T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Görsel Derin Öğrenme ile Döviz Kuru Hesaplama

EMRE JILTA
1158105153

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

TEZ DANIŞMANI: DR. ÖĞR. ÜYESİ CEM TAŞKIN

EDİRNE-2019

T.Ü. Fen Bilimleri Enstitüsü Onayı

Emre JILTA'nın hazırladığı **"Görsel Derin Öğrenme ile Döviz Kuru Hesaplama"** başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir **Yüksek Lisans** tezi olarak kabul edilmiştir.

Jüri Üyeleri: **İmza**

Dr. Öğr. Üyesi Cem TAŞKIN .....

Dr. Öğr. Üyesi Bora ASLAN .....

Dr. Öğr. Üyesi Tarık YERLİKAYA .....

Tez Savunma Tarihi: 10/07/2019

Bu tezin **Yüksek Lisans** olarak gerekli şartları sağladığını onaylarım.

Dr. Öğr. Üyesi Cem TAŞKIN
Tez Danışmanı

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof. Dr. Murat YURTCAN
Fen Bilimleri Enstitüsü Müdürü

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

10/07/2019

EMRE JILTA



YÜKSEK LİSANS TEZİ
Görsel Derin Öğrenme ile Döviz Kuru Hesaplama
T.Ü. Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Yıllar boyunca dış dünyanın bilgisayar tarafından insan beyni gibi verileri işlemesi için sayısallaştırılma işlemleri araştırılmıştır. Bu süreçte en çok zorlanılan problemler arasında görüntü işleme, nesne tanıma, ses tanıma sayılabilir. Uzun yıllar bu problemleri çözmede geliştirilen algoritmalar yeterli başarı oranına ulaşamamıştır. Bilgisayar görüsünün yeterli başarı oranına ulaşması yapay sinir ağlarıyla sağlanmıştır. Günümüzde yapay sinir ağlarının bir alt dalı olan derin öğrenme birçok zor problemin çözümünde kullanılmaktadır. Derin öğrenme, insan beyninden esinlenilerek geliştirilen bir makine öğrenmesi yöntemidir. Günümüzde ticari, finansal ve kişisel gereksinimlerden biri de döviz kurlarının hızlı bir şekilde birbirine dönüştürülmesidir. Bu tez çalışmasında görsel derin öğrenmenin bir modeli olan Evrişimsel Sinir Ağları ile mobil cihaz üzerindeki kameradan anlık olarak alınan görüntünün içerisinde yer alan Türk Lirası, Euro ve Amerikan Doları cinsinden banknotlar tespit edilmektedir. Tespit edilen banknotların günlük döviz kurları üzerinden diğer para cinslerine göre karşılığı hesaplanmaktadır. Geliştirilen sinir ağının eğitimi sırasında 11.400 adet etiketli fotoğraf kullanmıştır. Fotoğrafların tamamı manuel olarak etiketlenmiştir. Geliştirilen model 3 farklı para birimini yaklaşık olarak %80 oranında tespit edebilmektedir.

Yıl : 2019
Sayfa Sayısı : 61
Anahtar Kelimeler: Derin Öğrenme, Bilgisayar Görüsü, Yapay Sinir Ağları, Döviz Kuru, Nesne Tanıma

MASTERS THESIS

Exchange Rate Calculation with Visual Deep Learning

Trakya University Institute of Natural Sciences

Computer Engineering Department

ABSTRACT

Over the years, digitization processes have been researched for computing the external world as a human brain by computer. In this process, the most challenging problems are image processing, object detection and voice recognition. The algorithms have not achieved sufficient success rate which were developed to solve these problems for many years. Achievement of satisfying success rate of computer vision was provided by neural networks. Deep learning, which is a sub-branch of artificial neural networks, is used for many difficulties nowadays. Deep learning is a machine learning method inspired by the human brain. Nowadays, one of the increasing requirement is the rapid transformation of exchange rates into one another in commercial, financial and personal fields. In this thesis, Convolutional Neural Networks, which is a model of visual deep learning, is used to identify banknotes in Turkish Lira, Euro and American Dollars within the image taken by the camera on the mobile device. The equivalents of the determined banknotes are calculated according to the other currencies at daily exchange rates. 11.400 tagged photographs are used during the training of the developed neural network. All photographs were tagged manually. The developed model can detect 3 different currencies by approximately 80%.

Year : 2019

Number of Pages : 61

Keywords: Deep Learning, Computer Vision, Artificial Neural Networks, Exchange Rate, Object Recognition

TEŞEKKÜR

Yüksek lisans eğitimim süresince engin tecrübesiyle birlikte her konuda destek ve önerilerini esirgemeyen, anlayış ve sabırla karşılayan danışman hocam Dr. Öğr. Üyesi Cem TAŞKIN'a, tezimin araştırma ve yazım aşamasında akademik desteğini esirgemeyen değerli hocam Arş. Gör. Dr. Emir ÖZTÜRK'e, lisans ve yüksek lisans eğitimim süresince bilimsel ve mesleki deneyimlerini paylaşan tüm hocalarıma;

Hem lisans hem de yüksek lisans eğitimimde biz soydaşlarını hiçbir konuda yolda bırakmayan, her türlü desteği esirgemeksizin sağlayan Türkiye Cumhuriyeti ve Trakya Üniversitesine;

Son olarak da bugünlere gelmemde emeği geçen, hiçbir zaman desteğini esirgemeyen babam Dr. Driton JILTA, annem Esin JILTA ve yanımda olan kız kardeşim Ebru JILTA'ya sonsuz teşekkürlerimi sunuyorum.

KISALTMALAR

API : Application Programming Interface

DAE : Derin Oto-Kodlayıcılar

DBN : Derin İnanç Ağları

DVM : Destek Vektör Makinesi

ESA : Evrişimsel Sinir Ağları

GİB : Grafik İşlemci Birimi

ISO : International Organization for Standardization

LSTM : Uzun Kısa Vadeli Hafıza

RBM : Sınırlı Boltzmann Makineleri

RELU : Doğrultulmuş Doğrusal Birim

RNN : Tekrarlı Sinir Ağları

TCMB: Türkiye Cumhuriyeti Merkez Bankası

TDK : Türk Dil Kurumu

YSA : Yapay Sinir Ağları

YZ : Yapay Zekâ

ÇİZELGELER

Çizelge 3.1 Toplama Fonksiyonları.....9
Çizelge 3.2 Etkinleştirme Fonksiyonları.....10
Çizelge 3.3 Etkinleştirme Fonksiyonlarının Türevi.....18
Çizelge 3.4 Filtreler 22
Çizelge 3.5 Hata Matrisinin Hesaplanması.....25
Çizelge 4.1 Kullanılan Para Birimlerinin Listesi.....30
Çizelge 4.2 Para Birimlerinin Ön ve Arka Yüzleri.....32
Çizelge 4.3 Hata Matrisi 40
Çizelge 4.4 Kestirim Oranları.....41

ŞEKİLLER

Şekil 3.1 Sinir Hücresi 7
Şekil 3.2 Yapay Sinir Ağı 7
Şekil 3.3 Yapay Sinir Hücresi 8
Şekil 3.4 Perceptron 14
Şekil 3.5 Geliştirilmiş Perceptron 14
Şekil 3.6 Kohonen Kuralı 15
Şekil 3.7 Hopfield Ağı 16
Şekil 3.8 Geri Beslemeli Yapay Sinir Ağı 17
Şekil 3.9 Bir Evrişimsel Katmanın Grafiksel Gösterimi 20
Şekil 3.10 Bir Filtre Örneği 20
Şekil 3.11 Evrişim İşlemi 21
Şekil 3.12 Tekrarlı Sinir Ağı 23
Şekil 3.13 Derin İnanç Ağı 24
Şekil 4.1 CoreML Mimarisi 29
Şekil 4.2 CreateML İş Akışı 29
Şekil 4.3 Veri Kümesine Ait Örnekler 33
Şekil 4.4 CreateML Kütüphanesi Çalışma Parametreleri 34
Şekil 4.5 Geliştirilen Modelin Çalışma Şekli 35
Şekil 4.6 Geliştirilen Modelin Mimarisi 36
Şekil 4.7 Geliştirilen Uygulamanın Açılış Ekranı 37
Şekil 4.8 Fotoğraf Çekimi ile Banknot Tanıma 38
Şekil 4.9 Gerçek Zamanlı Banknot Tanıma 39

FONKSİYONLAR

(3.1) Net Girdi Değerinin Belirlenmesi 10
(3.2) Doğruluk.....26
(3.3) Kesinlik.....26
(3.4) Yakalama.....26
(3.5) Doğru Negatif Oranı.....26
(3.6) Yanlış Sınıflandırma Oranı.....26
(3.7) Yanlış Sınıflandırma Oranı.....27
(3.8) Yanlış Pozitif Oranı.....27
(3.9) F1 Skoru 27

İÇİNDEKİLER

ÖZET iv
ABSTRACT v
TEŞEKKÜR vi
KISALTMALAR vii
ÇİZELGELER viii
ŞEKİLLER ix
FONKSİYONLAR x
1. BÖLÜM 1
GİRİŞ 1
2. BÖLÜM 2
LİTERATÜR ARAŞTIRMASI 2
3. BÖLÜM 4
YAPAY ZEKÂ 4
3.1 Yapay Zekâ 4
3.2 Yapay Sinir Ağları 6
3.2.1 Yapay Sinir Ağlarının Özellikleri 11
3.2.2 Yapay Sinir Ağlarının Avantaj ve Dezavantajları 11
3.2.3 Yapay Sinir Ağlarında Öğrenme Türleri 12
3.2.3.1 Denetimli Öğrenme 12
3.2.3.2 Denetimsiz Öğrenme 12
3.2.3.3 Destekleyici Öğrenme 13
3.2.3.4 Karma Öğrenme 13
3.2.4 Yapay Sinir Ağlarında Öğrenme Kuralları 13
3.2.4.1 Perceptron 13
3.2.4.2 Hebb Kuralı 15
3.2.4.3 Delta Kuralı 15
3.2.4.4 Kohonen Kuralı 15
3.2.4.5 Hopfield Kuralı 16
3.2.5 Yapay Sinir Ağı Türleri 16
3.2.5.1 İleri Beslemeli Yapay Sinir Ağları 16
3.2.5.2 Geri Beslemeli Yapay Sinir Ağları 17
3.3 Derin Öğrenme 18

3.3.1 Derin Öğrenme Modelleri 19
3.3.1.1 Evrişimsel Sinir Ağları 19
3.3.1.2 Tekrarlı Sinir Ağları 22
3.3.1.3 Uzun Kısa Vadeli Hafıza Ağları 23
3.3.1.4 Sınırlı Boltzmann Makineleri 23
3.3.1.5 Derin İnanç Ağları 24
3.3.1.6 Derin Oto-Kodlayıcılar 24
3.3.2 Öğrenme Transferi 24
3.4 Hata Matrisi 25
4. BÖLÜM 28
GELİŞTİRİLEN YÖNTEM 28
4.1 CoreML ve CreateML Kütüphaneleri 28
4.2 İktisadî Kavramlar 30
4.3 Döviz Kurunun Hesaplanması 30
4.4 Para Birimleri ve Veri Kümesi 31
4.5 Modelin Uygulamada Kullanımı 34
4.6 Performans Ölçütleri 39
5. BÖLÜM 42
SONUÇ 42
KAYNAKLAR 43
EK-1 46
SÖZLÜK 46
EK-2 48
image.py 48
ÖZGEÇMİŞ 49
TEZ ÖĞRENCİSİNE AİT AKADEMİK MAKALELER 50

1. BÖLÜM

GİRİŞ

Yapay Zekâ (YZ)'nın temelini oluşturan Yapay Sinir Ağları (YSA) bilişim ve matematik alanının dışında disiplinler arası araştırmalarda da kullanılmaktadır. Günümüzde YSA görüntü işleme alanında da etkin olarak kullanılmaktadır. YSA'nın bu alanda kullanımı insan-bilgisayar etkileşimini büyük ölçüde arttırmıştır. Bu bağlamda YSA kullanılarak, sınıflandırma ve kümeleme işlemleri yapan başarılı bilimsel ve ticari örnekler bulunmaktadır.

YSA, insan beyninde bulunan biyolojik sinir hücrelerini taklit etmek üzere geliştirilen matematiksel modellerdir. Görme, duyma, anlama gibi bilişsel eylemler YSA aracılığıyla sanal ortamda modellenebilmektedir. Bu modellerden biri bilgisayar görüsü olup, insan görüsünü taklit etmektedir. Farklı kullanım amaçlarına yönelik geliştirilen birçok Derin Öğrenme modeli bulunmaktadır. Bu tez çalışmasında görüntü sınıflandırma alanında başarılı olan Evrişimsel Sinir Ağları (ESA) kullanılmıştır.

Birinci bölümde tez konusu ve amacı açıklanmıştır. İkinci bölümde literatür araştırması gerçekleştirilmiştir. Üçüncü bölümde YZ'nın kronolojik gelişimi, YSA'nın modellenmesi, öğrenme kuralları, derin öğrenmenin yapısı ve modelleri yer almaktadır. Dördüncü bölümde geliştirilen yöntemle ilgili kavramlar, veri kümesi ve model açıklanmıştır. Beşinci bölümde bulgular ve sonuçlar yer almaktadır.

2. BÖLÜM

LİTERATÜR ARAŞTIRMASI

Bu tez çalışmasında görsel derin öğrenmenin bir modeli olan ESA temel alınarak CreateML çatısı ile bir derin öğrenme modeli oluşturulmuştur. Elde edilen modelin eğitimi için manuel olarak etiketlenen 11.400 adet banknot fotoğrafı kullanılmıştır. Söz konusu model geliştirilen iOS uygulamasına eklenerek mobil cihaz üzerindeki kameradan alınan görüntüdeki Türk Lirası, Euro ve Amerikan Doları tespit edilmektedir. Tespit edilen banknotların Türkiye Cumhuriyeti Merkez Bankası (TCMB)'ndan alınan günlük döviz kurları üzerinden diğer para cinslerine göre karşılığı hesaplanmaktadır. Geliştirilen model 3 farklı para birimini yaklaşık olarak %80 oranında tespit edebilmektedir.

