T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

İŞ SÜREÇLERİNDE İNSAN GÖRÜSÜNÜ DERİN ÖĞRENME İLE DESTEKLEME

ALTUĞ YİĞİT

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Yrd.Doç.Dr. Cem TAŞKIN

EDİRNE-2017

T.Ü. Fen Bilimleri Enstitüsü onayı

Prof.Dr. Murat YURTCAN
Fen Bilimleri Enstitüsü Müdürü

Bu tezin Yüksek Lisans tezi olarak gerekli şartları sağladığını onaylarım.

Doç.Dr. M. Tolga SAKALLI
Anabilim Dalı Başkanı

Bu tez tarafımca (tarafımızca) okunmuş, kapsamı ve niteliği açısından bir Yüksek Lisans tezi olarak kabul edilmiştir.

Yrd.Doç.Dr. Cem TAŞKIN
Tez Danışmanı

Bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir Yüksek Lisans tezi olarak oy birliği ile kabul edilmiştir.

Jüri Üyeleri	İmza
Doç. Dr. Erdinç UZUN	
Yrd. Doç. Dr. Tarık YERLİKAYA	

Tarih: 27/07/2017

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİYÜKSEK LİSANS PROGRAMI
DOĞRULUK BEYANI

İlgili tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin kaynak gösterilerek ilgili tezde yer aldığını beyan ederim.

20/07/2017

Altuğ YİĞİT



Yüksek Lisans Tezi
İş Süreçlerinde İnsan Görüsünü Derin Öğrenme ile Destekleme
T.Ü Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Bilgisayarın dış dünyayı insan gibi algılamasını, nesneler arasındaki uzaysal ilişkileri yorumlamasını sağlamak için uzun yıllardır bilgisayar görüsü alanında çeşitli çalışmalar yürütülmektedir. Geleneksel bilgisayar görüsü ve makine öğrenmesi teknikleri el yazısı tanıma gibi karışık işlemlerde insan görüsünün elde ettiği başarı oranını yakalayamamaktadır. Bu çalışmalar genellikle nesnelerin önemli özelliklerinin seçilerek, bu özellikler üzerinden tespitler yapılmasına dayanmaktadır. Dış dünyadaki nesnelerin çok sayıda ve karmaşık olması tespit işlemini zorlaştırmaktadır. Günümüzde nesneleri ayırt etmede kullanılabilecek özellikleri verilerden kendisi öğrenen bir yaklaşım olan derin öğrenme kullanılarak başarılı bir nesne algılama sistemi geliştirilebilmektedir. Derin öğrenme insan beyni ilham alınarak geliştirilen bir makine öğrenmesi yaklaşımdır. Biyolojik sinir ağlarında reseptörler tarafından bilgiler toplanarak, retina ve görme korteksi arasındaki katmanlı yapıdan geçerler. Sadece katmanlardan geçen sinyaller algılanmaktadır. Derin sinir ağlarında da biyolojik sinir ağlarına benzer katmanlı bir mimari mevcuttur. Bu mimari sayesinde, sadece önemli özelliklere odaklanma sağlanarak, nesnelerin ayırt edilmesinde önemli bir avantaj sağlanmaktadır.

Teknolojinin gelişmesi ile birlikte insanların iş gücüne katkı sağlayacak makineler tasarlanmıştır. Ancak iş süreçlerinde kullanılan makinelerin yönetiminde insan müdahalesi ve yönetimi halen önem arz etmektedir. Özellikle insan görüsüne ihtiyaç duyulan iş süreçlerinde makineler insanların yönetiminde çalışırlar. Bilgisayar görüsünün bu iş süreçlerinde insan görüsüne yardımcı olarak kullanılması hata payını önemli ölçüde azaltmaktadır. Günümüzde otonom araçlar, sesli asistanlar, karar destek

sistemlerinin başarılı uygulamaları hızlıca yaygınlaşmakta ve bu yönde çok sayıda akademik çalışma yapılmaktadır.

Bu tezde insan görüsüne destek sağlamak amacıyla, tartılarak satılan ürünlerin yapay görü sistemi ile tespitini sağlayan bir derin öğrenme uygulaması geliştirilmiştir. Farklı modeller üzerinde 31 çeşit meyve ve sebze sınıfından oluşan görüntü verileri ile eğitim ve test işlemleri yapılmıştır. Eğitim ve test işlemleri sırasında kullanılan veri seti perakende satış otomasyonunun satış sırasında etiketlemiş olduğu sabit kameradan çekilmiş özgün görüntülerden oluşmaktadır. Derin öğrenme algoritmaları büyük boyutlardaki matrisler üzerinde yüksek işlemci gücü gerektiren süreçlerden oluştuğu için, uygulama yapay zekâ sistemlerinin geliştirilmesi için özel olarak tasarlanmış olan hazır bir platform üzerinde geliştirilmiştir. Platform üzerindeki merkezi işlemci birimi yerine 256 çekirdekten oluşan grafik işlemci birimi kullanılmıştır. Test işlemleri sonucunda %90' ın üzerinde başarı elde edilmiştir.

Yıl : 2017
Sayfa Sayısı : 97
Anahtar Kelimeler : Derin Öğrenme, Bilgisayar Görüsü, Obje Tanıma, Görüntü İşleme

Master’s Thesis

Supporting Human Brain with Deep Learning On Business Process

Trakya University Institute of Natural Sciences

Computer Engineering

ABSTRACT

Various studies have been carried out in the field of computer vision for a long time for making computer to perceive world as human and to interpret spatial relations between objects. Traditional computer vision and machine learning techniques fail to obtain the success rate of human vision in mixed processes such as handwriting recognition. These studies are usually based on the selection and determination of important properties of objects. Because of the complexity it is difficult to detect objects. Nowadays, using deep learning approach that learns from data can be used to distinguish objects. By using deep learning, a successful object detection system can be developed. Deep learning is a machine learning approach inspired by the human brain. In biological neural networks, information is collected by the receptors and passes through the layered structure between the retina and the visual cortex. Only the signals passed from the layers are perceived. Deep neural networks also have a layered architecture similar to biological neural networks. This architecture provides a significant advantage in distinguishing objects by focusing only on important features.

With the development of technology, the machines designed that can contribute people’s workspace. However, human intervention and management are still important in the management of machines used in business processes. Especially in the business processes in which human vision is needed, the machines work in people's management. The use of computer vision as an assistant to human vision during these business processes significantly reduces the amount of error. Today, successful applications of autonomous devices, voice assistants, decision support systems are rapidly spreading and there are many academic studies in this field.

In this thesis, in order to support human vision, a deeper learning application has been developed that enables artificial vision systems to determine the products sold by weighing. Training and test processes was done with image data of 31 different fruits and vegetables on different models. During the training and testing processes the original images captured with a fixed camera that are labeled by a point of sale automation are used. The application is developed on an embedded system that is specially designed for artificial intelligence and machine learning applications. Because the deep learning algorithms consist of processes that require high processing power in large dimensions. 256 core graphics processor unit was used instead of a processor. In tests, over 90% success is achieved.

Year : 2017
Number of Pages : 97
Keywords : Deep Learning, Computer Vision, Object Recognition, Image Processing

ÖNSÖZ

Bu tezde iş süreçlerindeki insan hatalarının en aza indirgenmesini amaçlayan, insan görüsünü desteklemek amacı ile bir derin öğrenme sistemi geliştirilmiştir.

Tez çalışması Trakya Üniversitesi Bilimsel Araştırma Projeleri (TÜBAP) Birim' i tarafından 2017/95 numaralı proje ile desteklenmiştir. TÜBAP komisyonuna desteklerinden dolayı teşekkür ederim. Ayrıca tez çalışması sırasında kullanılan verileri temin eden Sayın Erdinç PULLUK (Bereket TİCARET)' a teşekkürlerimi bir borç bilirim.

Tez danışmanım Yrd. Doç. Dr. Sayın Cem TAŞKIN(Trakya Üniversitesi)' a tez konusunu belirlerken isteklerim doğrultusunda bana yol gösterici olduğu için ve zaman içindeki olumlu gelişmeleri takdir ederek beni yüreklendirip çalışma boyunca esirgemediği desteğinden dolayı sonsuz teşekkürlerimi sunarım.

Çalışmaya ayırdığım zamanı anlayışla karşılayıp bana destek veren, bana olan güvenini hiçbir zaman esirgemeyen sevgili Ebru YILDIRIM' a çok teşekkür ederim. Hayatım boyunca maddi, manevi desteklerini esirgemeyip, bu yolda sürekli yanımda olan aileme de sonsuz teşekkür ederim.

Altuğ YİĞİT

ÖNSÖZ v
KISALTMALAR ix
ŞEKİLLER LİSTESİ x
TABLOLAR LİSTESİ xii
BÖLÜM 1 1
GİRİŞ 1
1.1. Tezin Amacı 2
1.2. İş Süreçlerinde İnsan Görüsünü Derin Öğrenme ile Destekleme 3
BÖLÜM 2 6
LİTERATÜR ARAŞTIRMASI 6
BÖLÜM 3 16
MATERYALLER VE METODLAR 16
3.1. İnsan Beyni ve Görüsü 16
3.2. Yapay Sinir Ağları 19
3.2.1. Tekli Nöron Modeli 19
3.2.2. Aktivasyon Fonksiyonları 20
3.2.2.1. ReLU 21
3.2.2.2. Sigmoid 22
3.2.2.3. Tanh 22
3.2.3. İleri Beslemeli Ağlar 23
3.2.3.1. Tek Katmanlı Algılayıcılar 23
3.2.3.2. Çok Katmanlı Algılayıcılar 24
3.2.4. Geri Beslemeli Sinir Ağları 25
3.3. Derin Öğrenme 26
3.3.1. Derin Öğrenme ile Görüntü Sınıflandırma 27
3.3.2. Konvolüzasyonel Sinir Ağları 28
3.3.2.1. Özellik Çıkarım Aşaması 29
3.3.2.1.1. Konvolüzasyon Katmanı 29
3.3.2.1.2. Havuzlama Katmanı 32

