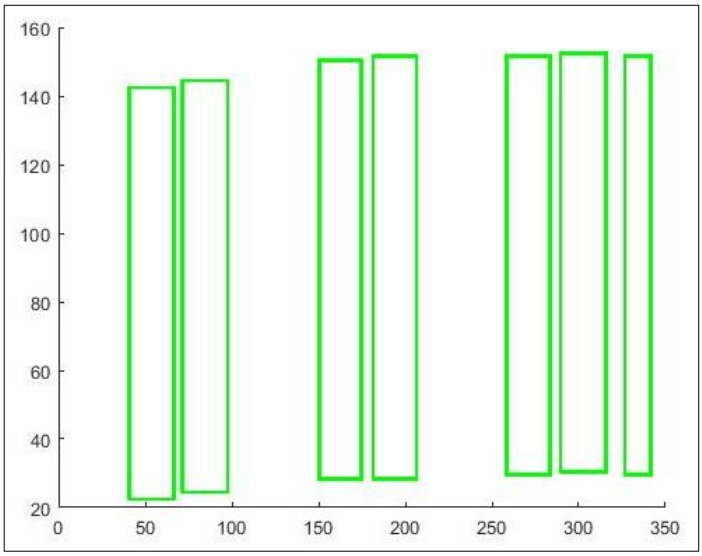

**Şekil 5.14.** Sadece Karakterlerin Olduğu Plaka Görüntüsü

```
[L,Etk]=bwlabel(picture2);
propied=regionprops(L,'BoundingBox');
holdon
```

Burada gürültülerden arındırılmış sadece karakterlerin olduğu 'picture2' görüntüsünün karakterleri etiketlenmiştir ve sayısı belirlenmiştir. For döngüsü için etiketli resme 'L' denilmiş ve karakterler 'Etk' ile ifade edilmiştir. Ve 'regionprops' komutuyla etiketli resimlerin konumların işaretlenmiş 'propied' değişkeni içerisine atanmıştır.

```
for n=1:size(propied,1)
rectangle('Position',propied(n).BoundingBox,'EdgeColor','g',
'LineWidth',2)
end
```

Burada bir önceki komutta konumları belirlenmiş karakterler for döngüsüne sokularak tek tek işaretlenme işlemi yapılmaktadır. 'hold' komutuyla bütün karakterler tek bir figürde işaretlenmiştir.




**Şekil 5.15.** Plaka Üzerindeki Karakterlerin Konumları


**Şekil 5.16.** Belirlenmiş Plaka Karakterleri

```
figure
plaka_cikis=[];
t=[];
```

Bu kod ile daha sonra açılacak text dosyası için boş bir karakter dizisi açma işlemi yapılmaktadır. Ve ismi 'plaka_cikis' olarak kayıt işlemi yapılmıştır. Ayrıca plakadaki bulunan karakterler ile daha önce eğitilen plaka karakter görselleri ile yapılan eşleştirmede her bir karakterin bulanan maksimum skorların yerleştirilmesi için 't' isminde boş bir hücre dizisi açılmıştır.

```
for n=1:Etk
[r,c] = find(L==n);
n1=picture(min(r):max(r),min(c):max(c));
n1=imresize(n1,[42,24]);
imshow(n1)
pause(0.1)
x=[ ];
```

Bu for döngüsünde plaka görüntüsü (L) içerisindeki etiketli karakterleri (Etk) sırasıyla eşleştirmede kullanılan karakterlerin boyutlarıyla aynı olacak şekilde 42x24 piksel boyutuna getirilmiştir. Ve 'x' ismiyle bir boş bir dizin açma işlemi yapılmıştır.

```
toplam_kar=size(plaka_okuma,2);
```

Plaka görüntüleri eşleşmesi için oluşturulmuş 2x46'lık 'plaka_okuma' hücre dizinin sütün sayısını 'toplam_kar' ismiyle kayıt işlemi yapılmıştır.

```
for k=1:toplam_kar
y=corr2(plaka_okuma{1,k},n1);
x=[x y];
end
```

Bir önceki for döngüsü bitmeden bir döngü daha açılmıştır. Plaka görselinde bulunan her karakter sırasıyla 'plaka_okuma' içerisindeki yüklü olan bütün karakterler ile tek tek korelasyon hesabı yapılmaktadır. Her karakter için hesaplanan korelasyon değeri 'x' dizini içerisine kayıt işlemi yapılmıştır.