Panah ve Masoumi tarafından 2017 yılında yayımlanan makalede İran banknotlarının görüntü işleme teknikleriyle değerlerinin bulunması sağlanmıştır. Bu çalışma iki aşamadan olup, ilk aşamada banknotların renkli görüntüleri alınmış ve ikinci aşamada alınan görüntüler gri tona dönüştürülerek banknotlardaki değerlerin çıkarımı sağlanmıştır. (Panah & Masoumi, 2017)

Kitagawa ve arkadaşları tarafından 2017 yılında yayımlanan makalede banknotların niteliklerinden yararlanılarak nominal değerlere göre sınıflandırma gerçekleştirilmiştir. Sınıflandırılacak banknotun yüzey büyüklüğü ve yönünün tespiti için ESA kullanılmıştır. Aday bölgelerin oluşturulması ve ESA tarafından her bölgenin bir portre içermesi olasılığını hesaplamak için hareketli pencereler kullanılmıştır. Daha sonra dikdörtgenlerin çıktısını almak için yüksek olasılığa sahip portreler ile Maksimum Olmayan Tutma tekniği kullanılmıştır. (Kitagawa vd, 2017)

Pham, Lee ve Park tarafından 2017 yılında yayımlanan makalede tek boyutlu bir çizgi sensörü tarafından görünür ışığa dayalı banknot resimleri çekilmiş ve her nominal

değerin boyut bilgisinin dikkate alınmasıyla ESA kullanılarak çok uluslu banknot sınıflandırma yöntemi geliştirilmiştir. (Pham, Lee, & Park, 2017)

Omatu, Yoshioka ve Kosaka tarafından 2007 yılında yayımlanan makalede değişkenler arasındaki doğrusal olmayan bağımlılıkları ortadan kaldırmak ve verilerin temel esas özelliklerini çıkarmak için yerel bir Temel Bileşen Analizi (PCA – Principal Component Analysis) yöntemi uygulanmıştır. Bu yöntemde veri alanı, bir Öz Örgütlemeli Eşleme (SOM – Self-Organizing Map) modeli kullanılarak bölgelere ayrılmakta ve daha sonra PCA her bir bölgede gerçekleştirilmektedir. Sistemin esas sınıflandırıcısı olarak bir Öğrenme Vektörü Niceleme (LVQ – Learning Vector Quantization) ağı kullanılmaktadır. (Omatu, Yoshioka & Kosaka, 2007)

Khashman, Şekeroğlu ve Dimililer tarafından 2006 yılında yayımlanan makalede Akıllı Madeni Para Tanıma Sistemi (ICIS – Intelligent Coin Identification System) kullanılarak slot makinelerinin kötüye kullanımı engellemek amacıyla 1 Türk Lirası ve 2 Euro madeni paraları arasında sınıflandırma sağlanmıştır. Bu bağlamda sistem iki aşamadan oluşmuştur, ilk aşamada madeni paraların anlamlı örüntüleri elde edilir. İkinci aşamada elde edilen madeni para görüntüleri geri yayılımlı YSA ile işlenerek paraların % 95.83 oranında tanınması sağlanmaktadır. (Khashman, Şekeroğlu & Dimililer, 2006)

Nastoulis, Leros ve Bardis tarafından 2006 yılında yayımlanan makalede olasılıksal sinir ağı modellerine dayalı banknot tanıma gerçekleştirilmiştir. Bu yöntemde, taranan para birimi görüntülerinde ağın giriş değerlerini oluşturan sayısal niteliklerin, öbek değerlerinin, alınması için simetrik eksen maske kümesi kullanılmıştır. Önerilen yöntem yaklaşık %100'lük başarım sağlamıştır. (Nastoulis, Leros & Bardis, 2006)

3. BÖLÜM

YAPAY ZEKÂ

3.1 Yapay Zekâ

YZ, bir bilgisayarın veya bilgisayar kontrolündeki bir robotun çeşitli faâliyetleri zeki canlılara benzer şekilde yerine getirme kabiliyetidir. (Artificial Intelligence, 2019) YZ'nin gelişimine Karel Čapek[1]'in 1920 yılında yazdığı *Rossum's United Robots* kitabı öncülük etmiştir. Yazdığı kitapta ilk kez "*robot*" sözcüğü yer almıştır. "Çalışmak" anlamına gelen robot sözcüğü, Çekçe *robota* ve Almanca *arbeit* sözcükleriyle aynı kökten gelmektedir. Arthur C. Clarke[2], Robert A. Heinlein[3] ve Isaac Asimov[4] gibi yazarlar bilim-kurgu eserleriyle bilim, teknoloji ve Dünya edebiyatına önemli ölçüde katkı sağlamışlardır. Yazılan eserler Dr. Who, Star Wars, Terminator, Star Trek, I Robot, Wall-E gibi dünyaca ünlü filmlere ilham kaynağı olmuştur.

İnsana benzeyen ve insan gibi düşünen makinelerin gelişimi düşüncesi antik çağlara dayanmaktadır. Eski Mısır'da piramit ve tapınakların yapımında kullanılan makinelerden, Antik Yunanların Güneş Sistemindeki gezegenlerin ve yıldızların tahminî yerlerin hesaplamasını sağlayan Antikythera makinesinden ve Aristoteles'in *Politika* adlı eserinde akıl-mantık ilişkisinin efendi-köle biçiminde ayrışmasını sağlayacak emirlere itaât eden otomatlardan bahsedilmektedir.

Blaise Pascal'ın icat ettiği ilk hesap makinesi toplama ve çıkarma işlemlerini gerçekleştirirken bu makine Gottfried Leibnitz tarafından geliştirilerek çarpma ve bölme işlemleri de eklenmiştir. 19. yüzyılda Charles Babbage'in tasarladığı Fark Makinesi

[1] Çekoslovakyalı bilim-kurgu yazarı, robot kavramının öncüsü.
[2] Britanyalı bilim-kurgu yazarı, fütürist ve televizyon yapımcısı.
[3] Amerikalı bilim-kurgu yazarı.
[4] Amerikalı akademisyen ve bilim-kurgu yazarı.

(*Difference Engine*) polinom fonksiyonlarının hesaplanması için öngörülen bir makine olarak icat edilmiştir. Ancak bazı olanaksızlıklardan dolayı bu makine tamamlanamamıştır. Bunun üzerine Fark Makinesinin daha gelişmiş bir hâli olan Analitik Makine (*Analytics Engine*) geliştirilmeye başlamıştır. Analitik Makinenin ayırt edici özelliği, delikli kart kullanılarak programlanabilir olmasından kaynaklanmaktadır. Programlama özelliğini de Ada Byron Lovelace geliştirerek ilk programlanabilir bilgisayar olmasını sağlamıştır.

YZ'nin başlangıcı bilgisayar bilimcisi Alan Turing'in 1950 yılında geliştirdiği Turing Testi olarak kabul edilmektedir. Bu test ile "Bilgisayarlar düşünebilir mi?" sorusunun yanıtlanması istenmektedir. Turing Testine göre bir denek ile bir sorgulayıcı, bir terminal aracılığıyla haberleştirilir. Sorgulayıcı, deneğin insan yada makine olup olmadığını anlayamazsa denek Turing Testini geçmiş sayılır. (Turing, 1950)

YZ'nin gelişimiyle birlikte birçok alt alanlar ortaya çıkmıştır. Bunlar YSA, Makine Öğrenmesi, Derin Öğrenme, Genetik Algoritmalar ve Bulanık Mantıktır.

YSA, insan gibi düşünen otomatlar tasarlayabilmek için biyolojik sinir ağlarından esinlenilerek matematiksel işlemler aracılığıyla çıkarım sağlayan modellerdir. Bu model sayesinde birçok alanda sınıflandırma ve kümeleme işlemleri sağlanmaktadır. Başlıca uygulama alanları sınıflandırma, tahmin ve modelleme olarak ele alınmaktadır. Ses tanıma, finans, bilgisayar görüsü, tıp ve daha birçok endüstri alanında kullanılmaktadır.

Makine Öğrenmesi, işlenmiş verilerdeki özelliklerin belirlenerek tahmin ve çıkarım yapılması işlemidir. Kaynak olarak kullanılan veriler üzerinden belirli algoritmalar kullanılarak çıkarım yapılmaktadır.

Derin Öğrenme, YSA'da bulunan gizli katman sayısının birden fazla arttırılmasıyla veri kümesinin ayrıntılı bir şekilde eğitilmesidir. Yapısal olarak YSA ile aynı olup, makine öğrenmesinin bir alt kümesidir. Derin öğrenme ağlarında yer alan katmanlardan her biri aracılığıyla ayrı birer özellik incelenir. İşlenmemiş verilerin öğrenilmesi için kullanılmaktadır.

Genetik Algoritmalar, kronolojik olarak 1967 yılında J. D. Bagley tarafından sunulan ve 1975 yılında öğrencisi John Holland tarafından geliştirilen evrimsel yaklaşım ilkeleri doğrultusunda rastlantısal araştırma yöntemlerini kullanarak kendi kendine öğrenme ve karar verme sistemlerinin düzenlenmesini hedef alan bir araştırma tekniğidir.

Bu bağlamda önemli özelliklerinden biri bir grup üzerinde çözümü araması ve bu sayede çok sayıda çözümün içinden en iyiyi seçmesidir. (Bagley, 1967) (Holland, 1975)

Bulanık Mantık, ilk olarak 1956 yılında duyurulmuştur. Bulanık mantığın 1965 yılında Lotfi Aliasker Zadeh tarafından yayımlanan bir makalede, insanların özel verilerini işleyebilme, onların deneyimlerinden ve önsezilerinden yararlanarak uygulamalara çalışabilme yeteneği verdiği söylenmektedir. Buna göre "az", "çok", "pek az", "pek çok", "biraz", "biraz çok" gibi günlük yaşamda sıkça kullanılan dilsel niteleyiciler (belirteç/zarf) doğrultusunda bir denetim gerçekleştirir. Bu durum da sözel ifadelerin matematiksel karşılıkları olarak klasik mantığın temeli olan (0, 1) ikili seviyesi değil, [0, 1] aralığında çok seviyeli işlemleri ifade etmektedir. (Zadeh, 1965)

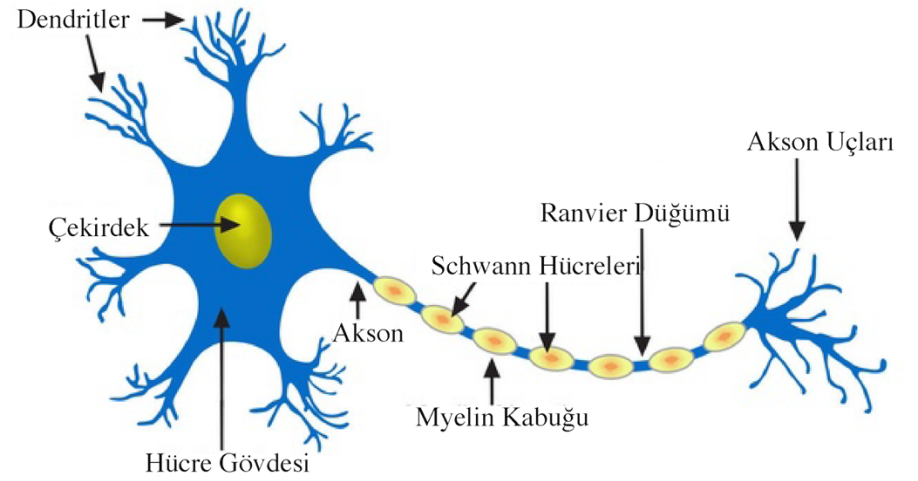
3.2 Yapay Sinir Ağları

YSA, biyolojik sinir ağlarının modellenerek insan beyni gibi düşünme ve karar vermesini sağlayan yapılardır. Her ne kadar biyolojik sinir ağlarına benzetilmeye çalışılsa da biyolojik sinir ağlarının karmaşıklığını yansıtmaktan uzaktır.