3.3.2.2. Sınıflandırma Aşaması 34
3.3.2.2.1. Softmax Fonksiyonu 34
3.3.3. Öğrenme Süreci 35
3.3.3.1. Denetimli Öğrenme 35
3.3.3.2. Denetimsiz Öğrenme 36
3.3.3.3. Takviyeli Öğrenme 36
3.3.4. Veri Setlerinin Seçimi 37
3.3.5. Maliyet Fonksiyonları 38
3.3.5.1. Ortalama Hata Karesi (Mean Squared Error) 38
3.3.5.2. Ortalama Mutlak Hata (Mean Absolute Error) 39
3.3.5.3. Karekök Ortalama (Root Mean Square) 39
3.3.5.4. Çapraz Entropi(Cross Entropy) 40
3.3.6. Eğim Azalımı (Gradient Descent) 41
3.3.7. Momentum 42
3.3.8. Stokastik Eğim Azalımı (Stochastic Gradient Descent) 43
3.3.9. Geri Yayılım (Back propagation) 44
3.3.10. Regularizayon Teknikleri 48
3.3.10.1. L1 Tekniği 48
3.3.10.2. L2 Tekniği 49
3.3.10.3. Dropout 50
3.3.11. Derin Öğrenme Kütüphaneleri 51
3.3.11.1. TensorFlow 51
3.3.11.2. Theano 53
3.3.11.3. Torch 54
3.3.11.4. Caffe 56
3.3.11.5. Keras 57

BÖLÜM 4 58
DENEYLER VE SONUÇLAR 58
4.1. Geliştirilen Derin Öğrenme Sistemi 58
4.2. Verilerin Hazırlanması 59
4.3. Veri Setleri 61
4.4. Performans Ölçütleri 65
4.5. Deneysel Sonuçlar 67
4.5.1. Test 1 68
4.5.2. Test 2 74
4.5.3. Test 3 83
4.5.4. Test 4 91
BÖLÜM 5 97
SONUÇ 97
KAYNAKLAR 113
ÖZGEÇMİŞ 118
TEZ ÖĞRENCİSİNE AİT TEZ İLE İLGİLİ BİLİMSEL FAALİYETLER 119

KISALTMALAR

BVLC **:**Berkeley Görü ve Öğrenme Merkezi

CE **:**Çapraz Entropi

CUDA **:**Birleştirilmiş Aygıt Hesaplama Mimarisi

CuDNN **:**Cuda Derin Sinir Ağı Kütüphanesi(Cuda Deep Neural Network Library)

ÇKA **:**Çok Katmanlı Algılayıcılar

GİB **:**Grafik İşlemci Birimleri(GPU)

KO **:**Karekök Ortalama

KSA **:**Konvolüzasyonel Sinir Ağları

MİB **:**Merkezi İşlem Birimi(CPU)

OHK **:**Ortalama Hata Karesi

OMH **:**Ortalama Mutlak Hata

PLU **:**Ürün Kodu(Price Look-Up Code)

POE **:**Ethernet Üzerinden Besleme(Power Over Ethernet)

RAM **:**Rastgele Erişimli Bellek

ReLU : **:**İyileştirilmiş Doğrusal Birim

RPN **:**Bölge Öneri Ağları

R-CNN **:**Bölge Bazlı Konvolüzasyonel Sinir Ağları

SVM **:**Destek Vektör Makineleri

VGG **:**Görsel Geometri Gurubu(Visual Geometry Group)

YSA **:**Yapay Sinir Ağları

ŞEKİLLER LİSTESİ

Şekil 1.1. Tartılı Ürün Satış Etiketi Örneği 3
Şekil 2.1 Yann Lecun KSA 6
Şekil 2.2 ImageNet yarışmasındaki en iyi hata oranları 7
Şekil 2.3 AlexNet KSA Mimarisi 8
Şekil 2.4 Görü Modülü 9
Şekil 2.5 Artık ağ hesaplaması 10
Şekil 2.6 Sahnelerin etiketlenmesi 11
Şekil 2.7 R-CNN 12
Şekil 2.8 Hızlı R-CNN 13
Şekil 2.9 Daha hızlı R-CNN 14
Şekil 2.10 El hareketlerini taklit edebilen robotik entegre sistem 15
Şekil 3.1 Biyolojik sinir sistemi 17
Şekil 3.2 Biyolojik sinir ağı modeli 18
Şekil 3.3 Sinir hücrelerinin hesaplaması 20
Şekil 3.4 ReLU aktivasyon fonksiyonunun grafiği 21
Şekil 3.5 Sigmoid aktivasyon fonksiyonunun grafiği 22
Şekil 3.6 Tanh aktivasyon fonksiyonunun grafiği 22
Şekil 3.7 Tek katmanlı algılayıcı modeli 24
Şekil 3.8 Çok katmanlı algılayıcı modeli 24
Şekil 3.9 Geri beslemeli sinir ağı modeli 26
Şekil 3.10 KSA aşamaları 28
Şekil 3.11 3x3 boyutunda filtre 30
Şekil 3.12 Bir adımlı konvolüzasyon işlemi 31
Şekil 3.13 İki adımlı, sıfır boşluk konvolüzasyon işlemi 32
Şekil 3.14 Ortalama havuzlama ve maksimum havuzlama işlemleri 33
Şekil 3.15 Doyma noktasına ulaşım 42
Şekil 3.16 Evrensel ve yerel minimum noktaları 43
Şekil 3.17 Basit zincir kuralı 45
Şekil 3.18 Sinir ağı üzerinde geri yayılım 46
Şekil 3.19 Gerekli olmayan ağırlıkların kullanılmaması 49

Şekil 3.20 YSA modeli ve dropout tekniği uygulanmış hali 50
Şekil 3.21 Üç farklı kütüphane ile yapılan test sonucu 54
Şekil 3.22 Yığın boyutuna göre yapılan test sonucu 54
Şekil 3.23 CPU ve GPU üzerinde ileri ve geri hesabın performansları 56
Şekil 3.24 Keras' ın işlev yapısı 57
Şekil 4.1 Sistemin çalışma modeli 59
Şekil 4.2 Kaynak görüntüden veri arttırımı 61
Şekil 4.3 Terazi üzerindeki ürünlerin görüntüleri 62
Şekil 4.4 Karmaşıklık matrisi 65
Şekil 4.5 Model 1' in doğruluk değişim grafiği 72
Şekil 4.6 Model 1' in maliyet fonksiyonu değişim grafiği 72
Şekil 4.7 Model 2' nin doğruluk değişim grafiği 73
Şekil 4.8 Model 2' nin maliyet fonksiyonu değişim grafiği 73
Şekil 4.9 Model 3' ün doğruluk grafiği 80
Şekil 4.10 Model 3' ün maliyet fonksiyonu değişim grafiği 80
Şekil 4.11 Model 4' ün doğruluk grafiği 81
Şekil 4.12 Model 4' ün maliyet fonksiyonu değişim grafiği 81
Şekil 4.13 Model 5' in doğruluk grafiği 82
Şekil 4.14 Model 5' in maliyet fonksiyonu değişim grafiği 82
Şekil 4.15 Model 6' nın doğruluk grafiği 88
Şekil 4.16 Model 6' nın maliyet fonksiyonu değişim grafiği 88
Şekil 4.17 Model 7' nin doğruluk grafiği 89
Şekil 4.18 Model 7' nin maliyet fonksiyonu değişim grafiği 89
Şekil 4.19 Model 8' in doğruluk grafiği 90
Şekil 4.20 Model 8' in maliyet fonksiyonu değişim grafiği 90
Şekil 4.21 Model 9' un doğruluk grafiği 94
Şekil 4.22 Model 9' un maliyet fonksiyonu değişim grafiği 94
Şekil 4.23 Model 10' un doğruluk grafiği 95
Şekil 4.24 Model 10' un maliyet fonksiyonu değişim grafiği 95

TABLOLAR LİSTESİ

Tablo 3.1TenserFlow kütüphanesinin performansının karşılaştırılması 52
Tablo 4.1 İlk veri setine ait sınıflar 63
Tablo 4.2 İkinci veri setine ait sınıflar 64
Tablo 4.3 Kullanılan veri setleri ve özellikleri 65
Tablo 4.4 Testler ve özellikleri 67
Tablo 4.5 Test 1'de kullanılan modellerinin özellikleri 68
Tablo 4.6 Model 1' in yapısı 69
Tablo 4.7 Model 2' nin yapısı 70
Tablo 4.8 Test 1' deki modellerin performansı 70
Tablo 4.9 Test 2'de kullanılan modellerinin özellikleri 74
Tablo 4.10 Model 3' ün yapısı 76
Tablo 4.11 Model 4' ün yapısı 77
Tablo 4.12 Model 5' in yapısı 78
Tablo 4.13 Test 2' deki modellerin performansı 79
Tablo 4.14 Test 3' te kullanılan modellerin özellikleri 83
Tablo 4.15 Model 6' nın yapısı 84
Tablo 4.16 Model 7' nin yapısı 85
Tablo 4.17 Model 8' in yapısı 86
Tablo 4.18 Test 3' teki modellerin performansı 87
Tablo 4.19 Test 4'te kullanılan modellerin özellikleri 91
Tablo 4.20 Model 9' un yapısı 92
Tablo 4.21 Model 10' un yapısı 93
Tablo 4.22 Test 4'teki modellerin performansı 93

BÖLÜM 1

GİRİŞ

Teknolojinin gelişmesiyle birlikte insan makine etkileşimi karmaşık bir hal almıştır. İnsan görüsü nesneleri tanınma işlemini beyninde ağırlıklı olarak semantik uzayda gerçekleşmektedir. Ayrıca nesnelerin hatları, kesimleri, şekilleri, sınırları gibi semantik anlamlı özelliklerini çıkarabilmektedir [1].

Nesne tanıma işlemi insanlar için kolay bir işlem olsa da dikkatin dağılması, dalgınlık, duygu değişimleri gibi bazı durumlarda yanlışlar yapılabilmektedir. Yapılan yanlışlar insanın yönetiminde çalışan makineler yardımıyla yürütülen iş süreçlerinin de yanlış işlemesine neden olmaktadır. İş süreçlerinde makinelerin olabildiğince az müdahale ile kullanılması, süreçlerin otomatik hale getirilmesi ile insan hataları daha aza indirgenebilir.