Korelasyon hesap formülü;

$\bar{x} =$ $x$ değişkenlerinin ortalaması

$\bar{y} =$ $y$ değişkenlerinin ortalaması

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \quad (27)$$

Korelasyon katsayısı, değişkenler arasındaki ilişkiyi göstermek için kullanılan bir değerdir. Değişken, aşağıdaki formülle hesaplandığında -1 ile 1 arasında bir değer alır. Negatif değerler negatif ilişkiyi, pozitif değerler ise pozitif ilişki olduğunu gösterir. Değerler 1 veya -1 olduğunda mükemmel bir ilişki vardır. Değerler 0'a yaklaştıklarında değişkenlerin aralarındaki ilişki de azalır. 0 olduğunda ilişki yoktur.

```
t=[t max(x)];
if max(x)>.15
z=find(x==max(x));
```

'x' dizini içerisine her karakter için kaydedilen korelasyon değeri bulunmaktadır. +1'e yakın olan en yüksek değer pozitif bir korelasyon olduğunu göstermektedir. 't' dizini içerisine plaka karakterlerinin maksimum korelasyonu kaydedilmiştir. Eğer 'x'

dizini içerisine kaydedilen korelasyon 0.15'ten büyükse maksimum korelasyonun indeksini 'z' değişkenine kayıt işlemi yapılmıştır.

```
cikis=cell2mat(plaka_okuma(2,z));
plaka_cikis=[plaka_cikis cikis];
```

Burada 'plaka_okuma' dosyası içerisindeki 2. satır z. sütundaki dizini karaktere çevirdikten sonra 'çıkış' değişkenine kayıt edilme işlemi yapılmıştır.

```
file = fopen('Plaka_Sonucu.txt', 'wt');
fprintf(file,'%s\n',plaka_cikis);
fclose(file);
winopen('Plaka_Sonucu.txt')
```

En son işlem olarak 'plaka_cıkıs' karakter değişkenini bir text açarak kayıt işlemi yapılmakta ve plaka okuma işlemini sonuçlanmaktadır.

### 5.2.3. OCR (Optical Character Recognition) İle Karakter Tanıma

```
ocrsonuc = ocr(picture2);
```

Çalışmanın ilk aşamasında araç plaka lokasyonu belirleme işlemi ve ikinci aşamada bulunan plaka lokasyonuna korelasyon işlemine tabi tutarak karakter tanıma işlemi yapılmıştı. Burada ilk aşamada bulunan plaka lokasyonuna korelasyon işleminden farklı olarak karşılaştırma yapmak için OCR ile karakter tanıma işlemi de yapılmıştır. OCR, farklı dillerden oluşan 8 yazı tipi, 94 farklı karakter çeşidi ve 60 binden fazla karakterle eğitilmiş evrişim sinir ağı modelidir.

# SONUÇ

Bu tez çalışmasında makine öğrenme yöntemlerinden derin öğrenme teknikleri kullanılarak araç plaka tanıma sistemi tasarlanmıştır. Plaka tanıma sistemi iki aşamadan oluşmaktadır. İlk aşamada araç plaka lokasyonu bulunması, ikinci aşamada ise plakanın okunması hedeflenmiştir.

Plaka lokasyonunun bulunmasında yapay sinir ağları kullanılmıştır. Kullanılan yapay sinir ağı modeli son yıllarda popüler olan ve görüntü işlemede de sık kullanılan derin öğrenme tekniği evrişim sinir ağları (CNN) kullanılmıştır. Sinir ağının eğitimi için 450 adet araç görseli kullanılmıştır. Eğitilmiş sinir ağını test etmek için 100 adet araç görseli kullanılmıştır. Sinir ağının eğitiminde ve test işlemlerinde ağ içinde parametre sayısı çok fazla olacağından evrişimli sinir ağı modeli bölgesel-evrişimli sinir ağı (R-CNN) tercih edilmiştir. R-CNN'in çalışma prensibi görüntüyü tek seferde işleme tutmaktan ziyade parçalar halinde işleme tabi tutmaktadır. Bu parçalar görüntü üzerinden seçici arama algoritması ile seçilen ortalama 2000 adet bölgeden oluşmaktadır. Evrişimli sinir ağı modelimiz (giriş, normalizasyon ve aktivasyon katmanları dahil) 15 gizli katmandan oluşmaktadır, öğrenme oranımız $10^{-6}$ ve epoch (ağ eğitimi tekrar sayısı) 10 olarak girilmiştir.