Sinir sistemi ve vücudun yönetim merkezi olan insan beyni, yaklaşık olarak 1.5 kg ağırlığında olup, tahminî olarak 10^{11} sinir hücresinden oluşmaktadır. İnsan davranışının temelini oluşturan öğrenme, hatırlama, düşünme ve algılama gibi tüm bilişsel davranışlar sinir hücreleri (nöron) aracılığıyla gerçekleşmektedir. (Akpınar, 2014)

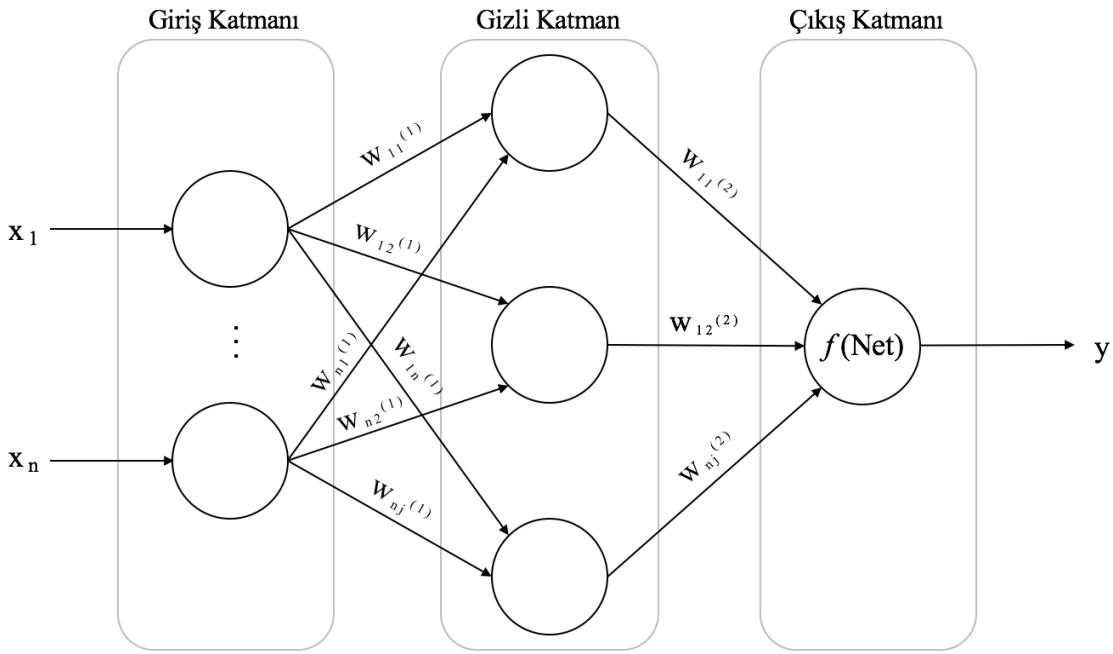
YSA'nın başarılı örneklerinin çoğu denetimli öğrenme sistemi ile çalışır. Sonuçların doğruluğu için eğitim amacı ile kullanılacak veri kümelerinin büyük, çeşitli ve doğru etiketlenmiş olması gerekir. Bu ise maliyetlidir. (Gürsakal, 2017)

Bir insanın doğuştan itibaren sahip olduğu sinir hücreleri yaşamı boyunca yenilenmemektedir. Sinir hücrelerinin büyümesiyle aralarında bağlantı kurması beynin gelişmesi ve ağırlık kazanmasını sağlamaktadır. Bir sinir hücresi Şekil 3.1'de gösterildiği üzere hücre gövdesi, dendrit ve aksondan meydana gelmektedir. Hücreler arasında bilgi alışverişini sağlayan aksonlar, Schwann hücreleri tarafından oluşturulan sinir akımının daha hızlı bir şekilde aktarımını sağlayan Myelin kabuğu ile kaplıdır. Bu kabuk bazen Ranvier boğumu adı verilen noktalarda kesintiye uğramaktadır. Sinir hücreleri arasındaki iletişim de sinaps adlı birleşme yerinde gerçekleşmektedir. Bu iletişim elektro-kimyasal bir süreç olan sinir akımları ile sağlanmaktadır.



Şekil 3.1 Sinir Hücresi

Bir YSA modeli, biyolojik sinir hücresine benzetilerek Şekil 3.2'de gösterildiği üzere, katmanlar (*layer*) ve bu katmanlar arasında işlem yapılabilmesi için hesaplama elemanları ile bağlantılardan oluşmaktadır. Birçok YSA modelinde kullanılan hesaplama elemanları yapay sinir hücresi (*neuron*), düğüm (*node*) ve birim (*unit*) olarak adlandırılmaktadır.



Şekil 3.2 Yapay Sinir Ağı

Bir YSA'da genel olarak 3 katman bulunmaktadır. Birinci katman giriş değerlerinin bulunduğu Giriş Katmanıdır (*Input Layer*). İkinci katman, ağırlık değerlerinin toplandığı Gizli Katmandır (*Hidden Layer*). Gizli katman, çalışılan modele göre farklılık göstermektedir. Gizli katman basit sinir ağlarında hiç olmadığı gibi daha karmaşık sinir ağlarında daha başarılı sonuçlar elde etmek için bir yada birden fazla sayıda olabilmektedir. Gizli katmanda bir düğümün net girdisi Toplama Fonksiyonları (yada Transfer/İletim Fonksiyonları) ile hesaplanmaktadır. Çizelge 3.1'de Toplama Fonksiyonu olarak kullanılabilecek fonksiyonlar listelenmektedir. Son katman, Çıkış Katmanıdır (*Output Layer*). Bu katmanda elde edilen net girdi değerleri Etkinleştirme Fonksiyonları ile çıkış sonucunu üretmektedir. Çizelge 3.2'de Etkinleştirme Fonksiyonları listelenmektedir.



Şekil 3.3 Yapay Sinir Hücresi

Bir yapay sinir hücresi Şekil 3.3'te görüldüğü üzere girdi değerleri (*v*), işlemlerin gerçekleştirildiği gövde ve çıktı değerlerinin sağlandığı kısımlardan oluşmaktadır. Gövde kısmında girdi değerleri ve bu değerlerin ağırlıklarının çarpılmasıyla elde edilen ağırlıklı ortalamanın değeri, Çizelge 3.1'de belirtilen herhangi bir Toplama Fonksiyonu aracılığıyla hesaplanmaktadır. Toplama fonksiyonu tarafından üretilen toplam girdi değeri eşik değeriyle (*threshold*) karşılaştırılır ve geçerli durum sağlanırsa Çizelge 3.2'de belirtilen herhangi bir Etkinleştirme Fonksiyonu ile yapay sinir hücresinin çıktı değeri üretilir. Her yapay sinir hücresinin çıktı değerleri birbirinden bağımsız farklı etkinleştirme fonksiyonlarından üretilebileceği gibi hepsi aynı fonksiyon aracılığıyla da hesaplanabilir. Eğer girdi katmanı ve gizli katmanda yer alan yapay sinir hücrelerinin

etkinleştirme fonksiyonu bulunmuyorsa, çıkış katmanında yer alan hücre tüm ağın toplam girdi değerini hesaplayan etkinleştirme fonksiyonu aracılığıyla çıktı değeri elde edilir.

Bir YSA'nın büyüklüğü, sahip olduğu yapay sinir hücrelerinin ve bu hücreler arasında bulunan ağırlıkların toplam parametre sayısıdır. YSA'nın karmaşıklığı ise büyüklüğün parametre sayıları ile belirtilir. Ağdaki parametre sayısı P ise gerekli olan veri sayısı kaba bir kural ile P^2 olur. (Gürsakal, 2017)

Bir YSA'da Model Parametresi ve Hiperparametresi bulunmaktadır. Bir modelin parametresi, kendi içindeki veriler tarafından değeri tahmin edilen bir değişkendir. YSA'da ağırlıklar model parametreleridir. Hiperparametreler ise, modelin dışında tahmin edilebilir ve değeri veriler tarafından tahmin edilemeyen değişkenlerdir. Bunlar modelin parametrelerinin tahmininde yardımcı olurlar, uygulamacı tarafından sezgisel olarak belirlenirler ve kestirimsel modellerde değerleri sık sık ayarlanır. YSA'nın öğrenme hızı bir hiperparametredir. (Gürsakal, 2017)

Çizelge 3.1 Toplama Fonksiyonları

Adı	**Fonksiyon**
Toplam	$Net = \sum_i x_i w_i$
Çarpım	$Net = \prod_i x_i w_i$
Maksimum	$Net = \max(x_i w_i)$
Minimum	$Net = \min(x_i w_i)$
Çoğunluk	$Net = \sum_i sgn(x_i w_i)$
Kümülatif Toplam	$Net = Net(eski) \sum_i sgn(x_i w_i)$

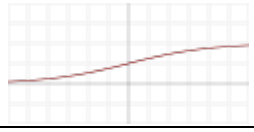
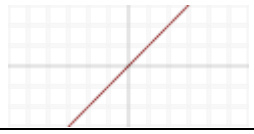
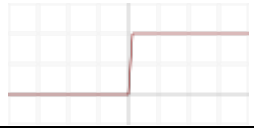
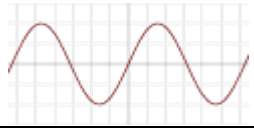
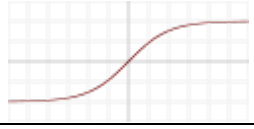
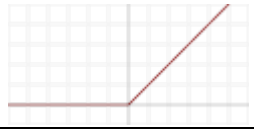
Bir düğümün birden fazla girdisi bulunabilir ancak ağdaki diğer düğümlere göndereceği tek bir çıktısı bulunmaktadır. Düğümler arasındaki bağlantılar birbiri arasında belirlenen ağırlık değerleri (w) ile gösterilir. Bir düğümün girdisi ile ağırlık değerinin çarpımı, bir sonraki düğümün Net Girdisini (*Net Input*) oluşturmaktadır. Her

net girdi, bir düğümde belirlenen Toplama Fonksiyonu için giriş değeridir. Düğümlerin çıkış değerleri Etkinleştirme Fonksiyonları ile hesaplanmaktadır. Net girdi değerinin belirlenmesini,

$$a_i(t) = F_i(a_i(t-1), net_i(t)) \qquad (3.1)$$

eşitliği ile ifade edilen, *i*. düğüm için etkinlik değerinin (*activation value*) hesaplanması sağlayacaktır. Eşitlikte $a_i(t)$, *t.* zamanda *i.* düğümün etkinlik değerini göstermektedir. Bir YSA'nın öğrenim süreci, girdi katmanındaki düğümlere uygulanan girdi değerlerine göre, kabul edilebilir düzeydeki doğru çıktıyı hesaplayacak ağırlık matrisinin oluşturulmasıdır. Bağlantı ağırlıklarının nasıl düzenleneceği uyum sağlama (*adaptation*) yada öğrenme olarak adlandırılmaktadır. (Akpınar, 2014)

Çizelge 3.2 Etkinleştirme Fonksiyonları

Adı	**Şekil**	**Fonksiyon**
Sigmoid		$f(Net) = \frac{1}{1+e^{-Net}}$
Doğrusal (*Linear*)		$f(Net) = Net$
Adım (*Step*)		$f(Net) = \begin{cases} 1 \text{ eğer } Net > \text{eşik değer} \\ 0 \text{ eğer } Net \leq \text{eşik değer} \end{cases}$
Sinüs (*Sinusoid*)		$f(Net) = \sin(Net)$
Hiperbolik Tanjant (*Hyperbolic Tangent*)		$f(Net) = \frac{e^{Net}+e^{-Net}}{e^{Net}-e^{-Net}}$
ReLU		$f(Net) = \max(0, Net)$

Düğümler arasında oluşturulan engelleyici bağlantılar negatif ağırlıklarla, uyarıcı bağlantılar ise pozitif ağırlıklarla çarpılmaktadır. Bu tür bağlantıların dışında kazandırıcı (*gain*), söndürücü (*quenching*) ve uyarıcı (*nonspecific arousal*) özel bağlantı türleri de bulunmaktadır. (Akpınar, 2014)

YSA, Veri Madenciliği alanında sınıflandırma ve kümeleme modeli olarak yer almaktadır. (Silahtaroğlu, 2016)

3.2.1 Yapay Sinir Ağlarının Özellikleri

Birçok model gibi YSA'nın da ayırt edici özellikleri bulunmaktadır. Bunlar:

- Doğrusal Olmama: YSA yapay sinir hücresi gibi doğrusal değildir.
- Öğrenme: YSA istenilen sonucun elde edilebilmesi için girdi ve çıktı verileri arasında eşleştirme yapılır. Bu da bağlantı ve uygun ağırlıkların belirlenmesiyle sağlanır.
- Uyum Sağlama: YSA bağlantı ağırlıklarının değerine göre güncellenebilir.
- Genelleme: YSA'da her sinir hücresi diğer hücrelerden etkilenebileceği için eğitim süresince karşılaşılmayan örneklere de istenen tepkiyi verebilir.
- Hata Toleransı: YSA küçük hatalardan etkilenmez. Girdi değerleri çok fazla sayıda olduğu için hata toleransı yüksektir.

3.2.2 Yapay Sinir Ağlarının Avantaj ve Dezavantajları

YSA'nın avantajları arasında öğrenme yeteneğinin olması ve farklı öğrenme algoritmalarının kullanılması sayılabilir. Matematiksel modele gereksinimi olmadığı için daha önce karşılaşılmayan örnekleri sistemden geçirerek onlar hakkında bir çıktı üretebilme yetisine sahiptir. Doğrusal olmadıkları için karmaşık problemleri daha etkin bir şekilde çözerler. YSA öğrenme süreci içerisinde aldığı geri beslemeleri dikkate alarak ağırlık katsayılarını güncelleyebilir.

YSA'nın dezavantajları arasında aşırı öğrenme (*overfitting*), girdi değerlerinin ölçeklenme zorunluluğu, gizli katmanların sayısının belirsizliği sayılabilir. Genellikle ağırlıklar için başlangıç değerleri sıfıra yakın rastgele değerler seçilir. Böylelikle model yaklaşık olarak doğrusal bir şekilde çalışmaya başlamışken, ağırlık değerlerinin artmasıyla doğrusal olmayan bir yapıya dönüşmektedir. Dolayısıyla değeri yüksek olan

ağırlıklarla yetersiz sonuçlar elde edilmektedir. Çok fazla ağırlık değerinin bulunması da aşırı öğrenme durumunu oluşturmaktadır. Bu durumdan sakınmak için ağ yapısına bir durdurma kuralının eklenmesi gerekmektedir.

3.2.3 Yapay Sinir Ağlarında Öğrenme Türleri

YSA'da girdilerin anlamlı çıktı oluşturması öğrenme türüne göre değişiklik gösterebilmektedir. YSA'nın yapısı ve eğitim kümesinin işlenişine göre 4 farklı öğrenme türü bulunmaktadır. Bunlar:

- Denetimli Öğrenme (*Supervised Learning*)
- Denetimsiz Öğrenme (*Unsupervised Learning*),
- Destekleyici Öğrenme (*Reinforcement Learning*)
- Karma Öğrenme (*Mixed Learning*).