Bilgisayar görüsü görüntüleri piksel matrisleri olarak algılamaktadır. Nesnelerin bilgisayarlar tarafından tanınması, sahnelerin algılanması üzerine uzun yıllardır çalışmalar yapılmaktadır. Bu çalışmalardan en popüler olanı derin öğrenmedir. Derin öğrenme makine öğrenmesinin yeni bir çalışma alanıdır. İnsan beyninden ilham alınarak geliştirilen yapay sinir ağlarını temel almaktadır. Yapay sinir ağlarındaki gibi katmanlı yapıya sahiptirler, çoklu katmanlardan oluşurlar. Derin öğrenme sayesinde bilgisayar görüsü alanı[2] dışında konuşma tanıma[3], doğal dil işleme[4] gibi çeşitli uygulama alanlarında başarı sağlanmıştır.

Derin öğrenme birçok çevrimiçi ve mobil servislerde de yaygın olarak kullanılmaktadır. Ses tanıma uygulamalarından Apple tarafından geliştirilen Siri, Google Assistant, Amazon Alexa, Microsoft Cortana, görüntü sınıflandırma uygulamalarından Google Photo ve Facebook örnek olarak verilebilir [5].

Derin sinir ağlarının eğitimi basit sinir ağı modellerine göre çok daha zordur. Bunun nedeni eğitimin çok sayıda veri ile gerçekleşmesi ve büyük modelin milyonlarca parametreye sahip olmasıdır. Eğitim süreci güçlü işlemci gücü gerektirmektedir.

Sürecin daha kısa sürmesini sağlamak amacıyla matris işlemlerini hızlı bir şekilde yapabilen grafik işlemci birimleri (GİB) kullanılmaktadır. Bu tezde makine (akıllı terazi) ile insan görüsüne yardımcı olarak kullanılması planlanan bir derin öğrenme sistemi geliştirilmiştir. İnsan görüsü ile gerçekleştirilebilen meyve, sebze tartımlarındaki iş sürecinin otomatik hale getirilmesi amaçlanmıştır. Bu amaçla yapılan çalışmada konvolüzasyonel yapay sinir ağları (KSA) kullanılarak derin öğrenme modeli geliştirilmiş ve özgün veri seti ile grafik işlemci birimi üzerinde eğitilmiştir. Farkı modeller ve veri setleri üzerinde iyileştirmeler yapılarak %90'nın üzerinde başarı sağlanmıştır.

1.1. Tezin Amacı

Günümüzde perakende satış noktalarında tartılarak satışı gerçekleştirilen ürünlerin satışı 2 farklı yöntem ile yapılmaktadır. İlk yöntemde tartım ürünün bulunduğu reyonda bir işletme personeli tarafından ürün kodu (PLU – Price Look-Up Code) girilerek gerçekleştirilmektedir. Satılan ürünün miktarı, ürünün açıklaması, son tüketim tarihi, satış fiyatı gibi bilgilerin yer aldığı barkod etiketi terazi tarafından termal bir yazıcı ile yazdırılmaktadır. Şekil 1.1' de tartılı bir ürüne ait barkod yazıcılı teraziden yazdırılmış olan bir etiket görülmektedir. Yazdırılan etiket ürünlerin konulduğu poşetin üzerine yapıştırılmaktadır. Müşteri ödeme noktasında geldiğinde barkod okuyucuya sahip yazarkasa, poşetin üzerinde yer alan barkod bilgisini yorumlayarak, satışın gerçekleştirilmesini sağlar. Burada satış elemanının PLU kodunu yanlış girmesi durumunda, maddi kayıp söz konusu olmaktadır. PLU kodları genellikle 3 haneden oluşup, satış elemanının barkod etiketini yazdırabilmesi için bu kodu numerik klavyeden doğru olarak girmesi zorunludur. Tartılarak satışı gerçekleştirilen ürünlerin etiketlerinin tamamının bir personel tarafından oluşturulması, o personelin sürekli olarak terazinin yanından çalışması anlamına gelmektedir. Bu açıdan bakıldığında, ciddi oranda iş gücü kaybı yaşanmaktadır.



Şekil 1.1. Tartılı Ürün Satış Etiketi Örneği

İkinci yöntemde ise tartılarak satışı yapılan ürünün miktarın tespiti, müşteri ödeme noktasında yer alan yazarkasa ile iletişim halinde bulunan bir terazi ile yapılmaktadır. Ödeme noktasında çalışan personel PLU kodunu, ürünü terazinin üzerine koyduktan sonra yazarak satış işlemini gerçekleştirir. Her iki yöntemde de PLU kodunun yanlış girilmesi halinde stok kontrolünü sırasında hatalar ile karşılaşılmaktadır.

İş süreçlerinde insan faktörü süreçlerin hatasız ve hızlı yürümesinin önünde önemli bir engeldir. Yukarıda anlatılan iki yöntemde de otomasyon sistemlerinin doğru olarak işlemesini sağlamak için ürüne ait PLU kodunun doğru olarak girilmesi gerekmektedir. Ürün çeşitlerinin çok olduğu bir satış operasyonunda yanlış girişlerin ve zaman kaybının önlenmesine destek olacak veya süreçlerin tamamen otomatik hale getirilmesini sağlayacak bir sistemin geliştirilmesi ile zaman ve maliyet kayıplarının önlemesi sağlanabilir.

Bu tez çalışmasında, insan görüsüne destek olabilecek bir derin öğrenme sistemi geliştirilmiştir. Bu sistem girdi olarak aldığı ürün görüntülerini sınıflandırma yaparak ürünün PLU kodunu tahmin etme yeteneğine sahiptir. Derin öğrenme ile son yıllarda özellikle görüntü sınıflandırma alanında önemli başarılar elde edildiği için bu çalışmada derin öğrenmenin konvolüzasyonel sinir ağları yöntemi tercih edilmiştir.

1.2. İş Süreçlerinde İnsan Görüsünü Derin Öğrenme ile Destekleme

İş süreçlerinde özellikle insan görüsüne bağlı, uzman bilgisi gerektiren süreçler bu konuda uzman kişi/kişiler tarafından yönetilebilmektedir. Bu sistemlerde dikkatsizlik, yorgunluk gibi nedenlerle yapılan hatalar iş süreçlerinde aksamalara veya süreçlerin yanlış olarak işlemesine neden olmaktadır.

İnsan bilgi işleme modeli üç ana bölümden oluşmaktadır. İlk bölüm hissetme ve algılama, ikinci bölüm işleme ve karar verme üçüncü bölüm ise eylem gerçekleştirmeden oluşmaktadır. İnsan davranışları gerçekleştirilirken hissetme ve algılama işlemi ile bilgi toplama işlemi yapıldıktan sonra elde edilen bilgi işlenir ve hangi eylem yapılacağına karar verilir, son olarak da eylem gerçekleştirilir. Hatalar bu bölümlerin herhangi birinde ortaya çıkabilir. Hatayı etkili olarak azaltma yöntemi de hatanın hangi bölümde ortaya çıktığına göre farklılık göstermektedir. Algılama bölümündeki bir hatanın nedeni ışıklandırma, fazla gürültü, kalitesiz yazdırılmış materyaller, karar verme sürecinde yorgunluk, bilgi eksikliği ve zaman stresi, eylem gerçekleştirmede yetersiz araç veya ekipman dizaynı, eksik prosedürler, dikkat dağıtıcı ortam, soğuk veya sıcak çalışma alanı gibi faktörler etkili olabilir [6]. Derin öğrenme gibi bir zeki sistemin yardımcı olarak kullanılması bilgi işleme modelindeki bölümlerde hataya neden olan faktörleri hafifleterek hataların en aza indirgenmesi ve sürecin kontrollü bir şekilde gerçekleştirilmesini sağlayabilir.

Son on yıldır, kurumsal kaynak yönetimi araçları şirketlerin verimliliklerini ve hizmet kalitesini arttırmak için çözümler sağlamıştır. Fakat günümüz dünya endüstrisinin rekabetçi yapısı şirketlerin rakipleri arasında konumlarını koruyabilmesi için daha yeni teknolojileri uygulamaya zorlamaktadır [7]. Bu rekabetçi iş ortamında şirketler kaliteyi ve üretkenliği attırma konusunda zorluklarla karşı karşıyadır. Birçok üretim sistemi akıllı araçların eksikliği nedeniyle bu zorlukların üstesinden gelmeye hazır değildir. Endüstri 4.0 ile birlikte üretim ve hizmet yenilikçiliğine dayanan bir dönüşüm başlamıştır [8].

Endüstri 4.0 döneminden önceki dönemlerde otomasyonun kullanılmaya başlanması, üretimde bir devrim yaratmıştır. Otomasyonda karmaşık işlerin makinelerin tekrar edebileceği gibi basit, adım adım işleyen talimatlara bölünmesi ile üretime katkı sağlanmıştır. Böyle bir sistemde farklı üretim süreçlerinde sistem montajı ve sistemi kontrol etme ihtiyacı doğmuştur. Bu işler genellikle çalışanlar tarafından gerçekleştirilmiştir, fakat bu tür eksiklikler bu işlerin bilgisayar görüsü ile yapılmasını daha cazip hale getirmiştir [9].

İnsanlar çoğu zaman işleri makinelerden daha iyi gerçekleştirmelerine rağmen çalışma hızları makinelerden düşüktür ve kısa zaman içerisinde kolayca yorulabilirler.

Ayrıca iş süreçlerinde çalışacak olan kişilerin çalışacağı konuda uzman olması zorunludur.