İkinci aşamada 2 teknik kullanılmış ve sonuçlar karşılaştırılmıştır. İkinci aşamada kullanılan ilk teknik klasik görüntü işleme teknikleriyle korelasyon işlemiyle karakter tanıma yapılmıştır. Bulunan plaka karakterlerinin eşleştirilmesi (korelasyon) için harf ve rakamlardan oluşan 48 adet karakter toplanmıştır. Bulunan plaka görüntüsü önce siyah-beyaz görüntüye çevrilmiş, sonra görüntü üzerindeki gürültüler yok edilmiştir. Daha sonra plaka üzerinde işaretlenen karakterler yeniden boyutlandırılarak test karakterleri ile bir korelasyon işlemine tabi tutulmuştur. En yüksek korelasyon değeri olan karakter doğruluğu kesin olan karakter olarak belirlenmiştir. İkinci aşamada kullanılan diğer teknik ise OCR (Optical Character Recognition)'dir. OCR, evrişimli sinir ağlarının (CNN) kullanıldığı eğitilmiş modeldir.

Bu çalışmada, derin öğrenme için kullanılan yöntemler araştırılarak araç plaka tanıma uygulaması yapılması amaçlanmıştır. Bu çalışmanın amacı, hem lisansüstü çalışmalarda derin öğrenme konusunda bir altyapı oluşturmak, hem de derin öğrenme

ile çalışılacak araç plaka lokasyonunun belirlenmesi ve tanınması üzerine bir uygulama geliştirmektir. Elde edilecek plaka tanıma sistemi ile insan hayatını kolaylaştıracak farklı sektörlerdeki bilgi birikimini artırmanın yanı sıra akıllı otopark sistemlerine de bir katkı sağlamak tezimizin en önemli amacıdır.

Çalışmanın sonucunda test görüntülerinden elde edilen doğruluk oranı % 94 olarak belirlenmiştir. Donanım hızının ve eğitim veri setinin artması durumunda daha hızlı derin öğrenme metotları kullanarak doğruluk oranında artış, test süresinde azalış görülebilir.

Araçlardaki plakaların tespiti için kameralardan gelen verilerle başa çıkabilmek için ve plakanın lokasyonunun belirlenip plakanın tanınması gibi benzer birçok alanda derin öğrenme gelecekte de kullanılacak bir yol olarak görülmektedir. Görüntülerden oluşan büyük verilerin en önemli problemlerinden bir tanesi de etiketleme maliyetidir. Bunun üstesinden gelmek için yine gelecekte denetimsiz derin öğrenme yöntemleri ön plana çıkacaktır. Derin öğrenme üzerinde araştırma yapan ekiplerin neredeyse hepsi doğal dil işleme alanında geliştirmeler yapmaktadır. Ancak çoğu çalışmanın uluslararası alanda yapıldığı gözlemlenmektedir [11-12]. Özellikle Ülkemiz için bu alanda yapılan çalışmalar büyük önem arz etmektedir. Görüntü işleme konusunda görüntüden nesne çıkarımı, sınıflandırma, tanıma gibi problemlerin çözümünde derin öğrenme algoritmalarının (özellikle RNN, LSTM, CNN gibi) başarı bir şekilde kullanıldığı görülmüştür. Bu sebeple, bu alana yönelmek, Ülkemizdeki bilgi birikimine katkı sağlamak için benzer çalışmalar değerli olacaktır.

Sonuç olarak yapılan bu çalışma sayesinde yerli ve milli olarak hazırlanabilen yazılımlar sayesinde plaka okuma konusunda derin öğrenme algoritmalarının kullanımı ve yaygınlaşması adına önemli bir çalışma olduğu kanaatinde olup benzer çalışmalara zemin hazırlayacağı ve tez içeriğinde bulunan teorik bilgilerin makine öğrenmesine başlayan öğrencilere ışık tutacağı düşünülmektedir.