3.2.3.1 Denetimli Öğrenme

Denetimli öğrenme (*Supervised Learning*) yönteminde bir eğiticiye gereksinim vardır. Eğitici, veri eğitim kümesi yada YSA sonuçlarının performansını derecelendiren bir gözlemci olabilir. Bu yöntemde girdi bilgisine göre elde edilen çıktı değeri, beklenen değer ile karşılaştırılarak ağırlıkların güncellenmesinde kullanılacak bilgi elde edilir. Girilen değer ile beklenen değer arasındaki fark hata değeri olarak önceden belirlenen değerden küçük oluncaya kadar devam edilir. Hata değeri beklenen değerin altına indiğinde bütün ağırlıklar sabitlenerek eğitime son verilir.

3.2.3.2 Denetimsiz Öğrenme

Denetimsiz öğrenme (*Unsupervised Learning*) yönteminde eğitici bulunmamaktadır. Bu yöntemde ağın doğru çıktı değerleri hakkında bilgisi yoktur. Beklenen çıktı değerleri olmadan girdi bilgilerinin özelliklerine göre ağırlık değerleri güncellenmektedir. Denetimsiz öğrenme yöntemine örnek olarak Hebbian ve Kohonen'in Özörgütlemeli Eşleştirme Ağları gösterilebilir.

3.2.3.3 Destekleyici Öğrenme

Destekleyici öğrenme yönteminde denetimli öğrenme yönteminde olduğu gibi eğitici bulunmaktadır. Bu yönteme göre ağ, doğrudan gerçek çıkış değerini vermez. Eğitici, çıkış değerinin doğru yada yanlış olarak değerlendirmesini yapar. Eğiticinin sağladığı sinyal doğrultusunda öğrenmeye devam eder. Performans genellikle ikili sayılıdır ve model denetim süreci sırasında başarısını gösterir.

3.2.3.4 Karma Öğrenme

Farklı öğrenme yöntemlerinin tek sistemde kullanıldığı öğrenme türüdür. İşleyiş bakımından iki çeşit karma öğrenme yöntemi bulunmaktadır. Bu yöntemler:

- Çevrimiçi Öğrenme (*On-line Learning*): Bu yöntemde sistemler gerçek zamanlı çalışırken bir yandan fonksiyonlarını yerine getirip, diğer yandan öğrenmeye devam eder. Her giriş ve çıkış değerlerinden sonra değişkenler güncellenir.
- Çevrimdışı Öğrenme (*Off-line Learning*): Bu yöntemde sistemin öğrenmesi gereken yeni bilgiler söz konusu olduğunda sistem kullanımdan çıkarılmakta ve çevrimdışı olarak yeniden eğitilmektedir.

3.2.4 Yapay Sinir Ağlarında Öğrenme Kuralları

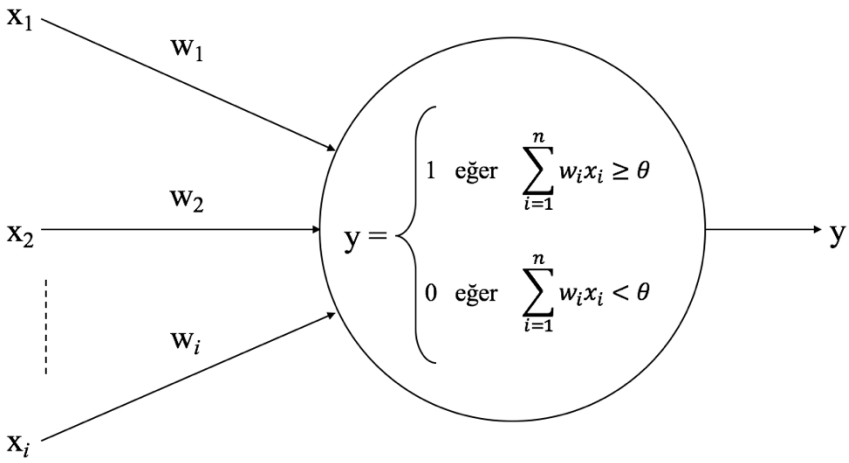
YSA'da geliştirildiği ilk zamanlardan bugüne kadar pek çok öğrenme kuralı önerilmiştir. Bir YSA'nın temel yapısını ve işleyişini betimleyen kurallar kronolojik sıralamaya göre Perceptron, Hebb, Delta, Kohonen ve Hopfield kurallarıdır.

3.2.4.1 Perceptron

YSA'nın atası olarak kabul edilen perceptron (algılayıcı), 1957 yılında Cornell Üniversitesinde akademisyenlik yapan psikolog Frank Rosenblatt tarafından geliştirilmiştir. Perceptron, belirli bir eğitim sürecinden sonra farklı örüntüleri birbirinden ayırt edebildiği için öğrenen bir cihazdır. (Rosenblatt, 1958)

En basit bir YSA olan perceptron'un birden fazla girişi olabileceği gibi tek çıkışı bulunmaktadır. Şekil 3.4'te örnek bir perceptron yer almaktadır. Bu örnekte çıkış değeri

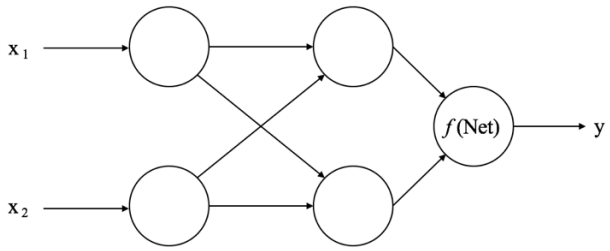
0 yada 1 olmaktadır. Girişlerin ağırlıklı toplamı eşik değeri (θ) ile karşılaştırılır. Ağırlıklı toplam eşik değerinden büyükse 1, aksi durumda sonuç 0 olmaktadır. Perceptronlar genellikle sınıflandırma işlemi için kullanılmıştır.



Şekil 3.4 Perceptron

1969 yılında Marvin Minsky ve Seymour Papert tarafından yayımlanan çalışmada, F. Rosenblatt tarafından önerilen tek katmanlı perceptron'un yetenek ve kısıtları belirtilmiştir. Buna göre, öğrenme, uyum sağlama ve öz örgütleme gibi doğrusal ayrılabilir örüntülerin sınıflandırılabildiği ancak problemlerin zorlaşmasıyla doğrusal ayrılabilir örüntülerin sınıflandırılamadığı kanıtlanmıştır. (Minsky & Papert, 1969)

Önermeler mantığında yer alan "ve", "veya" ve "değil" işlemleri tek katmanlı perceptron kullanılarak hesaplanabilirken, "özel-veya" işlemi doğrusal ayrılabilir olmadığından tek katmanlı perceptron ile hesaplanamamaktadır. Bunun için "özel-veya" işlemini hesaplamak üzere çok katmanlı perceptron kullanılmaktadır. Şekil 3.7'de çok katmanlı perceptron görülmektedir.



Şekil 3.5 Geliştirilmiş Perceptron

3.2.4.2 Hebb Kuralı

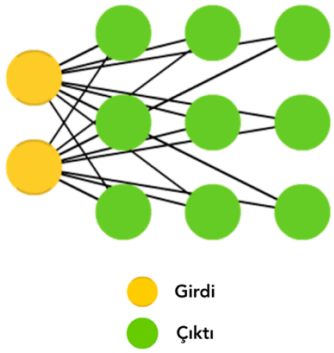
YSA öğrenme kuralları içerisinde geliştirilen ilk kuraldır. Donald Hebb tarafından 1949 yılında geliştirilmiştir. Temel olarak, bir sinir başka bir sinirden girdi değeri almışsa ve her ikisi de etkin ise sinirler arasındaki boyut güçlendirilir. (Hebb, 1949)

3.2.4.3 Delta Kuralı

En çok kullanılan kurallardan biri olan Delta kuralı, 1960 yılında Bernard Widrow ve Marcian Hoff tarafından geliştirilmiştir. Aynı zamanda Widrow-Hoff ve En Küçük Ortalamalar Karesi (*Least Mean Square*) olarak adlandırılmaktadır. Bu kurala göre, bir sinirin gerçek çıktısı ile istenilen çıktı değeri arasındaki farkı azaltmak için girdi bağlantılarının sürekli olarak geliştirilmesi gerekmektedir. Bu kural ağ hatasının karesini azaltmak için bağlantı boyutlarını değiştirir. Hata bir önceki katmana geri gönderilir. Her bir zaman dilimi için bir hata şeklinde bu geri çoğaltma işlemi ilk katmana ulaşılıncaya kadar devam eder. (Widrow & Hoff, 1960)

3.2.4.4 Kohonen Kuralı

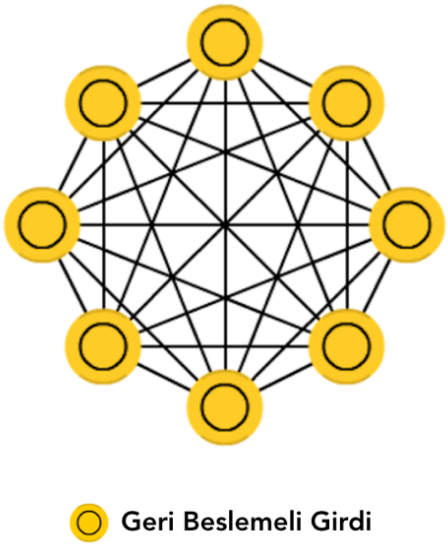
1982 yılında Teuvo Kohonen tarafından biyolojik sistemlerdeki öğrenmeden esinlenilerek geliştirilmiştir. Bu kurala göre, sinirler öğrenmek için elverişli durum yada ölçülerini güncellemek için yarışırlar. En büyük çıktı ile işlenen sinir, kazananı belirtir ve komşularına bağlantı değerlerinin güncellemeleri için izin verilir. Şekil 3.6'da Kohonen kuralının işleyişi görülmektedir. (Kohonen, 1982)



Şekil 3.6 Kohonen Kuralı (van Veen & Leijnen, 2019)

3.2.4.5 Hopfield Kuralı

1982 yılında John Hopfield tarafından geliştirilmiştir. Bu kurala göre, beklenen çıktı ve girdinin her ikisi de etkin yada durgun ise öğrenme katsayısı kadar ağırlık değerleri güncellenir. Ağ, her bir elemanı işleyerek ikili sayı biçimi oluşturur. Bunlar girdilerin toplamı ve çıktı sayılarının 0 yada 1 olarak hesaplanmasıdır. (Hopfield, 1982)



Şekil 3.7 Hopfield Ağı (van Veen & Leijnen, 2019)

3.2.5 Yapay Sinir Ağı Türleri

YSA, verilerin işlenmesi ve ağırlık katsayılarının güncellenmesi bakımından İleri Beslemeli (*Feed-Forward*) ve Geri Beslemeli (*Backpropagation*) olarak ikiye ayrılmaktadır.

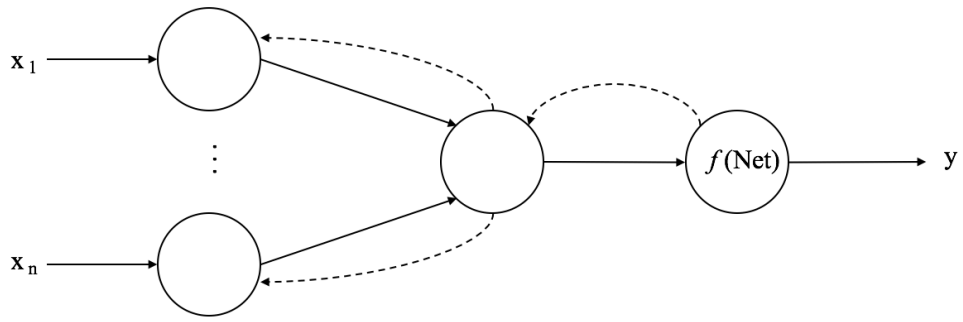
3.2.5.1 İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli öğrenen ağlar hiyerarşik yapıdadır. Girdi, çıktı ve en az bir gizli katman olmak üzere üç katmandan oluşmaktadırlar. Gizli katman ve gizli katmandaki düğüm sayısı değiştirilebilir. Düğüm sayısının artması ağın hatırlama yeteneğini arttırmakla birlikte öğrenme işleminin süresini de uzatmaktadır. Düğüm sayısının azaltılması eğitim süresini kısaltmakta fakat hatırlama yeteneğini azaltmaktadır. Giriş katmanındaki her bir düğüm gizli katmandaki her düğüme, gizli katman birden fazla ise

bu katmandaki her bir düğüm kendisinden sonra gelen katmandaki her düğüme bağlıdır. Bir katmandaki hiçbir düğüm kendi katmanındaki diğer bir düğüme bağlı değildir. Her katmanın çıktı değerleri bir sonraki katmanın girdi değerleridir. Bu şekilde girdi değerlerinin ağın girişinden çıkışına doğru ilerlemesine İleri Besleme (*Feed Forward*) denir. Şekil 3.2'de verilen YSA bu tür ağlara örnektir.

3.2.5.2 Geri Beslemeli Yapay Sinir Ağları

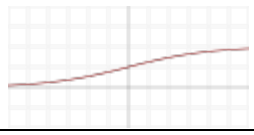
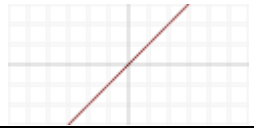
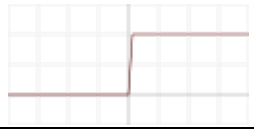
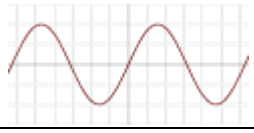
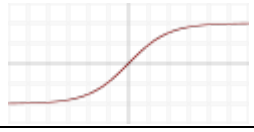
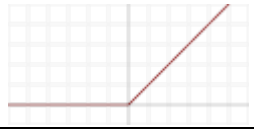
Geoffrey Hinton ve James McClelland tarafından geliştirilen bu model, genellikle denetimsiz öğrenme kurallarının uygulandığı ağlarda kullanılmaktadır. Bu tür ağlarda bir sinirin çıkışı diğer tüm sinirlerin girişine bağlıdır. Hatalar, ileri besleme aktarım fonksiyonu tarafından kullanılan aynı bağlantılar aracılığıyla geriye doğru yayılmaktadır. Öğrenme işlemi, bu ağda basit çift yönlü hafıza birleştirmeye dayanmaktadır. Delta ve Hopfield ağı, bu tür yapıya sahip birer YSA'dır. Şekil 3.8'de geri beslemeli yapay sinir ağı yer almaktadır.