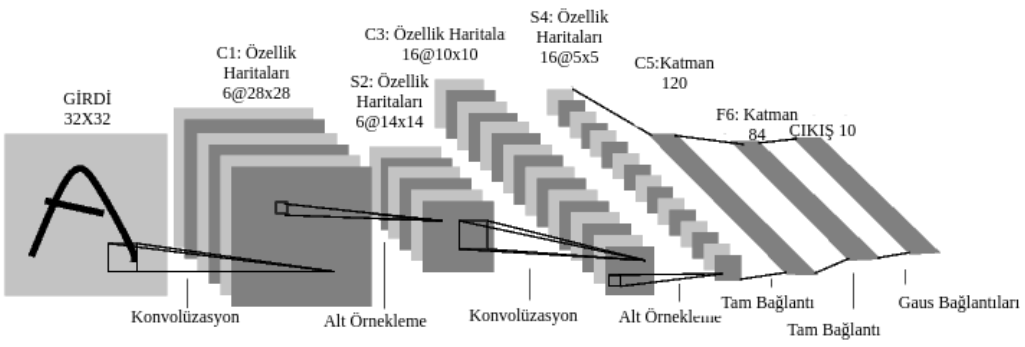
Buna karşın, endüstride çalışacak uzman bir kişinin bulunması ve uzun süre çalışmasının sağlanması oldukça zordur. Kişilerin yeteneklerinin ve tecrübelerinin geliştirilmesi için uzun zaman ve eğitim gerekmektedir. Çok iyi eğitimli bir uzman olsa da kontrol işleminin sıkıcı, zor ve sürekli tekrar ettiği durumlarda çalışan verimsiz olabilmektedir. Su altı kontrolü, nükleer endüstri, kimyasal endüstri gibi bazı ortamlarda kontrol işlemi zor ve tehlikeli olabilmektedir. Bilgisayar görüsü bu ve benzeri ortamlarda insan görüsü yerine kullanılabilir. Üretim sistemlerinde bilgisayar görüsü uygulamaları daha çok kalite kontrol işleminde kullanılmaktadır. Üretilen ürünlerin yapay görü sistemi ile taranarak kusurlu parçaların otomatik olarak çıkarılması örnek olarak verilebilir [10].

BÖLÜM 2

LİTERATÜR ARAŞTIRMASI

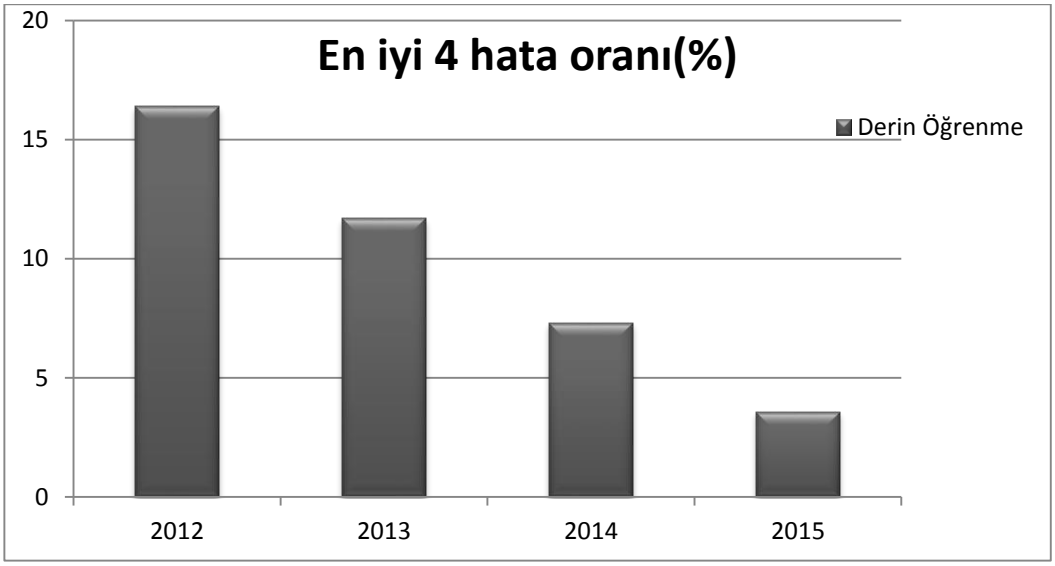
Son yıllarda bilgisayar görüsü uygulamalarında derin öğrenme kullanımı artmıştır. Nesne tanıma, görüntü sınıflandırma gibi işlemlerde yaygın olarak konvolüzasyonel sinir ağları kullanılmaktadır ve bu alanda birçok başarı elde edilmiştir [2, 11, 12].

Yann Lecun tarafından el yazısı ile yazılan rakamları ayırt edebilen bir model oluşturulmuştur. Bu model LeNet-5 olarak adlandırılmaktadır. Çok katmanlı sinir ağları türü olan standart bir konvolüzasyonel sinir ağı modelidir. Model Şekil 2.1' de görüldüğü gibi giriş katmanı hariç yedi katmandan oluşmaktadır. 32x32 boyutunda görüntü piksellerini girdi olarak almaktadır. MNIST olarak adlandırılan el yazısı ile yazılmış rakamların bulunduğu 60.000 eğitim ve 10.000 test veri seti üzerinde yapılan çalışmada doğrusal sınıflandırma algoritmalarına göre oldukça iyi bir sonuç olan %0.95 hata oranı elde edilmiştir. 1998' de yapılan bu çalışma günümüzdeki konvolüzasyonel sinir ağlarının temelini oluşturmaktadır [13].



Şekil 2.1 Yann Lecun KSA

2010 yılından günümüze geleneksel olarak her yıl yapılan bilgisayar görüsü algoritmalarının yarıştığı ImageNet adlı Geniş Çaplı Görsel Tanıma Yarışması'nda sınıflandırılmak üzere yüzlerce nesne kategorisi, eğitim ve test verisi olarak milyonlarca görüntü kullanılmaktadır. Bu yarışmada katılımcılar nesneleri kategorilere göre sınıflandırma ve tanıma işlemini en iyi şekilde gerçekleştirecek, insan görüsünün doğruluğunu yakalamayı amaçlayan algoritmalar geliştirmektedir. Görüntü içerisinde nesnelerin saptanması şeklinde başlayan yarışma sonrasında görüntü içerisinde bulunan nesnelerden her sınıfa ait birer örnek işaretlenmesi şeklinde devam etmiştir. Ayrıca 2013 yılından itibaren görüntü içerisindeki nesnelerin her birinin işaretlenmesi istenmektedir [14].

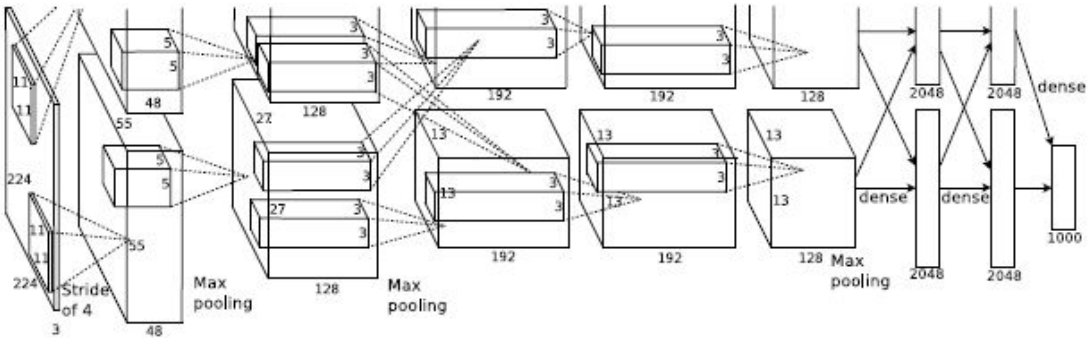


Şekil 2.2 ImageNet yarışmasındaki en iyi hata oranları

Giderek zorlaşan görevler eklense de derin öğrenme modellerinin kullanılmasıyla birlikte hata oranları bu zorluğa ters orantılı olarak azalmaktadır. Şekil 2.2' de de görüldüğü gibi ImageNet yarışmasında 2012 yılı ve sonrasında geniş çaplı konvolüzasyonel sinir ağı modellerinin kullanılmasına bağlı olarak klasik makine öğrenmesi yöntemlere göre daha başarılı sonuçlar elde edilmiştir.

SuperVision isimli takım 2012 yılında AlexNet olarak adlandırılan derin konvolüzasyonel sinir ağı modeli ile yarışmaya katılmıştır. Şekil 2.3' te görülen bu model hem görüntü sınıflandırma hem de lokalizasyon görevlerinde birinci olmuştur.

Model beş konvolüzasyonel katman ve üç tam bağlı katmandan oluşmaktadır. Oldukça geniş olan model 650.000 adet nörona sahiptir. Model 1000 farklı kategoride 1.2 milyon görüntüyü sınıflandıracak şekilde iki GİB üzerinde eğitilmiştir. Eğitim 5-6 gün kadar sürmüştür. Modelin eğitim verilerini ezberlemesini engellemek için dropout tekniği kullanılmıştır. İşlenecek veri olarak, görüntüyü oluşturan 224x224 boyutunda 3 adet kırmızı, yeşil ve mavi renk kanallarına ait bilgileri içeren matrisler kullanılmıştır. Testler sonucunda 2012 yılına kadar elde edilemeyen %16.4 hata oranı elde edilmiştir [2, 14].