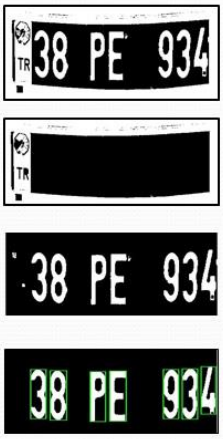
## KORELASYON İLE ELDE EDİLEN BAŞARILI TEST SONUÇLARI



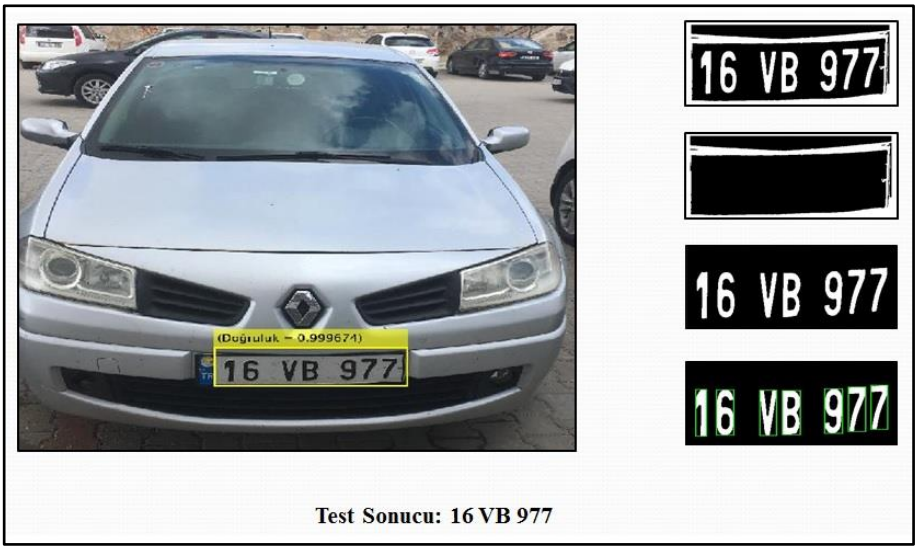
**Test Sonucu: 66 DU 689**



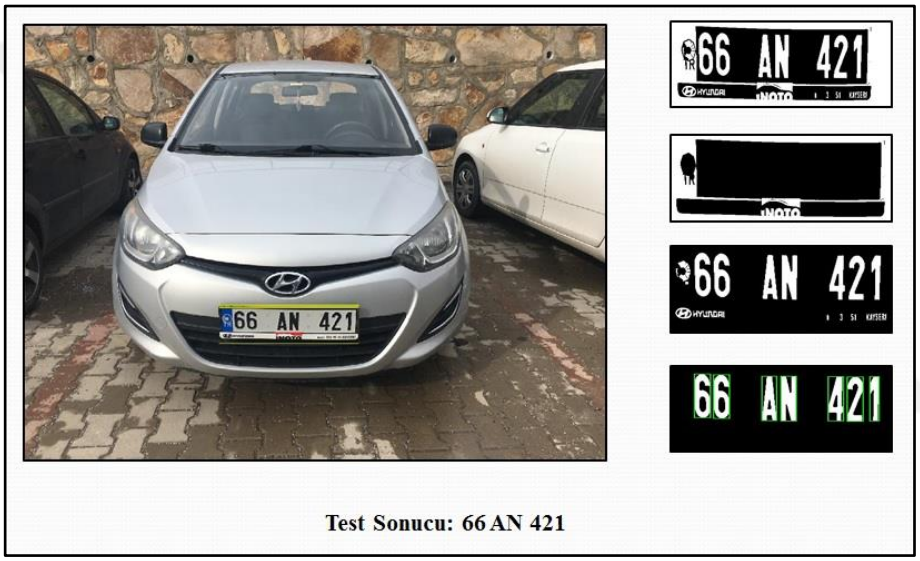
**Test Sonucu: 66 LB 021**

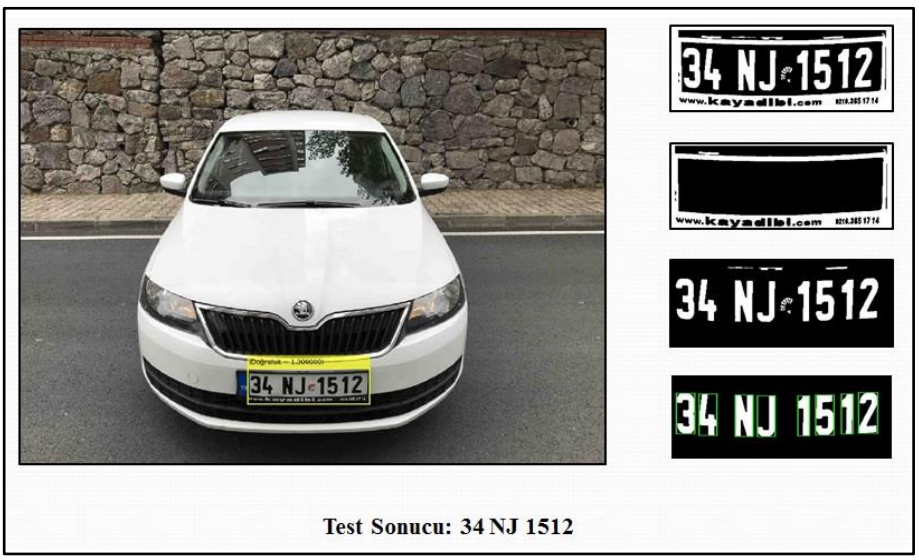

**Test Sonucu: 38 PE 934**



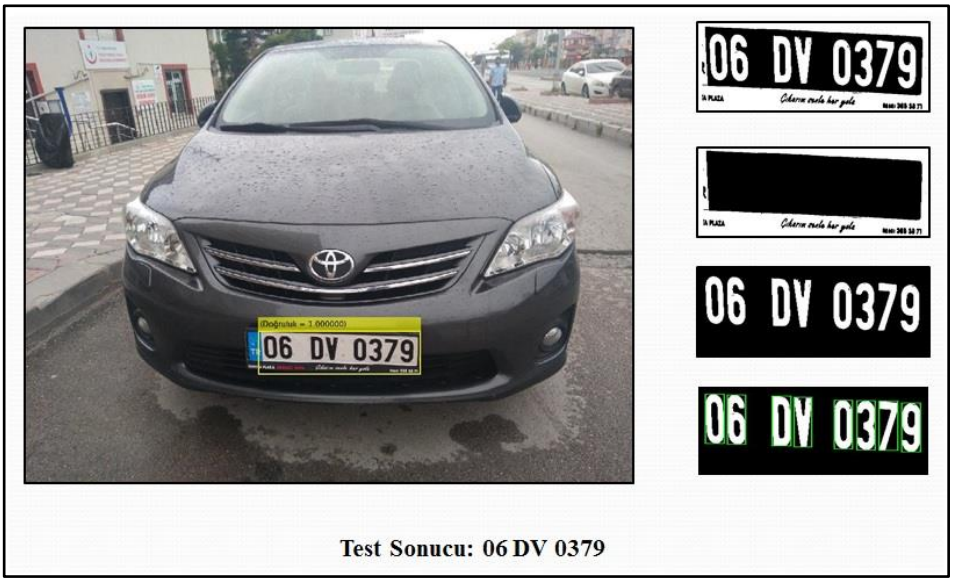

**Test Sonucu: 16 VB 977**




**Test Sonucu: 66 AN 421**




**Test Sonucu: 34 NJ 1512**




## OCR İLE ELDE EDİLEN BAŞARILI TEST SONUÇLARI


| | ocrsonuc.Words |
|---|---|