Şekil 3.8 Geri Beslemeli Yapay Sinir Ağı

Geri beslemeli YSA kullanıldığında, sonraki katmanların hataları kullanılarak gizli katmanın ağırlıkları güncellenir. Böylece çıktı katmanında hesaplanan hatalar kullanılarak son gizli katman ile çıktı katmanı arasındaki ağırlıklar ayarlanır. Aynı şekilde, bu işlemler ilk gizli katmana ulaşıncaya kadar tekrarlanır. Bu yolla hatalar katmanlar arasında ilgili katmanın ağırlık düzeltmeleri yapılarak geriye doğru yayılır. Tamamlanan çalışma süresi içinde toplam hata en aza indirilinceye kadar bu işlem devam eder. Çizelge 3.2'de verilen etkinleştirme fonksiyonlarının türevleri Çizelge 3.3'te gösterilmiştir.

Çizelge 3.3 Etkinleştirme Fonksiyonlarının Türevi

Adı	Şekil	Fonksiyon
Sigmoid		$f'(Net) = f(Net)(1 - f(Net))$
Doğrusal (*Linear*)		$f'(Net) = 1$
Adım (*Step*)		$f'(Net) = \begin{cases} 0, & Net \neq 0 \ için \\ ?, & Net = 0 \ için \end{cases}$
Sinüs (*Sinusoid*)		$f'(Net) = \cos(Net)$
Hiperbolik Tanjant (*Hyperbolic Tangent*)		$f'(Net) = 1 - f(x)^2$
ReLU		$f'(Net) = \begin{cases} 0, & Net < 0 \ için \\ 1, & Net \geq 0 \ için \end{cases}$

3.3 Derin Öğrenme

Makine Öğrenmesinin bir alt alanı olan Derin Öğrenme terimi George Hinton ve arkadaşları tarafından 2012 yılında yayımlanan makalede ses tanıma sistemlerinde Gizli Markov Modeli kullanmak yerine çok katmanlı Geri Beslemeli Yapay Sinir Ağları için Derin Sinir Ağları şeklinde önerilmiştir. (Hinton vd., 2012)

Derin Öğrenmenin kullanımının artışı birçok alana dâhil olmasına neden olmuştur. Görüntü işleme, doğal dil işleme, sinyal işleme, ses tanıma gibi birçok alanda sınıflandırma işlemleri için kullanılmaktadır. Sınıflandırma oranının doğruluğu, eğitim için kullanılan görüntü sayısının fazla olmasına bağlıdır. Ancak gereğinden fazla verinin kullanımı sistemin öğrenmesini yavaşlatmaktadır. Buna aşırı öğrenme denir.

Son yıllarda popülerliği artan Derin Öğrenme birçok programlama dili ile geliştirilmektedir. (Yazılımın Yükselen Alt Katmanı: 4 Maddede Deep Learning (Derin Öğrenme) Nedir?, 2018) Bu yazılım dilleri:

- Python (%57), ortak dil
- C/C++ (%44), gömülü sistemler ve işlemciler
- Java (%41), Python'a benzer
- R (%37), istatistik ve veri madenciliği
- JavaScript (%28), tamamlayıcı destek dili

3.3.1 Derin Öğrenme Modelleri

Kullanım alanlarının artmasıyla birlikte farklı amaçlara yönelik olarak çeşitli modeller geliştirilmiştir. Görüntü sınıflandırma için ESA, ses ve el yazı tanıma için Özyineli Sinir Ağları (RNN) ile Uzun Kısa Vadeli Bellek Ağları (LSTM), işbirlikçi filtreleme ve öznitelik çıkarımı için Sınırlı Boltzmann Makineleri (RBM), resim ve videoları tanıma, sınıflandırma ve kümeleme için Derin İnanç Ağları (DBN) kullanılmaktadır.

3.3.1.1 Evrişimsel Sinir Ağları

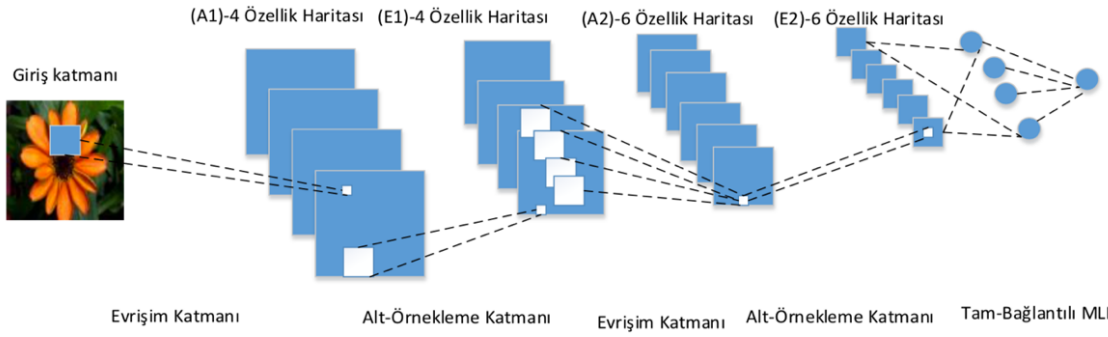
Derin Öğrenme modelleri arasında en çok kullanılanlardan biri ESA'dır. Çok Katmanlı Algılayıcıların (*Multi-Layered Perceptron*) bir türüdür. Bu model aracılığıyla sayısal görüntü işleme teknikleri gerçekleştirilerek sınıflandırma sağlanmaktadır. Sahip olduğu katmanlı yapısı, resimler üzerindeki gizli öznitelikleri çıkararak işlem yapmaktadır.

Sözcük anlamı olarak evrişim, matematiksel iki sinyalin birleşip, üçüncü sinyalin oluşturulması işlemidir. Bir evrişim, bir *g* fonksiyonu diğer bir *f* fonksiyonunun üstünden geçerken çakışma miktarını gösteren integraldir. YSA'da evrişim işlemi, sıkça gerçeklenen bir filtreleme işlemi olarak tanımlanmaktadır. (Gürsakal, 2017)

1988 yılında Yann LeCun ve arkadaşları tarafından yayımlanan makalede önerilmiştir. Makalede önerilen LeNet mimarisi, 1998'li yıllara kadar geliştirilmiştir. Bu mimaride, alt katmanlar art arda yerleştirilmiş evrişim ve maksimum havuzlama

katmanlarından oluşmaktadır. Diğer üst katmanlar ise tamamen bağlı geleneksel Çok Katmanlı Algılayıcıya karşılık gelmektedir. (LeCun vd., 1988)

Gizli katman sayısının artması belirli aşamaya kadar olumlu sonuçlar elde edilmesini sağlamaktadır. Ancak katmanların fazlalığı bulma hızını düşürdüğü için performansı olumsuz yönde etkilemektedir. Şekil 3.9'da bir ESA'nın yapısı gösterilmiştir. Ağda iki evrişim katmanı, evrişim katmanlarını takip eden iki alt örnekleme katmanı ve en üst katmanda tam bağlantılı katman bulunmaktadır.



Şekil 3.9 Bir Evrişimsel Katmanın Grafiksel Gösterimi (Cengil & Çınar, 2016)

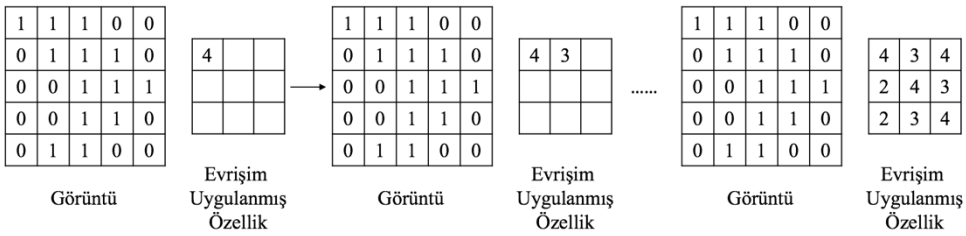
ESA'da bir filtre, ağırlıklardan oluşan bir matris olarak gösterilir. Filtre, bir girdi öbeğinin bir özelliğe ne kadar benzediğinin ölçülmesini sağlamaktadır. Sınıflandırma, girdi olarak alınan verinin çıktısının ne olduğunun yada hangi sınıfta bulunma olasılığının hesaplandığı bir işlemdir. Şekil 3.10'da bir filtre örneği gösterilmiştir.

1	0	1
0	1	0
1	0	1

Şekil 3.10 Bir Filtre Örneği

Şekil 3.10'da verilen filtre örneği bir görüntü üzerinde kaydırılıp, filtredeki katsayıların görüntüdeki piksellerin katsayılarıyla çarpılması ve sonucun toplamıyla

3x3'lük matris şeklinde elde edilir. Şekil 3.11'de Şekil 3.10'da verilen filtre aracılığıyla evrişim işleminin evreleri ve sonucu gösterilmiştir.



Şekil 3.11 Evrişim İşlemi

Evrişim işleminde uygulanmak istenen filtre amacına yönelik olarak farklılık gösterebilir. Bu durumda aynılık, kenar belirleme, keskinleştirme, bulanıklaştırma gibi işlemler için farklı filtreler kullanılabilmektedir. Çizelge 3.4'te farklı işlemlerin gerçekleştirilmesini sağlayan filtreler yer almaktadır.

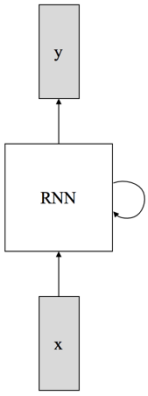
Çizelge 3.4 Filtreler

İşlem	Filtre
Aynılık	$\begin{bmatrix}0 & 0 & 0\\0 & 1 & 0\\0 & 0 & 0\end{bmatrix}$
Kenar Belirleme	$\begin{bmatrix}1 & 0 & -1\\0 & 0 & 0\\-1 & 0 & 1\end{bmatrix}$
	$\begin{bmatrix}0 & 1 & 0\\1 & -4 & 1\\0 & 1 & 0\end{bmatrix}$
	$\begin{bmatrix}-1 & -1 & -1\\-1 & 8 & -1\\-1 & -1 & -1\end{bmatrix}$
Keskinleştirme	$\begin{bmatrix}0 & -1 & 0\\-1 & 5 & -1\\0 & -1 & 0\end{bmatrix}$
Bulanıklaştırma	$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$
Gauss Bulanıklaştırma	$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$

3.3.1.2 Tekrarlı Sinir Ağları

TSA modeli ilk olarak Jeff Elman tarafından Basit Tekrarlı Ağ (*Simple Recurrent Network - SRN*) şeklinde tasarlanmıştır. Tümce yapısında yer alan her bir sözcük için gizli kalıpların üzerinde ortalama örüntü kümeleme sonucunda ad ve eylem sınıflarını düzgün bir şekilde ayrılması sağlanmıştır. (Elman, 1990)

RNN, yapay sinir hücreleri arasında bulunan bağlantıların yönlendirilmiş bir döngü oluşturduğu YSA'dır. İleri beslemeli ağlardan farklı olarak, kendi giriş belleğini girdilerin rastgele dizilerini işlemek için kullanmaktırlar.



Şekil 3.12 Tekrarlı Sinir Ağı

3.3.1.3 Uzun Kısa Vadeli Hafıza Ağları

Özel bir RNN türü olan LSTM, 1997 yılında Sepp Hochreiter ve Jürgen Schmidhuber tarafından yayımlanan makalede önerilmiştir. (Hochreiter & Schmidhuber, 1997)

LSTM mimarisi girdi, unutma ve çıktı kapılarıyla birlikte, blok girişi, Sabit Hata Döngüsü, çıkış etkinleştirme fonksiyonu ve gözetleme bağlantılarından oluşmaktadır. Blokun çıktısı tekrarlı bir şekilde kendi girişine ve tüm kapılara bağlanmaktadır. Kendi durumunu sıfırlamak için unutma kapısı (Gers, Schmidhuber, & Cummins, 2000) ile kesin zamanlamaları öğrenmeyi kolaylaştırmak için gözetleme bağlantıları (Gers & Schmidhuber, 2000) sonradan eklenmiştir.

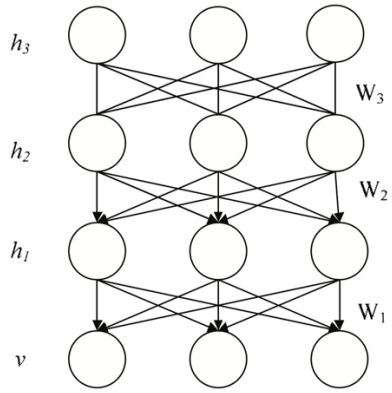
3.3.1.4 Sınırlı Boltzmann Makineleri

RBM ilk olarak 1986 yılında Harmonium (Smolensky, 1986) adıyla ortaya atılmış ancak 2006 yılında Hinton ve Salakhutdinov tarafından yayımlanan makalede (Hinton & Salakhutdinov, 2006) ön plâna çıkarılmıştır. Boltzmann Makinelerinin bir türü olup, girdi kümesi üzerinde olasılık dağılımını öğrenebilen ve öznitelik çıkarımını sağlayan bir YSA'dır.