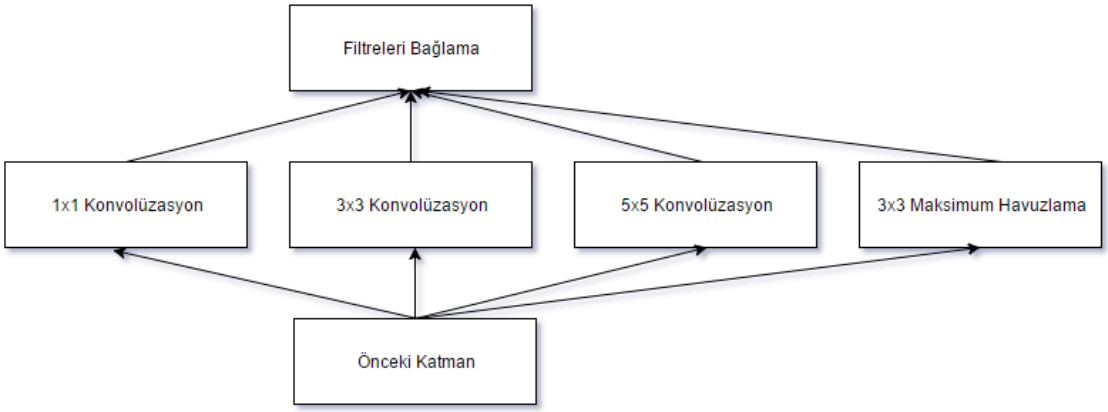
Şekil 2.3 AlexNet KSA Mimarisi

AlexNet modeline çok benzeyen Clarifai takımı tarafından geliştirilen ZfNet modeli 2013 yılında ImageNet görüntü sınıflandırma alanında birincilik elde etmiştir. Bu model AlexNet modelini geliştirerek oluşturulmuştur. Bu iki modelde giriş katmanları aynı olmasına rağmen, ZfNet' te ilk konvolüzasyonel katmanında 11x11 boyutunda 4 adımlı filtreler yerine 7x7 boyutunda 2 adımlı filtreler kullanılmıştır. Diğer konvolüzasyonel katmanların boyutu büyütülmüştür. Son katmalar ise iki katına çıkarılmış ve boyutu 4096 olarak belirlenmiştir. Eğitim tek GİB üzerinde yaklaşık 12 gün sürmüştür. Test sonuçlarında %11.7 hata oranı elde edilmiştir [11, 14].

Google tarafından tasarlanan 22 katmandan oluşan GoogLeNet modeli 2014 yılında ImageNet görüntü sınıflandırma alanında birincilik elde etmiştir. Diğer modellerden farklı olarak görü modülü (Inception module) geliştirilmiştir. Bu modül sayesinde modelin hesaplama maliyeti sabit kalırken derinleşmesi sağlanabilmektedir. Bu model eski yıllarda tasarlanan diğer modellere kıyasla daha başarılı ve diğer modellerden daha az parametreye sahiptir. Örneğin AlexNet' ten yaklaşık %5 daha az

hata oranına sahip iken 12 kat daha az parametreye sahiptir. Bu modelin tasarımı mobil ve gömülü sistemlerde kullanabilecek şekilde optimize edilmiştir.

Bu modelde konvolüzasyon ve maksimum havuzlama (pooling) işlemleri yapıldıktan sonra görü modülü işletilmektedir. Şekil 2.4' te görü modül yer almaktadır. Bu modülde eş zamanlı olarak bir önceki katmandan gelen girdi üzerinde 1x1, 3x3, 5x5 konvolüzasyon filtresi ve 3x3 maksimum havuzlama filtresi aynı anda işletilmektedir. Elde edilen sonuç değerleri ise bir filtre bağlama adımında birleştirilmektedir. Bu sayede aynı anda çok katmanlı özellik çıkarımı yapılabilmekte ve parametre sayısında azalma sağlanabilmektedir [12].



Şekil 2.4 Görü Modülü

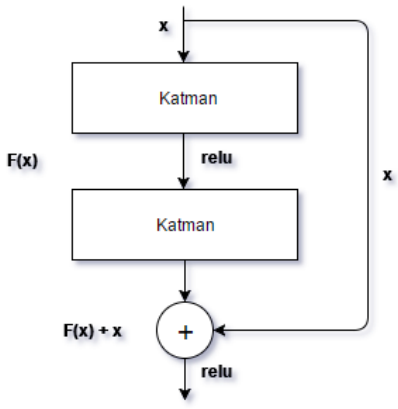
GoogLeNet modelinde standart konvolüzasyonel sinir ağlarının son katmanında bulunan Tam Bağlı Katmanlar yerine 7x7 boyutunda Ortalama Havuzlama işlemi kullanılmıştır. Bunun sonucunda doğruluk oranı yaklaşık %0.6 oranında artmıştır. Test sonuçlarında yaklaşık %6.7 hata oranı elde edilmiştir. Yapılan testlerde insan görüsü hata oranın yaklaşık %5 olduğu belirlenmiştir. GoogLeNet modeli insan hata oranına oldukça yaklaşmıştır ve insan görüsü ile rekabet edecek duruma gelmiştir [12, 14].

VGG (Visual Geometry Group – Oxford University) adlı ekip tarafından geliştirilen KSA modeli, ImageNet 2014 yılında lokalizasyon alanında birincilik ve sınıflandırma alanında ikincilik elde etmiştir. Bu ekip KSA başarısının artmasını için ne kadar derinleşmesi gerektiğini araştırmıştır. Konvolüzasyon katmanında 3x3 boyutunda çok küçük filtreler kullanılarak 16-19 arasında konvolüzasyon katmanı eklenerek ağın

derinleşmesi sağlanmıştır. Sınıflandırma testleri sonucunda %7.3 hata oranı elde edilmiştir [15].

Microsoft Araştırmacıları tarafından geliştirilen ResNet modeli 2015 yılında ImageNet yarışmasında görüntü sınıflandırma, nesne tanıma ve lokalizasyon alanlarında birincilik elde etmiştir. GoogLeNet modelinden daha başarılı bir modeldir. Test sonuçlarında %3.57 hata oranı elde edilerek yaklaşık %1.43 farkla insan görüsünü geçmeyi başarmıştır. Diğer modellere kıyasla oldukça derin bir yapıya sahip olan bu model 152 katman içermektedir. Derin ağların eğitim işlemini kolaylaştıracak artık(residual) ağlar kullanılmıştır. Yapılan deneyler artık ağların daha kolay optimize edildiğini ve derinlikleri arttığında doğruluklarının arttıklarını göstermektedir. Artık ağlar, standart konvolüzasyonel sinir ağı modellerine kısa yol bağlantılarının eklenmesiyle oluşturulan ağ modelleridir.

Şekil 2.5'te artık öğrenmenin çalışma modeli yer almaktadır. Şekildeki F(x) fonksiyonu iki ağırlık katmanının çıktısını temsil etmektedir. İşlemler sonucunda elde edilen F(x) fonksiyonu katmanların giriş değeri olan x ile toplanır ve bu değer aktivasyon fonksiyonun girdisi olarak kullanılır. İşlemin yapılabilmesi için katmanların girdi ve çıktı boyutu birbiri ile aynı olmalıdır. Aynı olmadığı takdirde değişken sıfır değerleri ile doldurularak boyutların eşitlenmesi sağlanabilir.



Şekil 2.5 Artık ağ hesaplaması

18 ve 34 katmanlı standart konvolüzasyonel sinir ağları ile ImageNet 2012 veri seti kullanarak yapılan testler sonucunda katman sayısı fazla olmasına rağmen 34

katmanlı sinir ağının eğitim ve test hata oranları 18 katmanlı ağdan daha fazla olarak ölçülmüştür. Aynı deney artık ağlar ile yapıldığında 34 katmanlı ağın hata oranlarında düşüş olduğu ve normal ağlara göre daha çabuk doyma noktasına ulaştığı gözlemlenmiştir [16].

Literatürde geçen konvolüzasyonel sinir ağlarının tamamı Yann Lecun rakamları tanımak için tasarladığı LeNet modele benzemeleriyle birlikte daha fazla nöron sayısı ve katman içermektedir. Bunun nedeni dünyadaki nesnelerin yapısının rakamlarda olduğu gibi basit özellikler içermemesidir. Modellerin boyutları arttıkça daha fazla nesne özelliklerini temsil edebilmektedir. ImageNet yarışma sonuçlarından da görüldüğü gibi modelin derinleştirilmesi başarı oranını arttırmaktadır.

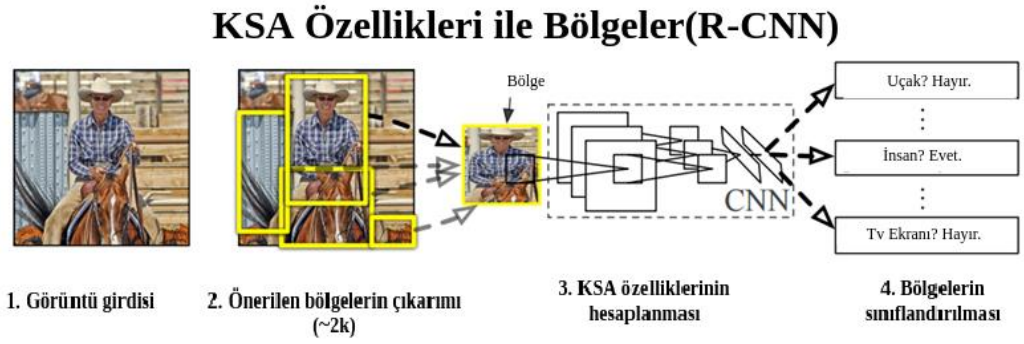
Stanford Üniversitesi'nden Andrej Karpathy ve Li Fei-Fei görüntülerin içerisinden hem nesneyi hem de sahneyi ayırt edebilen bir çalışma yapmışlardır. Yapılan çalışmada nesnelerin ayırt edilebilmesi için Konvolüzasyonel Sinir Ağları ve sahnelerin açıklanması, cümlelerin oluşturulabilmesi için Tekrarlı Sinir Ağları kullanılmış ve modeller birleştirilip Multimodal(Çok Modlu) Tekrarlı Sinir Ağları yapısı tanımlanmıştır. Şekil 2.6' da görülen örnekler elde edilen modelin görüntü içerisinden anlamlı sonuçlar ürettiğini göstermektedir [17].



Şekil 2.6 Sahnelerin etiketlenmesi

Berkeley Üniversitesi araştırmacıları tarafından bölge bazlı konvolüzasyonel sinir ağları(R-CNN) olarak adlandırılan bir yaklaşım geliştirilmiştir. Bu yaklaşımda girdi olarak alınan görüntü üzerinden kullanılan bir algoritmanın önerdiği yaklaşık iki bin farklı bölge belirlenir. Elde edilen her bölge derin konvolüzasyonel sinir ağına girdi

olarak verilir ve ağın özellik çıkarım yeteneği kullanılarak her bölge için otomatik özellik çıkarımı yapılır. Elde edilen özellikler doğrusal destek vektör makineleri yöntemi ile sınıflandırma işlemine sokulur ve görüntü üzerinde hangi nesnelerin bulunduğu tespit edilir [18]. Şekil 2.7' de R-CNN çalışma modeli görülmektedir.