| | 1 |
| 1 | 50 |
| 2 | BH |
| 3 | 406 |


| | ocrsonuc.Words |
|---|---|
| | 1 |
| 1 | 01 |
| 2 | V |
| 3 | 0365 |

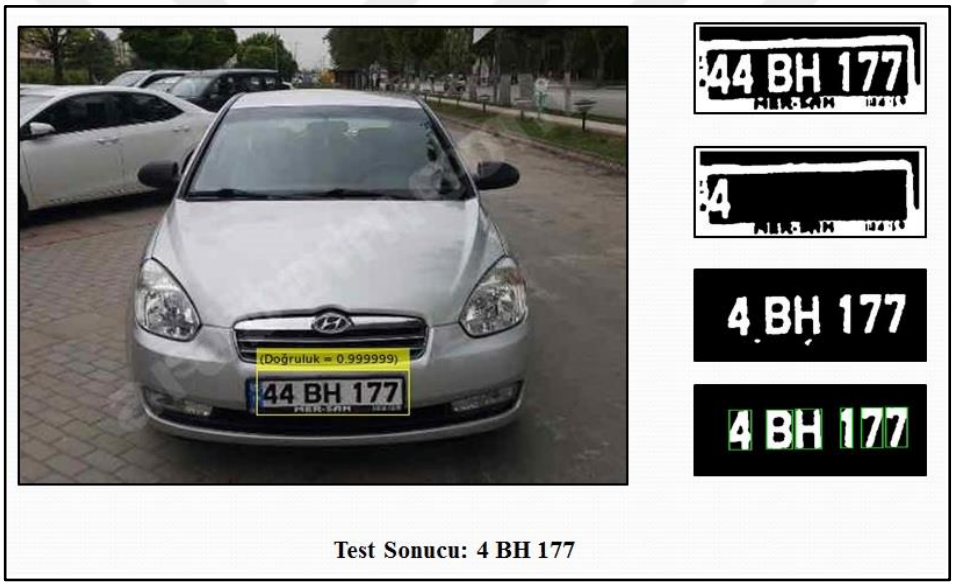
| | ocrsonuc.Words |
|---|---|
| | 1 |
| 1 | 66 |
| 2 | KK |
| 3 | 677 |


| | ocrsonuc.Words |
|---|---|
| | 1 |
| 1 | 15 |
| 2 | ABK |
| 3 | 734 |

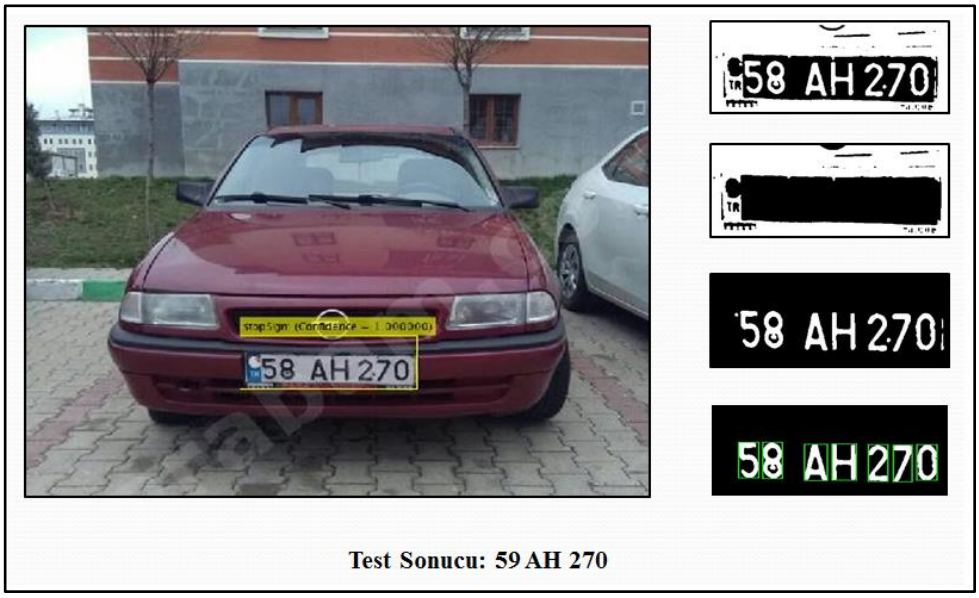

| | ocrsonuc.Words |
|---|---|
| | 1 |
| 1 | 06 |
| 2 | S |
| 3 | 0581 |

## HATALI TEST SONUÇLARI


Test Sonucu: 06 E2 2885




Test Sonucu: 4 BH 177




**Test Sonucu: 59 AH 270**

## TEST SONUÇLARI

| | | | |
|---|---|---|---|
| Ağ Eğitiminde Kullanılan Görüntü Sayısı | | | 450 |
| Test Edilen Toplam Görüntü Sayısı | | | 100 |
| | | | |
| R-CNN İle Plaka Konumu Belirleme Sonuçları | | | |
| Doğru | 95 | Hatalı | 5 |
| | | | |
| Korelasyon İşlemi İle Plaka Okuma Sonuçları | | | |
| Doğru | 94 | Hatalı | 6 |
| OCR İle Plaka Okuma Sonuçları | | | |
| Doğru | 92 | Hatalı | 8 |

| | | | |
|---|---|---|---|
| Toplam Test Sonucu Doğruluk Oranı | | | |
| Doğruluk | % 94 | Hata | % 6 |