Görünür ve gizli olmak üzere aralarında simetrik bir bağlantı bulunan iki parçalı çizgelerden oluşmaktadırlar. Bir çizge içinde bulunan düğümlerde aralarında bağlantı bulunmayıp, RBM gizli birimleri arasında bağlantı bulunmaktadır.

3.3.1.5 Derin İnanç Ağları

DBN, RBM'lerin yığını olarak tanımlanabilir. Tek RBM'nin sağladığı öznitelik çıkarma işlemi DBN'nin sahip olduğu yapısından dolayı daha fazla özniteliğin çıkarılmasını sağlamaktadır. Üst katmanların aralarında simetrik bağlantılara sahip olduğu, ileri beslemeli ağların aksine yönlendirilmemiş, üretken bir modeldir. Alt katmanlar işlenen bilgiyi üstlerindeki katmanlardan yönlendirilmiş bağlantılardan alır.



Şekil 3.13 Derin İnanç Ağı

3.3.1.6 Derin Oto-Kodlayıcılar

Diablo Ağı olarak da bilinen DAE, denetimsiz öğrenme için kullanılan bir YSA'dır. Bir veri kümesi için boyut indirgeme amacıyla bir kodlama öğrenmeyi hedefler. Girdi verisinin sıkıştırılmış gösteriminden en iyi özelliklerin öğrenilmesini amaçlayan bir YSA'dır.

DAE'ye göre girdi verisi şifreleme ve şifre çözme işleminden sonra çıktı olarak yine aynı girdiyi görene kadar ağırlıkları güncellenmektedir. Hedefe ulaşıldığında gizli katmandaki düğüm sayısı ile girdi verisi temsil edilmektedir.

3.3.2 Öğrenme Transferi

Bir çalışma alanında öğrenilen bilginin benzer olan başka bir çalışma alanına aktarılması işlemidir. 1901 yılında Edward Thorndike ve Edward Woodworth tarafından psikoloji alanında yayımlanan makalede belirtilmiştir. (Thorndike & Woodworth, 1901)

Öğrenme transferi bilişsel psikolojinin temel alanlarından biri olup, kişilerin belirli bir çalışma alanında elde ettikleri edinimlerin benzer fakat farklı bir çalışma alanına uygulanmasıdır. Bu durum, elde edilen tecrübenin farklı çalışma alanlarında kolaylıkla gerçekleştirilmesine yardımcı olmaktadır.

Makine öğrenmesinde ise belirli bir alanda eğitilmiş olan modelin farklı işlemlerde kullanılmak üzere tekrar eğitilmesidir. Model daha önce eğitildiği için yeni eğitim safhasında öğrenme süresi kısalmaktadır.

3.4 Hata Matrisi

Hata matrisi veya karmaşıklık matrisi, iki veya çok sınıflı sınıflandırma probleminde modelin doğruluğunu ve performansını ölçmek için kullanılmaktadır. Bulunan ve bulunmayan öznitelikler 1 ve 0 ile gösterilmek üzere, matriste yer alan tüm özniteliklere işaretlenerek doğru pozitif, doğru negatif, yanlış pozitif ve yanlış negatif değerler arasındaki ilişkilerin hesaplanmasında kullanılmaktadır.

- *Doğru Pozitif – DP (True Positive – TP)*: Verinin gerçek değerinin pozitif (1) ve tahmin edilen değerin de pozitif (1) olduğu durum.
- *Doğru Negatif – DN (True Negative – TN)*: Verinin gerçek değerinin negatif (0) ve tahmin edilen değerin de negatif (0) olduğu durum.
- *Yanlış Pozitif – YP (False Positive – FP)*: Verinin gerçek değerinin negatif (0) fakat tahmin edilen değerin pozitif (1) olduğu durum.
- *Yanlış Negatif – YN (False Negative – FN)*: Verinin gerçek değerinin pozitif (1) fakat tahmin edilen değerin negatif (0) olduğu durum.

Çizelge 3.5 Hata Matrisinin Hesaplanması

		Gerçek Durum	
		Pozitif (1)	Negatif (0)
Tahmin Edilen	Pozitif (1)	DP	YP
	Negatif (0)	YN	DN

Hata matrisinde doğru ve yanlış değerler arasında performansın ölçümünü sağlayan birkaç ölçüt bulunmaktadır. Bunlar doğruluk, kesinlik, yakalama, doğru negatif oranı, yanlış sınıflandırma oranı, yanlış pozitif oranı ve F1 skorudur.

Doğruluk (*Accuracy*): Sınıflandırma problemlerinde doğru tahminlerin bütün tahminlere oranıdır. Sınıflayıcının hangi sıklıkla doğru olduğunu açıklar.

$$Doğruluk = \frac{DP + DN}{DP + DN + YP + YN} \tag{3.2}$$

Kesinlik (*Precision*): Pozitif tahminde bulunulan verilerin gerçekte hangi oranda pozitif olduğu sorusunu yanıtlamaktadır .

$$Kesinlik = \frac{DP}{DP + YP} \tag{3.3}$$

Yakalama (*Recall*) veya Hassaslık (*Sensitivity*): Doğru Pozitif Oranı (DPO) olarak da adlandırılan bu ölçüt, gerçekte pozitif olanların ne kadarının doğru tahmin edildiğini ölçer.

$$Yakalama = \frac{DP}{DP + YN} \tag{3.4}$$

Doğru Negatif Oranı (*True Positive Rate*) veya Belirlilik (*Specifity*): Doğru Negatif Oranı (DNO) olarak da adlandırılan bu ölçüt, gerçekte negatif olanların ne kadarının doğru tahmin edildiğini ölçer.

$$Belirlilik = \frac{DN}{DN + YP} \tag{3.5}$$

Yanlış Sınıflandırma Oranı (*Misclassification Rate*): Sınıflayıcının hangi sıklıkla yanlış olduğunu açıklar.

$$YSO = \frac{YP + YN}{DP + DN + YP + YN} \tag{3.6}$$

ayrıca,

$$YSO = 1 - Doğruluk \tag{3.7}$$

Yanlış Pozitif Oranı (*False Positive Rate*): Kestirimin doğru ama gerçek durumun yanlış olması durumudur.

$$YPO = \frac{YP}{YP + DN} \tag{3.8}$$

F1 Skoru: Her seferinde kesinlik ve yakalama ölçütleri ile ayrı ayrı uğraşmak yerine ikisini birlikte temsil eden bir sınıflandırma performans ölçütü kullanmak mümkündür. Kesinlik ve yakalama ölçütlerinin ağırlıklı ortalamaları ile hesaplanan F1 Skorunun doğruluk ölçütünden (*accuracy*) daha kullanışlı olduğunu söylemek mümkündür.

$$F1\ Skoru = \frac{2 \cdot Kesinlik \cdot Yakalama}{(Kesinlik + Yakalama)} \tag{3.9}$$

4. BÖLÜM

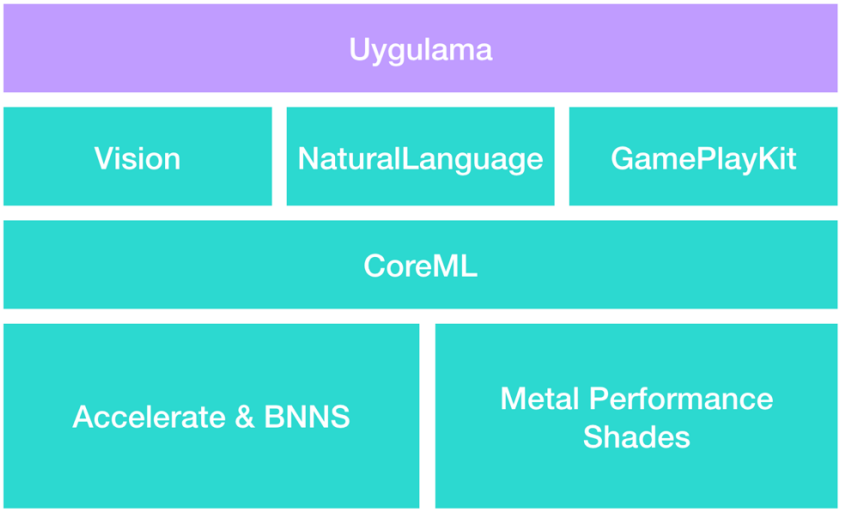
GELİŞTİRİLEN YÖNTEM

Bu tez çalışmasında Türk Lirası, Euro ve Amerikan Doları banknotları derin öğrenme yöntemleri kullanılarak mobil cihazlar üzerinde anlık olarak tespit edilmesi için bir uygulama geliştirilmiştir. Geliştirilen uygulama iOS işletim sistemi üzerinde çalışabilmektedir. Geliştirme aracı olarak XCode tümleşik geliştirme ortamı, geliştirme dili olarak Swift 4.0 kullanılmıştır. Derin öğrenme algoritmasının girdi olarak kullandığı eğitim kümesi fotoğrafları Python programlama dili kullanılarak geliştirilmiş olan yardımcı uygulama ile standartlaştırılmıştır. Derin öğrenme kütüphanesi olarak iOS işletim sistemine ait CoreML ve CreateML kütüphaneleri kullanılmıştır. Eğitim kümesi için 11.400 fotoğraf kullanılmıştır. Eğitim süresi kullanılan veri arttırma yöntemine göre farklılık göstermektedir.

4.1 CoreML ve CreateML Kütüphaneleri

CoreML kütüphanesi, 2017 yılında Apple tarafından duyurulan bir makine öğrenmesi çatısıdır. Kütüphanenin kolay entegre edilebilir yapısı sayesinde nesne tanıma uygulaması kolaylıkla geliştirilebilmektedir. CoreML çatısı ile farklı araçlar kullanılarak geliştirilmiş modeller hızlıca iOS uygulamalarına dönüştürülebilmektedir.

CoreML kütüphanesi, birçok etki alanına özgü işlevsellikler barındırmaktadır. Temel olarak görüntü işleme için *Vision*, doğal dil işleme için *NaturalLanguage* ve karar ağaçları için *GamePlayKit* çatıları kullanılmaktadır. (CoreML, 2017) Şekil 4.1'de CoreML mimarisi görülmektedir.



Şekil 4.1 CoreML Mimarisi

CreateML kütüphanesi ise makine öğrenmesi uzmanlığına gerek kalmadan modellerin oluşturulmasını sağlayan bir makine öğrenmesi çatısıdır. XCode 10 ile gelen bu özellik, gerçek zamanlı olarak modellerin başka bir araç kullanmaya gerek kalmadan oluşturulmasını sağlamaktadır. Bu kütüphane regresyon, görüntü sınıflandırma, sözcük etiketleme, tümce sınıflandırma gibi birçok alanda kullanılmaktadır. Bir uzak sunucuya gerek kalmadan derin öğrenme modeli herhangi bir Macintosh bilgisayarda kolaylıkla oluşturulabilmektedir.[5] (CreateML, 2018) Şekil 4.2'de CreateML kütüphanesindeki iş akışı görülmektedir.



Şekil 4.2 CreateML İş Akışı (CreateML, 2018)

[5] Bu özellik MacOS Mojave (version 10.14) güncellemesi ile kullanılabilmektedir.

CreateML, iOS işletim sistemine dâhil olan makine öğrenmesi altyapısını kullanır. Bu sayede geliştirilen modeller daha hızlı eğitilmektedir. Öğrenme transferi olarak bilinen bu yöntem ile sıfırdan bir model eğitmek yerine, daha önce eğitilmiş olan bir modelin ağırlıkları kullanılarak son katman eğitilir. Birçok resim türünün tanınması için Apple geliştiricileri tarafından büyük bir veri kümesi ile eğitilen *Vision* kütüphanesine ait *VisionFeaturePrint_Screen* derin öğrenme modeli öğrenme transferini sağlamaktadır. Veri kümesinden çıkarılan öznitelikler üzerinde lojistik regresyon eğitimi yapılmaktadır.

4.2 İktisadî Kavramlar

Türk Dil Kurumu (TDK)'na göre döviz, "ülkeler arası ödemelerde kullanılabilen para, çek ve poliçe gibi her türlü ödeme aracı" veya "yabancı ülke parası" olarak tanımlanmaktadır.

Döviz kuru, bir birim ülke parasının diğer birim ülke parasına karşılık olarak ödenmesi gereken miktardır. İki taraflı bir ilişkiyi içerdiğinden Nominal Döviz Kuru olarak da adlandırılmaktadır.

Bir ülkenin para birimi ISO tarafından belirlenen ISO-4217 standardında yer alan kodlar ile gösterilmektedir. (ISO 4217 Currency Codes, 2019) Tez çalışmasında tespit edilebilen para birimleri ve kodları Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1 Kullanılan Para Birimlerinin Listesi

Para Birimi	**Kodu**
Türk Lirası	TRY
Dolar	USD
Euro	EUR

4.3 Döviz Kurunun Hesaplanması

Tez çalışmasında para birimleri arasındaki dönüşümü sağlamak için TCMB'nin resmi web sitesi kullanılmıştır. İlgili sitenin tüm birimleri içeren Application

Programming Interface (API)[6]'i günlük olarak tüm para birimlerinin Türk Lirası cinsinden değerlerini göstermektedir.

4.4 Para Birimleri ve Veri Kümesi

Tez çalışmasında Türkiye Cumhuriyeti para birimi Türk Lirası, Avrupa Birliği para birimi Euro ve Amerika Birleşik Devletleri para birimi Amerikan Doları kullanılmıştır.

Para birimlerine ait veri kümesinde her birime ait farklı sayıda sınıf bulunmaktadır. Türk Lirası para biriminde 6, Amerikan Doları para biriminde 7 ve Euro para biriminde 6 olmak üzere toplam 19 sınıf bulunmaktadır. Bu sınıflar Çizelge 4.2'de yer almaktadır.