Şekil 2.7 R-CNN

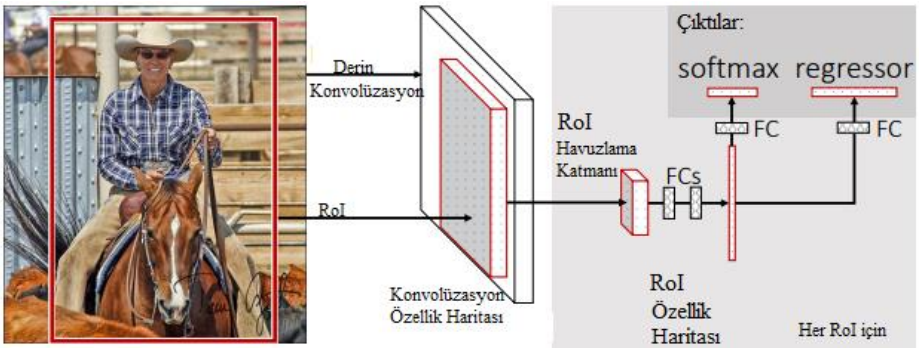
R-CNN yönteminde görüntü içerisindeki nesnelerin yerlerinin belirlenmesi için Seçici Arama Algoritması kullanılabilmektedir. Seçici Arama Algoritması kullanılarak olası tüm nesnelerin yerleri belirlenmeye çalışılır. Görüntü içerisinde her pikseli arama işleminin hesaplama maliyeti fazladır. Bu yüzden arama uzayı belirli bir bölgede olacak şekilde ölçeklenmelidir. Nesne yerlerinin saptanmasında bir takım zorluklar mevcuttur. Örnek olarak görüntü içerisinde aynı nesneler farklı ölçeklerde bulunabilir veya nesneler birbirinin parçası olabileceği için tek bir teknik ile bölgelerin belirlenmesi doğru olmayabilir. Bir görüntü içerisinden nesne olarak arabanın saptanması istendiğinde sadece renk bazlı bir ayrım yapılmaya çalışıldığında arabanın tekerlekleri arabadan farklı bir nesne olarak algılanabilmektedir. Nesneler belirli bir hiyerarşik yapıda tanımlanabilir. Seçici Arama Algoritması segmentasyon işlemlerinde kullanılan aşağıdan yukarıya hiyerarşik gruplama işlemlerini içermektedir. Küçük bölgelerden büyük bölgelere doğru hiyerarşi oluşturulmaktadır. Seçici Arama Algoritması üç temel adımdan oluşur.

1) Birçok başlangıç bölgesi oluşturulur.

2) Bölgeler arasında komşuluklar belirlenir.

3) Aç Gözlü (Greedy) Algoritması kullanılır. Aç Gözlü algoritmasında ilk olarak en çok benzeyen iki adet başlangıç bölgesi seçilir. İki başlangıç bölgesi birleştirilerek tek bir bölge oluşturulur. Bu adım tek bir bölge oluşana kadar devam etmektedir [19].

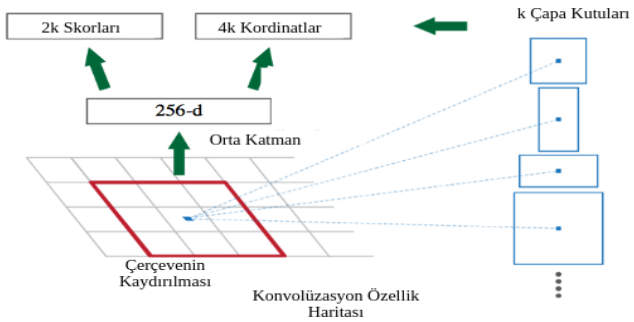
Bölge bazlı konvolüzasyonel sinir ağlarının yavaş olması nedeniyle Ross Girshick tarafından hızlı sürümü olan Fast R-CNN yöntemi geliştirilmiştir. R-CNN çok aşamalı işlemlerden oluştuğundan dolayı hesaplama maliyeti ve zaman kaybı oldukça fazladır. Görüntü üzerinden nesne tespiti işlemi 47 saniye sürmektedir. Bu yöntemde nesne tespiti süresi 0.3 saniyeye indirilmiştir. Bu yöntemde çok aşamalı işlemlerin tek bir aşamaya indirilmesi önerilmektedir. Görüntüye belirlenen parametrelere göre birkaç konvolüzasyon ve havuzlama işlemi uygulanır. Elde edilen özellik haritalarına havuzlama(pooling) katmanı uygulanarak sabit boyutlu özellik matrisleri oluşturulur ve matrislerin her biri tam bağlı katmanlara iletilir. Sistemin iki çıkışı bulunur. Çıkışlardan birinde softmax fonksiyonu ile olasılıklar hesaplanır. Diğer çıkışta ise işaretlenecek bölgenin yerini belirten regresyon değeri elde edilir. Bu yöntem R-CNN'e göre daha başarılıdır [20].



Şekil 2.8 Hızlı R-CNN

Faster R-CNN olarak adlandırılan bir yöntemde Fast R-CNN yönteminin hızlanması için konvolüzasyonel katmanların çıkışına Bölge Öneri Ağlarının(RPN) eklenmesi önerilmektedir. Bölge Öneri Ağları özellik haritalarını girdi olarak alır ve nesne skorları ile birlikte olarak bir dizi bölge önerisi verir. İşlem gerçekleştirilirken özellik haritasından belirli bir boyutta çerçeve belirlenir çerçevenin ortasına çapa(anchor) yerleştirilir. Ölçek ve en boy oranına göre bölgeler belirlenir ve iki tam

bağlı katmandan geçirilerek sonuçlar üretilir [21]. Şekil 2.9' da Faster R-CNN yer almaktadır.



Şekil 2.9 Daha hızlı R-CNN

2014 yılında yemek sınıflandırma yapan bir yaklaşımda ise 2 konvolüzasyon katmanı bulunan KSA ile yaklaşık %74 başarı elde edilmiştir. Aynı veri seti üzerinde Destek Vektör Makineleri(SVM) yöntemi kullanılarak %50-60 oranında başarı elde edildiği görülmüştür [22]. 2016 yılında Yuzhen Lu tarafından geliştirilen bu yaklaşıma benzer bir yaklaşımda ise SVM ile 3 konvolüzasyon katmanı bulunan KSA karşılaştırılmış ve benzer oranlar elde edilmiştir [23]. 2015 yılında Keiji Yanai ve Yoshiyuki Kawano 6 konvolüzasyon katmanlı daha derin bir KSA ile geliştirilen başka bir yemek sınıflandırma yaklaşımında ise yaklaşık %79 başarı elde edilmiştir [24].

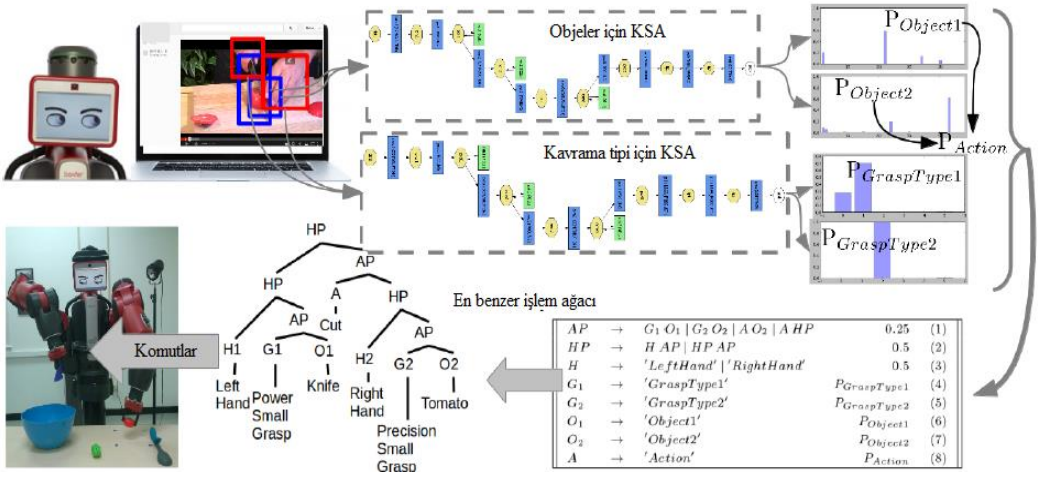
2016 yılında Sidney Üniversitesi Robotik Alanı Araştırmacıları tarafından meyve bahçelerinde meyvelerin miktarının belirlenmesi işleminde KSA ve ÇKA yöntemleri kullanarak meyvenin olup olmadığını ayırt eden ve segmentasyon yapan yaklaşımlar geliştirilmiş ve sonuçlar karşılaştırılmıştır. Meyve bahçesinde yapılan tahmin işlemlerinde ÇKA ile yaklaşık %75, KSA ile yaklaşık %82 başarı elde edilmiştir [25].

Japon çiftçiler ürünlerini renk, boyut, şekil gibi bazı özelliklere göre sınıflandırmaları daha yüksek fiyatla satılmalarına olanak sağlamaktadır. Çiftçilerin ürün sınıflandırılmasını öğrenmesi aylar sürmekte ve bu işlem için her gün belirli bir zaman harcamaktadır. Gömülü sistem tasarımcısı Makoto Koike, zaman maliyetini düşürmek için derin öğrenme ile insan görüsüne yardımcı olacak bir sistem

geliştirmiştir. Gömülü sistemden gönderilen görseller Google Cloud üzerinde işlendikten sonra elde edilen tahmin değerine göre sınıflandırma yapılmaktadır [26].