# KAYNAKLAR


1. Googfellow, Y. Bengio, A. Courville, “Deep Learning” London, (2015), 16-20

2. E. ÇAMAŞIROĞLU, “Araç Plaka Algılaması ve Tanıma”, (Yüksek Lisans Tezi, Ankara Üniversitesi, 2007)

3. İ. KAŞIKÇI, “Otomatik Plaka Tanıma Sistemi”, (Yüksek Lisans Tezi, Ege Üniversitesi, 2008)

4. N. S. S. Angara, “Automatic License Plate Recognition Using Deep Learning Techniques”, (Yüksek Lisans Tezi, University Of Texas At Tyler, 2015)

5. H. Jorgensen, “Automatic Lisence Plate Recognition Using Deep Learning Techniques”, (Yüksek Lisans Tezi, Norwegian University Of Science And Technology, 2017)

6. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning Applied to document recognition,” Proc. IEEE, Nov. 1998.

7. A.Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep Convolutional Neural Networks,” in Advances in Neural Information Processing Systems *25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.),pp. 1097–1105, Curran Associates, Inc., 2012.

8. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Computer Vision and Pattern Recognition (CVPR), 2015.

9. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Computer Vision and Pattern Recognition, 2014.

10. Ali BAKKALOĞLU, “Araç Plaka Tanıma Sistemi”, (Yüksek Lisans Tezi, Selçuk Üniversitesi, 2011)

11. A.Krizhevsky, I. Sutskever, and G. E. Hinton, \Imagenet classification with deep convolutional neural networks," in Advances Neural Inform. Proc. Syst., 2012, pp. 1097-1105.

12. Y. Bengio, \Learning deep architectures for ai," Found. Trends Mach. Learning, vol. 2, no. 1, pp. 1{127, Jan. 2009. [Online].

13. Lee, H., R. Grosse, R. Ranganath and A. Y. Ng (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Proceedings of the 26th annual international conference on machine learning, ACM

14. K. Simonyan And A. Zisserman. “Very Deep Convolutional Networks For Large-Scale İmage Recognition” In Iclr, 2015

15. “Convolutional Neural Networks for Visual Recognition” (March 2019), http://cs231n.stanford.edu/

16. Özcan, H., “Çok Düşük Çözünürlüklü Yüz İmgelerinde Derin Öğrenme Uygulamaları”, (Yüksek Lisans Tezi, Deniz Harp Okulu, Bilgisayar Müh. Bölümü), 2014

17. “Deep Learning A-Z™ Python ile Derin Öğrenme”, (Nisan 2019), https://www.udemy.com/derin-ogrenmeye-giris/

18. “Derin Öğrenme Yöntemleri ile Geri Dönüşüm Malzemelerinin Tanınması”, (2018), https://medium.com/deep-learning-turkiye

19. Emine Cengil, Ahmet Çınar. (2016) "A NEW APPROACH FOR IMAGE CLASSIFICATION: CONVOLUTIONAL NEURAL NETWORK", European Journal of Technic, Vol 6, Number 2, pp: 96-103.

20. LeCun Y., Touretzky D., Hinton G. and Sejnowski T.,(1988). A theoretical frame work for Back-Propagation”.in Proceedings of the 1988 Connectionist Models Summer School, Pittsburgh, 1988, pp. 21-28.

21. Hacıoğlu, C., Derinlikli Öğrenme Kullanılarak Konuşmadan Uykululuk/Uykusuzluk Tespiti, Master thesis, İstanbul Technical University, Istanbul, Turkey, 2014.

22. Zhou S., Chen Q., Whang X., (2013). Convolutional Deep Networks for Visual Data Classification. Neural Process Lett, 38:17-27.

23. A.Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks," in Advances Neural Inform. Proc. Syst., 2012, pp. 1097-1105.

24. Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision," in Circuits and Syst. (ISCAS), Proc. of 2010 IEEE Int. Symp., pp. 253-256.

25. D. Ciresan, U. Meier, and J. Schmidhuber, \Multi-column deep neural networks for image classification," in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 3642-3649.

26. Y. Bengio, \Learning deep architectures for ai," Found. Trends Mach. Learning, vol. 2, no. 1, pp. 1{127, Jan. 2009. [Online].

27. Lee, H., R. Grosse, R. Ranganath and A. Y. Ng (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. Proceedings of the 26th annual international conference on machine learning, ACM.

28. LeCun, Y., Y. Bengio and G. Hinton (2015). "Deep learning." Nature 521(7553): 436-444.

29. A. Berg, J. Deng, and L. Fei-Fei. Larges Calve Visual Recognition Challenge 2010. www.imagenet.org/challenges. 2010.

30. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks, Papers.Nips.cc 2012

31. G. Adorni, F. Bergenti, and S. Cagnoni, "Vehicle license plate recognition by means of cellular automata," in Proc. IEEE Int. Conf. Intelligent Vehicles, 1998, pp. 689–693.

32. M. H. T. Brugge, J. H. Stevens, J. A. G. Nijhuis, and L. Spaanenburg, "License plate recognition using DTCNNs," in Proc. 5th IEEE Int. Workshop on Cellular Neural Networks and Their Applications, 1998, pp. 212–217

33. K. R. Castleman, Digital Image Processing. Englewood Cliffs, NJ: Prentice-Hall, 1996, pp. 550–554

34. S. W. Chen, G. C. Stockman, and K. E. Chang, "SO dynamic deformation for building of 3-D models," IEEE Trans. Neural Networks, vol. 7, pp. 374–387, June 1996.

35. J. R. Cowell, "Syntactic pattern recognizer for vehicle identification numbers," Image and Vision Computer Vol. 13, no. 1, pp. 13–19, 1995

36. Y. Cui and Q. Huang, "Character extraction of license plates from video," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1997, pp. 502–507.

37. P. Davies, N. Emmott, and N. Ayland, "License plate recognition technology for toll violation enforcement," Inst. Elect. Eng. Colloquium Image Analysis for Transport Applications, pp. 7/1–7/5, 1990

38. R. Smith. "An Overview of the Tesseract OCR Engine". Google Inc. (2007), 629-633

39. Gerber C., Mokdong C. "Number Plate Detection with a Multi-Convolutional Neural Network Approach with Optical Character Recognition for Mobile Devices". Journal Of Information Systems, (2016), 100-108

40. M. M. Shaifur Rahman, "Bangla License Plate Recognition Using Convolutional Neural Networks (CNN)", (Yüksek Lisans Tezi, BRAC University, 2017)

41. Kai Tang, "Chinese License Plate Recognition", (Yüksek Lisans Tezi, State University of New York, 2018)

42. Lele Xie, Tasweer Ahmad, Lianwen Jin "A New CNN-Based Method for Multi-Directional Car License Plate Detection" IEEE. (2018)

# ÖZGEÇMİŞ

1985 yılında Yozgat-Çekerek'te doğan Talip ÇAY, lise öğrenimini Çekerek Endüstri Meslek Lisesi Elektrik Bölümünde tamamlamıştır. 2009 yılında Gazi Üniversitesi Teknik Eğitim Fakültesi Elektrik Öğretmenliği Bölümünü, 2016 yılında Yozgat Bozok Üniversitesi Elektrik-Elektronik Mühendisliği Bölümünü başarıyla bitirmiştir.

2016 yılında yüksek lisans eğitimine Yozgat Bozok Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Mühendisliği Anabilim Dalında başlamıştır. Halen Yozgat Bozok Üniversitesi Yapı İşleri ve Teknik Daire Başkanlığında Mühendis olarak çalışmakta olan Talip ÇAY, evli ve 2 çocuk babasıdır.

**İletişim Bilgileri**

Adres: Yozgat Bozok Üniversitesi Rektörlüğü Yapı İşleri ve Teknik Daire Başkanlığı Merkez/YOZGAT

Telefon: (354) 242 10 70

Faks: (354) 242 10 68

E-posta: talip.cay@gmail.com