[6] http://www.tcmb.gov.tr/kurlar/today.xml

Çizelge 4.2 Para Birimlerinin Ön ve Arka Yüzleri

Birim	Ön Yüz	Arka Yüz
TRY		
USD		
EUR		

Veri kümesinin oluşturulması için toplam 11.400 adet fotoğraf manuel yolla elde edilmiştir. Elde edilen fotoğraflar RGB (Red-Green-Blue) renk uzayında bulunmaktadır. Eğitim ve test kümelerinin seçilmesi Pareto İlkesi[7] doğrultusunda gerçekleştirilmiş olup, %70'i eğitim ve %30'u test girdisi olarak ayrılmıştır. Fotoğraflar para birimlerinin cinsine göre gruplandırılmıştır. Şekil 4.3'te veri kümesine ait örnekler yer almaktadır.

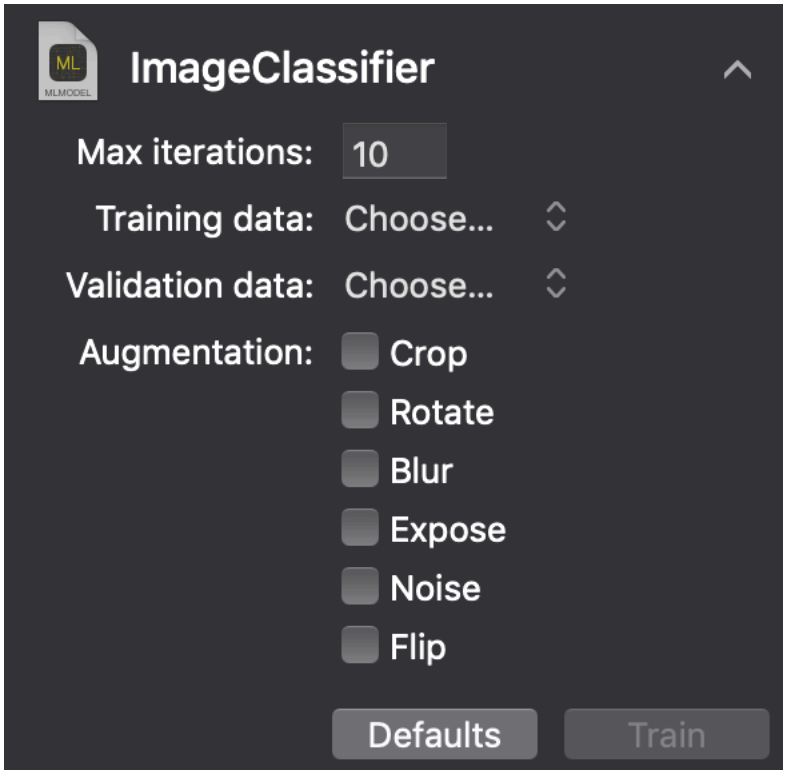
[7] Pareto İlkesi: Çoğu olay için, etkilerin kabaca % 80'i etkenlerin % 20'sinden kaynaklanır.



Şekil 4.3 Veri Kümesine Ait Örnekler

Elde edilen resimler modelle uyumlu olması açısından 300x300 piksel boyutlarında kullanılmıştır. Boyutlandırma işlemi için Python programlama diline ait *Pillow* ve *resizeimage* adlı görüntü işleme kütüphaneleri kullanılarak bir uygulama geliştirilmiştir (EK-2: *image.py*).

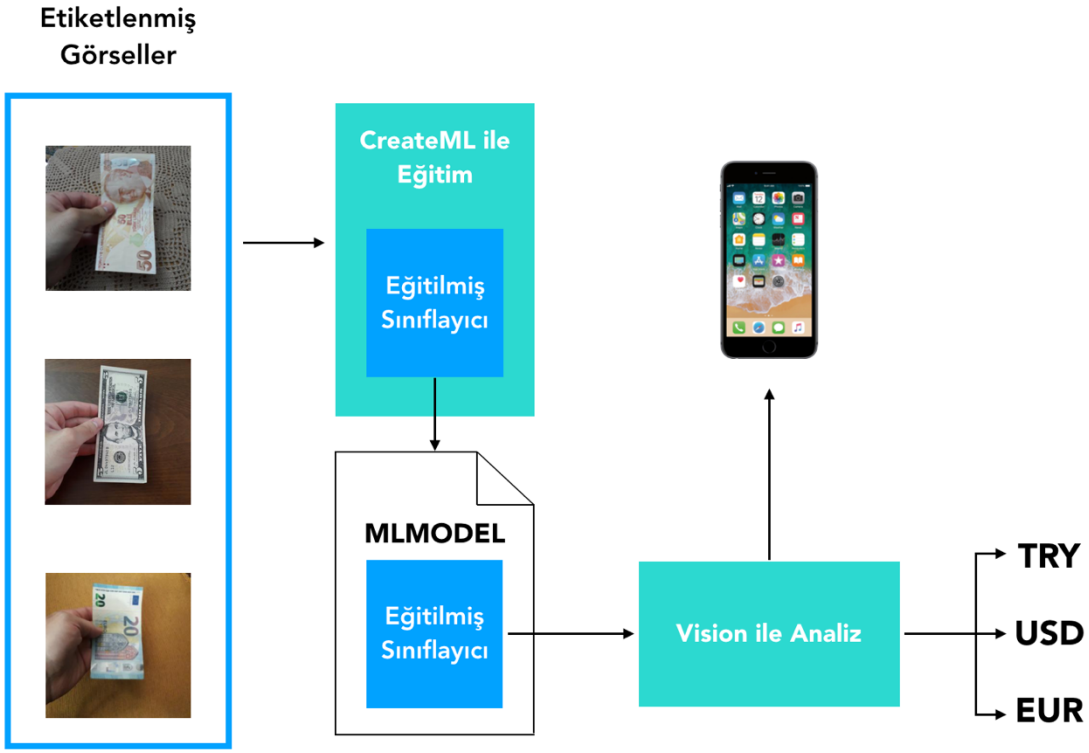
Elde edilen veri kümesi CreateML çatısı ile eğitilmiştir. Çatı dâhilinde bulunan *MLImageClassifierBuilder* sınıfı aracılığıyla resimlerin eğitilmesini sağlayan sınıflayıcı kullanılmıştır. Sınıflayıcı içerisinde veri kümesindeki verilerin daha fazla miktarda olmasını sağlayan veri arttırma özelliği bulunmaktadır. Bu özellik sayesinde kesme, döndürme, bulanıklaştırma, pozlama, gürültü ve ters çevirme işlemleri ile veri arttırma sağlanmaktadır. Veri kümesine kesme, döndürme ve bulanıklaştırma işlemleri uygulanarak yapay bir şekilde veri arttırılması sağlanmıştır. Şekil 4.4'te CreateML kütüphanesinin çalışma parametreleri görülmektedir.



Şekil 4.4 CreateML Kütüphanesi Çalışma Parametreleri

4.5 Modelin Uygulamada Kullanımı

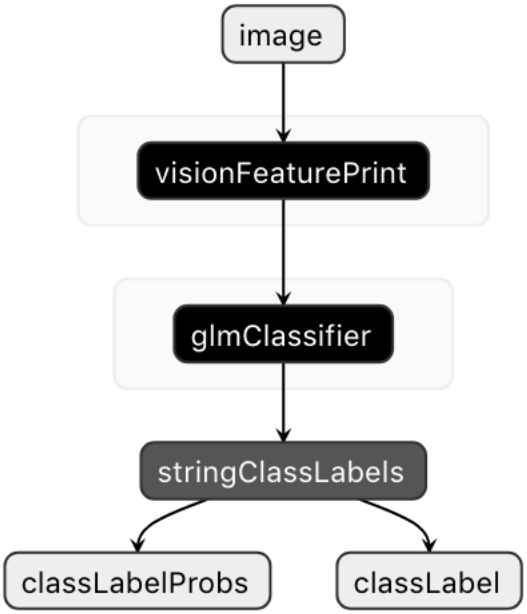
Veri kümesinin CreateML çatısı ile eğitilmesi sonucunda makine öğrenimi dosya formatı olan CoreML dosyasına dönüşümü sağlanmıştır. (*Doviz.mlmodel*) Dönüşüm sonucunda oluşturulan modelin CoreML çatısında yer alan Vision kütüphanesi ile analizi sağlanarak para birimlerin tespit edilmesi işlemi gerçekleşmektedir. Şekil 4.5'te geliştirilen modelin çalışma şekli gösterilmektedir.



Şekil 4.5 Geliştirilen Modelin Çalışma Şekli

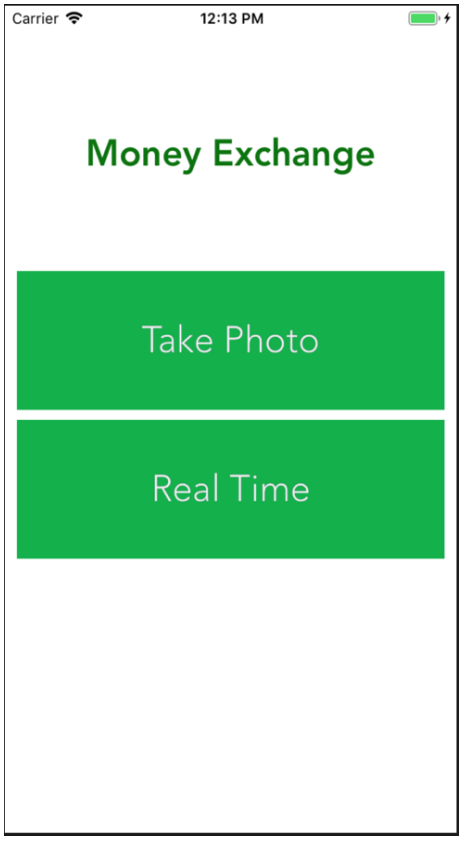
Elde edilen model Türk Lirası, Amerikan Doları ve Euro banknotlarının tespit edilmesini sağlamaktadır. Tespit edilen para birimleri arasında günlük TCMB döviz kurları üzerinden dönüşüm sağlanabilmektedir.

Model, girdi olarak 300x300 piksel boyutlarında eğitilen resimleri almaktadır. Çıktı olarak her sınıf için olasılık değeri *classLabelProps* ve en olası sınıf olan *classLabel* döndürülmektedir. *classLabelProps* bir sözlük (*dictionary*) ve *classLabel* bir dizge (*string*) veri yapısındadır. *visionFeaturePrint*, Apple tarafından oluşturulan derin öğrenme modeline öğrenme transferi uygulayarak elde edilen veri kümesinin eğitilmesini sağlamaktadır. *glmClassifier*, iOS dâhilinde yer alan ESA'nın çıktı katmanına lojistik regresyon uygulayarak çıktının elde edilmesini sağlamaktadır. Şekil 4.6'da modelin mimarisi gösterilmektedir.



Şekil 4.6 Geliştirilen Modelin Mimarisi

Geliştirilen uygulamada para birimlerinin tanınması hem fotoğraf çekerek hem de gerçek zamanlı tanınma ile sağlanmaktadır. Bu bağlamda daha önceden çekilmiş ve uygulama içinde çekilen fotoğraflar ile uygulama içinde gerçek zamanlı olarak görüntüler elde edilerek para birimlerinin tespiti sağlanmaktadır. Şekil 4.7'de geliştirilen uygulamanın açılış ekranı yer almaktadır.



Şekil 4.7 Geliştirilen Uygulamanın Açılış Ekranı

Uygulamanın açılış ekranında yer alan ilk buton (*Take Photo*), fotoğraf çekerek banknot tanıma işlemini gerçekleştiren ekrana geçişi sağlamaktadır. Açılan ekranda sol alt butona tıklanarak yeni bir fotoğraf çekilmesi veya galeride bulunan bir fotoğrafın kullanılması istenmektedir. Her iki şekilde de banknot fotoğrafının elde edilmesi sonucunda tanınma işlemi gerçekleşmektedir.

Her iki yöntemden birinin seçilmesi durumunda tanıma işleminin ardından ekranın alt kısmında yer alan panelin sol kısmında hangi birimlerin ne kadar oranla tespit edilebildiği bulunmaktadır. Sağ kısımda ise sol kısımdaki ilk birimin en az %70 tahmin oranında olması durumunda diğer birimlere TCMB günlük döviz kurları üzerinden dönüşümü sağlanmaktadır. Şekil 4.8'de fotoğraf çekimi ile banknot tanıma işlemi yer almaktadır.



Şekil 4.8 Fotoğraf Çekimi ile Banknot Tanıma

Uygulamanın açılış ekranında yer alan ikinci buton (*Real Time*), gerçek zamanlı banknot tanıma işlemini gerçekleştiren ekrana geçişi sağlamaktadır. Açılan ekran ile doğrudan bir banknot okutulup tespit edilebilmektedir. Bir önceki seçenekte yer alan panelin aynısı bu ekranda da yer almaktadır. Panelin sol kısmında tespit edilen birim ve oranı görüntülenirken, sağ kısmında da birimin diğer birimlere TCMB günlük döviz kurları üzerinden dönüşümü sağlanmaktadır. Şekil 4.9'da gerçek zamanlı banknot tanıma işlemi yer almaktadır.



Şekil 4.9 Gerçek Zamanlı Banknot Tanıma

4.6 Performans Ölçütleri

Manuel olarak elde edilen 11.400 fotoğraf CreateML çatısı ile eğitilerek bir banknot tespit modeli geliştirilmiştir. Veri arttırma yöntemiyle kesme, döndürme ve bulanıklaştırma işlemleri uygulanarak fotoğraf sayısı arttırılmıştır. Geliştirilen model Türk Lirası, Amerikan Doları ve Euro para birimlerini tespit edebilmektedir. Öznitelik çıkarımı, eğitim kümesinden 7 saat 25 dakika ve test kümesinden 4 dakika sürede sağlanmış olup, 50 devirde eğitilmiştir. Bu bağlamda, % 91 oranında eğitim ve % 79 oranında doğrulama başarımı sağlanmıştır.