2016 yılında yapılan bir çalışma da ise çileklerin endüstriyel olarak sınıflandırmasına imkân sağlayan kenar tanıma algoritmasıyla birlikte YSA kullanılmıştır. Bu çalışmada sınıflandırma için derin öğrenme yöntemi yerine iki katmanlı yapay sinir ağı kullanılmıştır. Eğitim işleminden sonra ağın başarılı bir şekilde renk, şekil ve anormallikleri saptayabildiği görülmüştür [27].

2015 yılında Yezhou Yang ve arkadaşları tarafından yapılan çalışmada internet üzerinden elde edilen videolardan insanların el hareketlerini öğrenip taklit edebilen bir robotik sistem geliştiriştir. İki adet KSA bulundan sistemde ilk ağ nesnelerin tanınması için kullanılırken ikinci ağ ise el hareketlerinin taklit edilmesi amacıyla kullanılmaktadır. Ağların çıktı değerlerini kullanarak robotik sisteme belirli bir hiyerarşik düzende komutlar gönderilmektedir. El hareketleri ve nesne tanıma ile birlikte genel olarak robot sistemin videolardaki sahneyi taklit edebilmesinde %79 başarı sağlanmıştır [28]. Şekil 2.10‘ da entegre sistemin modeli görülmektedir.



Şekil 2.10 El hareketlerini taklit edebilen robotik entegre sistem

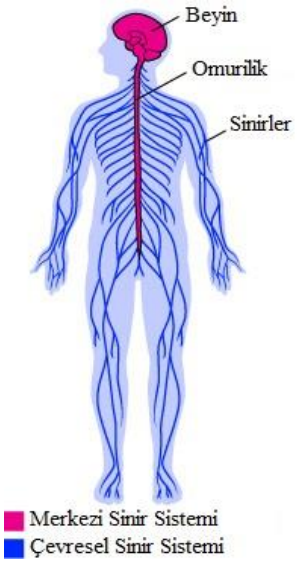
BÖLÜM 3

MATERYALLER VE METODLAR

3.1. İnsan Beyni ve Görüsü

YSA' nın çalışma mantığının anlaşılmasında insan beyninin yapısının bilinmesi oldukça önem taşımaktadır. İnsan beyni, beş duyu organından bilgiler toplamaktadır. Aynı zamanda veya farklı zamanda gerçekleşen bilgi toplama işleminden sonra elde edilen bilgilerden anlamlar çıkartılır ve bilgiler bellekte saklanabilir. Beyin, düşüncelerimizi, hafızamızı, konuşmamızı, el ve kol hareketlerimizi ve vücuttaki birçok organın fonksiyonlarını kontrol etmektedir. Ayrıca kalp ve solunum hızını düzenleyerek stres altında nasıl cevap vereceğimize karar vermektedir.

İnsan sinir sistemi merkezi ve çevresel sistemler olarak ikiye ayrılmaktadır. Merkezi sinir sistemi beyin ve omurilikten oluşmaktadır. Şekil 3.1' de görüldüğü gibi çevresel sinir sistemi ise omurilikten dallanan omurilik sinirleri ve beyinden gelen kafa sinirlerinden oluşmakta, nefes alma, çiğneme, kalp atış hızı gibi hayati fonksiyonları kontrol eden otonom sinir sistemini içermektedir [29].



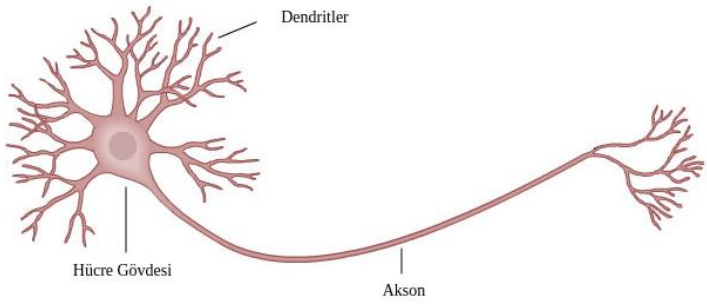
Şekil 3.1 Biyolojik sinir sistemi

Beyin nöron adı verilen sinir hücreleri ve gliya adı verilen hücrelerinden oluşmaktadır. Üç ana tip nöron mevcuttur. Bunlar; duyusal nöronlar, motor nöronlar ve internöronlardır.

Duyusal nöronlar, iç ve dış çevrenin çeşitli özelliklerini algılamak ve bunlara tepki vermek için reseptörlere bağlıdır. Işık, ses, mekanik ve kimyasal uyaranlardaki değişikliklere duyarlı olan reseptörler, görme, işitme, dokunma, koku ve tat duyularının elde edilmesinde yardımcı olurlar. Uyarıcılar belirli bir yoğunluğu aştığında, doku hasarına neden olabilirler ve nosiseptörler olarak adlandırılan özel bir reseptör grubunu etkinleştirirler. Bunlar hem koruyucu refleksleri, hem de ağrı hissini yaratırlar.

Motor nöronlar kas aktivitelerini kontrol etmektedir, konuşma da dâhil olmak üzere her türlü davranıştan sorumludurlar. İnternöronlar, duyusal ve motor nöronların arasında yer alır ve insan beyninde en çok sayıda bulunan nöronlardır [30].

Nöronlar dendrit, akson ve hücre gövdeden oluşmaktadır. Birbirleri ile iletişim içindedirler. Aktarılacak olan bilgileri elektriksel ve kimyasal sinyaller vasıtası ile iletirler. Birbirlerine akson adı verilen bağlar bağlıdırlar. Bağlantı noktalarına ise sinaps adı verilir. Akson üzerinden farklı bir nörondan gelen bilgileri dendritler yardımı ile alırlar ve hücre gövdede bilgilerin sinyal halinde iletilip iletilmeyeceğine karar verilir.



Şekil 3.2 Biyolojik sinir ağı modeli

Tüm biyolojik beyinlerde birbirine sıkıca bağlanmış nöronlardan oluşan sinir ağları bulunmaktadır. İnsan beyni yaklaşık yüz milyar sinir hücresi, 3.2 milyon km bağlantı içermektedir. Ağırlığı sadece 1.5 kg ve harcadığı enerji 10 watt elektrik kadardır. Silikon çipler kullanarak beyne benzer bir yapı inşa etmeyi denersek bu yapı yaklaşık 10 megawatt yani bir şehrin elektriğini karşılamaya yetecek kadar elektrik tüketecektir. Beynin nasıl bu kadar etkili ve ekonomik olarak çalıştığını keşfetmek ve aynı çalışma prensibini kullanarak beyin benzeri makinelerin inşa edilmesi zor bir iştir [30].

İnsan beyninin yaklaşık dörtte biri görsel işleme ile ilgilenmektedir. Bu oran diğer duyulara kıyasla daha fazladır. Dünyayı görmemizi ve algılamamızı sağlayan görme duyusu en hassas ve karmaşık duyularımızdan biridir. Görme işlemi, korneanın içinden geçen ışıkla ve odaklanmayı ayarlayan lens ile başlar. Kornea ve mercek, retinadaki fotoreseptör tabakası üzerinde görsel dünyanın net bir görüntüsünü üretmeye yardımcı olur. Fotoreseptörler ışığı absorbe eder ve gözün arka tarafını kaplayan nöronlara elektrik sinyalleri gönderir. Kamerada olduğu gibi retinadaki görüntü de ters çevrilir. Göz bebeğinin büyüklüğü gözün içerisine ne kadar ışık gireceğini ayarlar. Ve göz bebeği iris tarafından kontrol edilir. Lensin şekli irisin hemen arkasındaki kaslar tarafından değiştirilir. Böylece retina üzerinde yakın veya uzak nesnelere odaklanılabilir. Her insanın gözünde yaklaşık 125 milyon fotoreseptör bulunmaktadır. Fotoreseptörler ışığı elektrik sinyallerine çeviren uzmanlaşmış nöronlardır. Çubuklar ve koniler olmak üzere iki formdan oluşurlar. Çubuklar loş ışığa karşı duyarlıdır ve renk taşımazlar. Koniler parlak ışık altında çalışır ve detaylı, siyah-beyaz ve renkli görmeden

sorumludur. İnsan gözü her biri farklı renk aralığına duyarlı üç tür koni içermektedir. Sadece bu üç tip koniyi kullanarak binlerce rengi ayırt edebiliriz.

Retina yapay sinir ağları ile modellenebilecek yapıda üç aşamalı nöronlar içermektedir. İlk katmanda çubuk ve koni tabakaları sinyallerini orta katmana gönderir. Orta katmandaki bu sinyaller üçüncü katmana, oradan da görsel kortekse ulaştırılır. Üçüncü katmanın aksonları optik siniri oluşturan ganglion hücrelerinden oluşur. Orta ve üçüncü katmandaki her hücre önceki katmandaki birçok hücreden girdi alır. Ancak girdi sayısı retinanın içinde değişiklik gösterir. Bu katmanlı yapı çok ince detayları çözmemize izin verir. Görsel bir alana görsel nöronun girişini sağlayan bölge, alıcı alan olarak adlandırılır [31].

3.2. Yapay Sinir Ağları

Yapay Sinir Ağları, karmaşık işlem yeteneğine sahip biyolojik beyin yapısını ilham alan hesaplama modelleridir [32]. Biyolojik beyin yapısına benzer bir şekilde birbirine bağlı birçok yapay nörondan oluşurlar. YSA nöron gruplarından oluşan katmanlara sahiptir. YSA' lar tek bir katmandan oluşabileceği gibi birçok katmandan oluşabilen YSA' da geliştirilebilir.

Günümüzdeki bilgisayarlar birçok algoritmik problemi kolayca çözebilmektedir. Fakat insanlar için kolay olan görme, konuşma, duyma gibi davranışların bilgisayarlar tarafından çözülebilmesi oldukça zordur. YSA bilgisayarların insan gibi davranmasını sağlamak için bir zemin hazırlamaktadır. Verilerden öğrenerek deneyim kazanırlar ve bir problemi çözmek amacıyla elde edilen deneyimleri kullanırlar.

YSA' nın tarihi 1940'ların başlarına dayanmaktadır. 1943 yılında Warren McCulloch ve Walter Pitts tarafından nöronlara dayalı nörolojik ağların eşik anahtarlarının modelleri ortaya konulmuştur. Ayrıca basit bir ağ modellerinin bile neredeyse tüm mantık veya aritmetik fonksiyonları hesaplayabildiği gösterilmiştir [33].