Sınıflanan veriler arasında başarım oranı en yüksek olan 2 USD olurken (174 adet), en düşük olan da 100 TRY (120 adet) olmuştur. Çizelge 4.3'te üç para birimine ait toplam 19 sınıfın eğitimi sonucunda elde edilen hata matrisi yer almaktadır.

Çizelge 4.3 Hata Matrisi

	5 ₺	10 ₺	20 ₺	50 ₺	100 ₺	200 ₺	5 €	10 €	20 €	50 €	100 €	200 €	1 $	2 $	5 $	10 $	20 $	50 $	100 $
5 ₺	125	7	14	9	6	0	9	2	2	0	1	0	0	0	1	1	0	3	0
10 ₺	2	138	13	7	2	6	3	3	1	2	2	1	0	0	0	0	0	0	0
20 ₺	9	12	134	4	4	5	3	1	0	3	1	0	0	1	0	0	0	2	1
50 ₺	9	5	2	144	5	2	2	5	2	3	1	0	0	0	0	0	0	0	0
100 ₺	9	12	8	3	120	5	7	3	7	0	1	0	0	1	1	1	0	2	0
200 ₺	5	7	4	4	7	142	2	2	2	0	0	1	0	0	4	0	0	0	0
5 €	1	1	4	2	0	0	138	8	9	7	6	0	0	0	0	0	2	2	0
10 €	1	2	1	2	3	3	10	133	9	7	5	2	0	1	1	0	0	0	0
20 €	2	3	0	0	7	6	5	10	135	2	7	3	0	0	0	0	0	0	0
50 €	0	2	3	1	2	2	2	9	2	144	4	8	0	0	1	0	0	0	0
100 €	2	0	4	1	1	2	7	1	13	5	123	10	0	0	2	0	0	3	6
200 €	0	3	0	0	1	0	3	0	6	4	6	156	0	0	0	0	1	0	0
1 $	3	0	0	0	0	0	0	0	1	0	0	0	154	2	14	0	0	5	1
2 $	0	0	1	1	0	0	0	0	0	0	2	0	0	174	0	2	0	0	0
5 $	2	1	2	0	1	1	1	1	1	1	5	0	6	2	149	2	0	5	0
10 $	0	0	1	0	0	0	1	0	0	0	2	0	0	5	4	144	16	4	3
20 $	0	0	0	0	1	0	0	0	0	0	1	0	1	3	1	14	157	2	0
50 $	1	2	4	0	4	6	1	0	1	0	1	0	4	0	19	2	2	130	3
100 $	1	0	0	0	0	0	1	0	0	0	3	3	2	0	4	4	0	0	162

Hata matrisindeki verilerden elde edilen sonuçlara göre performans ölçütlerinin kestirim oranları hesaplanmıştır. Bu bağlamda oluşturulan modelde doğruluk, kesinlik, hassaslık (yakalama), doğru negatif oranı, yanlış sınıflandırma oranı, yanlış pozitif oranı ve F1 skoru ölçütleri hesaplanmıştır. Çizelge 4.4'te hesaplanan ölçütlerin kestirim oranları verilmiştir.

Çizelge 4.4 Kestirim Oranları

Kestirim	**Oran**
Doğruluk	% 79,00
Kesinlik	% 79,72
Yakalama	% 79,00
Doğru Negatif Oranı	% 87,84
Yanlış Sınıflandırma Oranı	% 20,99
Yanlış Pozitif Oranı	% 12,15
F1 Skoru	% 79,36

5. BÖLÜM

SONUÇ

Tez çalışmasında 11.400 adet banknot fotoğrafının CreateML kütüphanesi kullanılarak eğitilmesi sağlanmış ve CoreML dosya formatına dönüştürülmüştür. (*Doviz.mlmodel*) Geliştirilen model iOS uygulamasına entegre edilmiştir. Elde edilen 11.400 fotoğrafa ek olarak veri arttırma yöntemiyle kesme, döndürme ve bulanıklaştırma işlemleri uygulanarak yapay bir şekilde fotoğraf sayısı arttırılmıştır. Veri kümesinin eğitilmesiyle elde edilen model uygulamaya eklenerek tüm banknotlar tanınır hâle getirilmiştir. Tanınan banknotlar, TCMB günlük döviz kurları kullanılarak farklı para birimlerine dönüştürülmektedir.

Tez çalışmasında geliştirilen modelde yaklaşık olarak %80 oranında sınıflandırma başarımı, %91 oranında eğitim ve %79 oranında doğrulama başarımı sağlanmıştır. 19 sınıftan en yüksek başarım oranına 2 USD ve en düşük başarım oranına da 100 TRY sahiptir. Hata matrisindeki verilerden elde edilen sonuçlara göre doğruluk %79, kesinlik %79,72, yakalama (hassaslık) %79, doğru negatif oranı %87,84, yanlış sınıflandırma oranı %20,99, yanlış pozitif oranı %12,15 ve F1 skoru %79,36 oranında hesaplanmıştır.

Bu çalışmadan tecrübe alınarak, bu proje para birimi türlerinin de arttırılmasıyla bozuk para denetimi ve sahte para denetimi gibi çalışmalarla sürdürülebilir.

KAYNAKLAR

Akpınar, H. (2014). *Data: Veri Madenciliği.* İstanbul: Papatya.

Artificial Intelligence. (2019, Mayıs 09). 06 13, 2019 tarihinde Encyclopedia Britannica: https://www.britannica.com/technology/artificial-intelligence adresinden alındı

Bagley, J. D. (1967). The Behavior of Adaptive Systems Which Employ Genetic and Correlative Algorithms. *Doktora Tezi*. Michigan Üniversitesi.

Cengil, E., & Çınar, A. (2016). Görüntü Sınıflandırma için Yeni Bir Yaklaşım: Evrişimsel Sinir Ağları. *European Journal of Technic, 6*(2), 96-103.

CoreML. (2017). Mart 15, 2019 tarihinde Apple Developer: https://developer.apple.com/documentation/coreml adresinden alındı

CreateML. (2018). Mart 15, 2019 tarihinde Apple Developer: https://developer.apple.com/documentation/createml adresinden alındı

Elman, J. L. (1990). Finding Structure in Time. *Cognitinve Science, 14*(2), 179-211.

Gürsakal, N. (2017). *Makine Öğrenmesi ve Derin Öğrenme.* Bursa: Dora.

Gers, F. A., & Schmidhuber, J. (2000). Recurrent Nets That Time and Count. *IEEE-INNS,ENNS International Conference on Neural Networks, 3*, 189-194.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation, 12*(10), 2451-2471.

Hebb, D. O. (1949). *The Organization of Behavior.* New York: Wiley & Sons.

Hinton, G., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science, 313*(5786), 504-507.

Hinton, G., Deng, L., Dong, Y., Dahl, G., Mohamed, A.-r., Jaitly, N., . . . Kingsbury, B. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine, 29*(6), 82-97.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* MIT.

ISO 4217 Currency Codes. (2019, Mayıs 17). Mayıs 28, 2019 tarihinde ISO: https://www.iso.org/iso-4217-currency-codes.html adresinden alındı

Khashman, A., Şekeroğlu, B., & Dimililer, K. (2006). Coin Identification Using Neural Networks. *SIP'06 Proceedings of the 5th WSEAS international conference on Signal processing*, (s. 88-92). İstanbul.

Kitagawa, R., Mochizuki, Y., Iizuka, S., Simo-Serra, E., Matsuki, H., Natori, N., & Ishikawa, H. (2017). Banknote Portrait Detection Using Convolutional Neural Network. *15th IAPR International Conference on Machine Vision Applications (MVA)*, (s. 410-413). Nagoya.

Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics, 43*(1), 59-69.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning. *IEEE Commun. Mag.*, (s. 41-46).

Minsky, M., & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry.* MIT.

Nastoulis, C., Leros, A., & Bardis, N. (2006). Banknote Recognition Based On Probabilistic Neural Network Models. *10th WSEAS International Conference on SYSTEMS*, (s. 802-805).

Omatu, S., Yoshioka, M., & Kosaka, T. (2007). Banknote Classification Using Neural Networks. *IEEE Conference on Emerging Technologies and Factory Automation (EFTA)*, (s. 413-417).

Panah, N., & Masoumi, H. (2017). Banknotes detected using Image Processing Techniques. *International Journal of Computer Science and Mobile Computing, 6*(5), 34-44.

Pham, T. D., Lee, D. E., & Park, K. R. (2017). Multi-National Banknote Classification Based on Visible-light Line Sensor and Convolutional Neural Network. *MDPI Sensors, 17*(7).

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological View, 65*(6), 386-408.

Silahtaroğlu, G. (2016). *Veri Madenciliği Kavram ve Algoritmaları.* İstanbul: Papatya.

Smolensky, P. (1986). Information Processing in Dynamical Systems: Foundations of Harmony Theory. D. E. Rumelhart, & J. L. McClelland içinde, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Cilt 1, s. 194-281). Cambridge: MIT.

Thorndike, E., & Woodworth, R. S. (1901). The Influence of Improvement in One Mental Function Upon the Efficiency of Other Functions. *Psychological Review, 8*(3), 247-261.

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind, 49*, 433-460.

van Veen, F., & Leijnen, S. (2019). *Neuural Network Zoo*. 2019 tarihinde Asimov Institute: http://www.asimovinstitute.org/neural-network-zoo/ adresinden alındı

Widrow, B., & Hoff, M. E. (1960). Adaptive Switching Circuits. *WESCON Convention Record : Western Electronic Show and Convention, 4*(4), 96-140.

Yapay Zekâ Terimler Sözlüğü. (2018). 12 14, 2018 tarihinde Deep Learning Türkiye: http://sozluk.deeplearningturkiye.com adresinden alındı

Yazılımın Yükselen Alt Katmanı: 4 Maddede Deep Learning (Derin Öğrenme) Nedir? (2018, Ekim 23). Aralık 11, 2018 tarihinde CEOtudent: https://ceotudent.com/deep-learning-derin-ogrenme-nedir adresinden alındı

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control, 8*(3), 338-353.

EK-1

SÖZLÜK

İngilizce	Türkçe
Accuracy	Doğruluk
Activation	Etkileşim
Artificial Intelligence	Yapay Zekâ
Artificial Neural Network	Yapay Sinir Ağı
Autoenconder	Otokodlayıcı
Backpropagation	Geri Yayılım
Bias Unit	Ek Birim
Classification	Sınıflandırma
Convolution	Evrişim
Convolutional Neural Network	Evrişimsel Sinir Ağı
Deep Auto-Encoder	Derin Oto-Kodlayıcı
Deep Belief Network	Derin İnanç Ağı
Deep Learning	Derin Öğrenme
Epoch	Devir
Feature Extraction	Öznitelik Çıkarımı
Feed Forward	İleri Yönlü
Gradient Descent	Meyilli Azalım
Graphical Processing Unit	Grafik İşlem Birimi
Input	Girdi
Long Short-Term Memory	Uzun Kısa Vadeli Hafıza
Misclassification	Yanlış Sınıflandırma

Overfit	Aşırı
Outcome	Sonuç, Çıktı
Perceptron	Algılayıcı
Precision	Kesinlik
Prevelance	Yaygınlık
Recall	Yakalama
Recurrent Neural Network	Tekrarlı Sinir Ağı
Rectified Linear Unit	Doğrultulmuş Doğrusal Birim
Restricted Boltzmann Machines	Sınırlı Boltzmann Makineleri
Support Vector Machine	Destek Vektör Makinesi
True Negative	Doğru Negatif
False Positive	Yanlış Pozitif
Weight	Ağırlık

EK-2

image.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os
import glob
from PIL import Image
from resizeimage import resizeimage

class Resize(object):
    liste = []

    def __init__(self):
        print("\tResize Class")

    def setImageList(self):
        for filename in os.listdir(os.getcwd()):
            imageExtension = [".jpg", ".jpeg", ".png", ".gif"]
            for ex in imageExtension:
                if filename.endswith(str(ex)):
                    self.liste.append(filename)

    def getImageList(self):
        self.setImageList()
        return self.liste

    def resize(self):
        list = self.getImageList()
        print("\nTotal: %d images\n" % len(list))
        for i, l in enumerate(list):
            with open(l, 'r+b') as f:
                with Image.open(f) as image:
                    cover = resizeimage.resize_cover(image, [300, 300])
                    cover.save('r-' + l, image.format)
            print("%d. --- %s  -- processing..." % (i + 1, l))
        print("Done!")

resim = Resize()
resim.resize()
```

ÖZGEÇMİŞ

Emre JILTA, 20 Mayıs 1992 tarihinde Kosova Cumhuriyeti'nin başkenti Priştine'de doğdu. İlköğrenimine eski Yugoslavya döneminde Vuk Karaciç okulunda başladı, ancak Kosova savaşı nedeniyle ailesi ile birlikte İstanbul'a göç ederek eğitimine Güngör Tekiner İlköğretim Okulunda devam etti. Savaşın bitmesiyle ülkesine döndükten sonra, eğitimine başladığı okulda (şimdiki adıyla Elena Cika) devam etti. Liseyi de yaşadığı şehirde bulunan Sami Frashëri Lisesi Fen Bilimleri dalında tamamladı. Lisans eğitimini Trakya Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Anabilim dalında tamamladı. Lisans eğitiminin ardından yüksek lisans eğitimine 2016 yılında Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim dalına başlamıştır. İngilizce, Arnavutça, Sırpça ve Almanca bilmektedir.

TEZ ÖĞRENCİSİNE AİT AKADEMİK MAKALELER

Jılta, E. (2017) Türkiye'de Geliştirilen GNU/Linux Dağıtımlarının İncelenmesi, *II. Lisansüstü Öğrenci Kongresi*, Edirne DOI: 10.13140/RG.2.2.31835.57121