3.2.1. Tekli Nöron Modeli

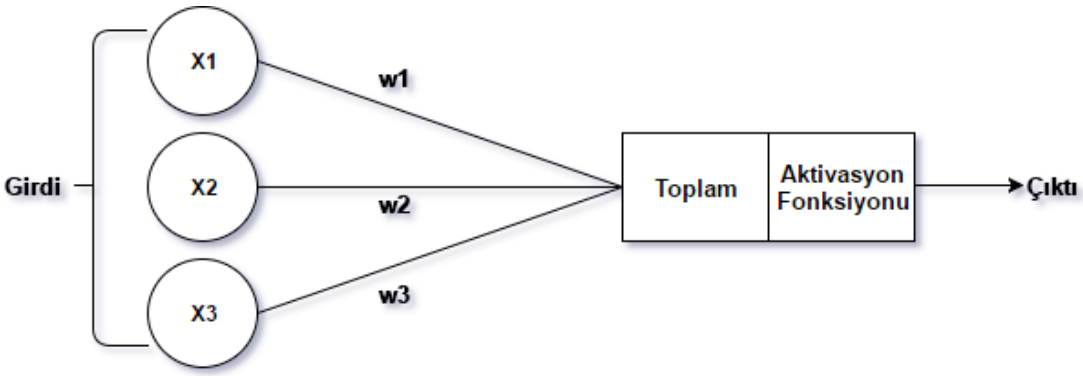
Nöron olarak adlandırılan sinir hücresi en basit hesaplama birimidir. Giriş değerlerini alarak bir çıktı değeri üretir. Şekil 3.3'teki gibi girdi değerleri belirli ağırlık

değerleri ile çarpılır ve mevcut ise nöronun bias değeri de eklenerek toplam elde edilir. Herhangi bir katmanındaki tüm nöronların sayısı n olmak üzere toplam fonksiyonu

$$\mathbf{T(x)} = \sum_{\mathbf{i=1}}^{\mathbf{3}} \mathbf{w_i x_i} + \mathbf{b} \qquad (3.1)$$

$$\mathbf{T(x)} = \mathbf{w_1 x_1} + \mathbf{w_2 x_2} + \mathbf{w_3 x_3} + \mathbf{b} \qquad (3.2)$$

şeklinde tanımlanabilir. Elde edilen toplam değeri aktivasyon fonksiyonuna girdi olarak verilir ve nöron sonuç değerine göre aktive olur.



Şekil 3.3 Sinir hücrelerinin hesaplaması

YSA' da ağırlık ve bias değerleri öğrenilebilen parametrelerdir. Her nöronda bulunan bias değerlerinin kullanım amacı kullanılan aktivasyon fonksiyonun eksen üzerinde hareket etmesini sağlamaktır. Ağırlıklar aktivasyon fonksiyonun grafiğine etki etse bile bias değerleri kullanılarak ağın daha farklı durumları ifade etmesi sağlanabilmektedir.

3.2.2. Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları YSA' da Doğrusal Olmayan Fonksiyon olarak ta adlandırılmaktadır. Adından da anlaşılacağı gibi genellikle aktivasyon fonksiyonu olarak doğrusal fonksiyon seçilmediği için nöronlara doğrusal olmayan bir yapı sağlar

ve daha karmaşık fonksiyonları temsil edebilecek yetenek kazandırırlar. Aktivasyon fonksiyonu seçiminde doğrusal fonksiyonların kullanımı tercih edilirse ağ, doğrusal olarak çözülebilen problemleri ifade edebilmektedir.

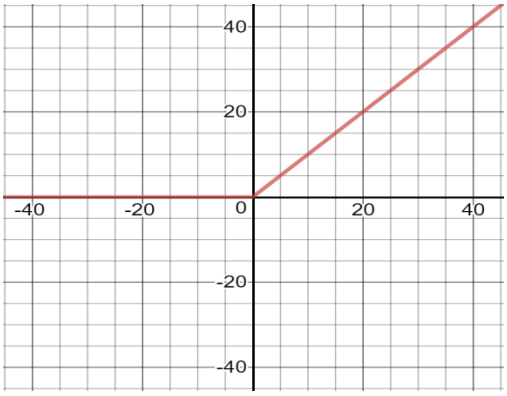
Nöronlar aktivasyon fonksiyonlarına göre aktive olurlar. Seçilen aktivasyon fonksiyona göre belirli bir eşik değeri altında kalan değerlere sahip nöronlar aktive edilmez. Aktive edilemeyen nöronlar kullanılmaz.

Çeşitli aktivasyon fonksiyonları mevcuttur. Bu fonksiyonlardan ReLU(Rectified Linear Unit) modern derin modellerde yaygın olarak kullanılmaktadır. Ayrıca Sigmoid ve Tanh fonksiyonları da aktivasyon fonksiyonu olarak tercih edilmektedir.

3.2.2.1. ReLU

ReLU' nun açılımı İyileştirilmiş Doğrusal Birim' dir. Denklem 3.3'de tanımlandığı gibi fonksiyona girdi olarak verilen negatif değerleri sıfır, sıfırdan büyük değerleri de kendi değerlerine atayarak eşikleme işlemi yapılmaktadır. Bu fonksiyonu diğerlerin ayıran en önemli özellik aktivasyon işlemini hızlı bir şekilde gerçekleştirilmesidir. Fonksiyonun grafiği Şekil 3.4'te yer almaktadır.

$$\varphi(x) = \max(0, x) \qquad (3.3)$$

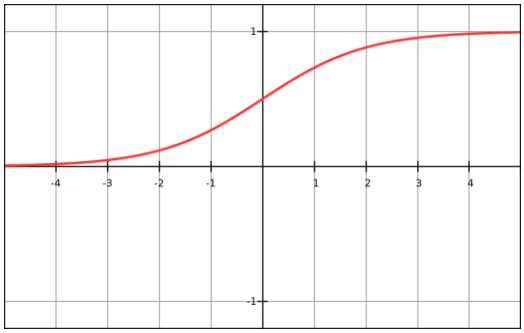


Şekil 3.4 ReLU aktivasyon fonksiyonunun grafiği

3.2.2.2. Sigmoid

Sigmoid fonksiyonu girdi olarak aldığı değerleri 0 ve 1 arasında ifade etmektedir. Fonksiyondan Şekil 3.5'te gösterildiği gibi eğri şeklinde bir grafik elde edilmektedir.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad (3.4)$$

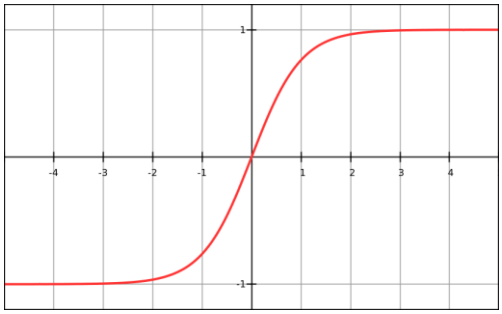


Şekil 3.5 Sigmoid aktivasyon fonksiyonunun grafiği

3.2.2.3. Tanh

Tanh fonksiyonu Sigmoid fonksiyonu benzemektedir. Farklı olarak girdi değerlerini -1 ve 1 arasında ifade etmektedir.

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \qquad (3.5)$$



Şekil 3.6 Tanh aktivasyon fonksiyonunun grafiği

3.2.3. İleri Beslemeli Ağlar

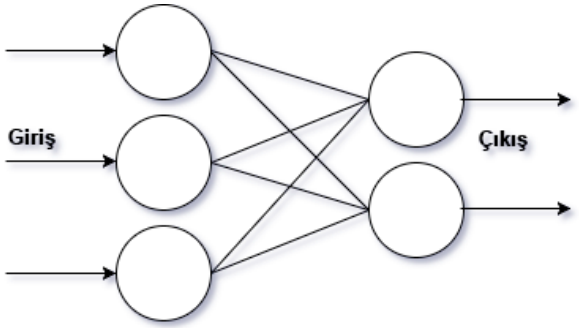
İleri Beslemeli Ağlar farklı sayıda nöron gruplarını içeren katmanlardan oluşmaktadır ve ileri katmanlara doğru bilgi akışı sağlanmaktadır. Ağlarda bulunan nöronlar düğüm olarak ta adlandırılmaktadır. Katmanlar da bulunan düğümler birbirine ağırlık değerlerini içeren bağlantılar ile bağlıdır. Aynı katmandaki düğümler birbiri ile bağlantılı değildir, çünkü bu şekilde ağ üzerinde döngü oluşturulup ileriye doğru iletim sağlanamayabilir. Giriş katmanı, gizli katman ve çıkış katmanı olmak üzere üç farklı katman tipi vardır.

Giriş katmanından belirlenen düğüm sayısına göre veri girişi sağlanır. Gizli katman giriş ve çıkış katmanlarının arasında bulunan bir katmandır. Bu katman isteğe bağlı olarak eklenir. Özellikle ağ derinleştirilmek istendiğinde gizli katmanların sayısı arttırılabilmektedir. Girdi değerlerinin hesaplamalar yapılarak ağ üzerinde ileri yönde iletilmesiyle ileri doğru bilgi akışı sağlanarak değerler çıkış katmanına ulaştırılır.

İleri beslemeli sinir ağları içerisinde yer alan fonksiyonların birlikte nasıl çalıştıklarını yönlü düz çizgeler (directed acyclic graph) açıklayabilir. Bir zincir içerisinde bağlı olan üç adet fonksiyon tanımlarken $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$ şeklinde bir gösterim sağlanabilir. Bu zincir yapısı YSA' da ortak olarak kullanılan yapıdır. $f^{(1)}$ ilk katman olarak tanımlandığında sırasıyla diğer fonksiyonlarda diğer katmanları ifade edecektir. Zincirin toplam uzunluğu, modelin derinliğini vermektedir [34].

3.2.3.1. Tek Katmanlı Algılayıcılar

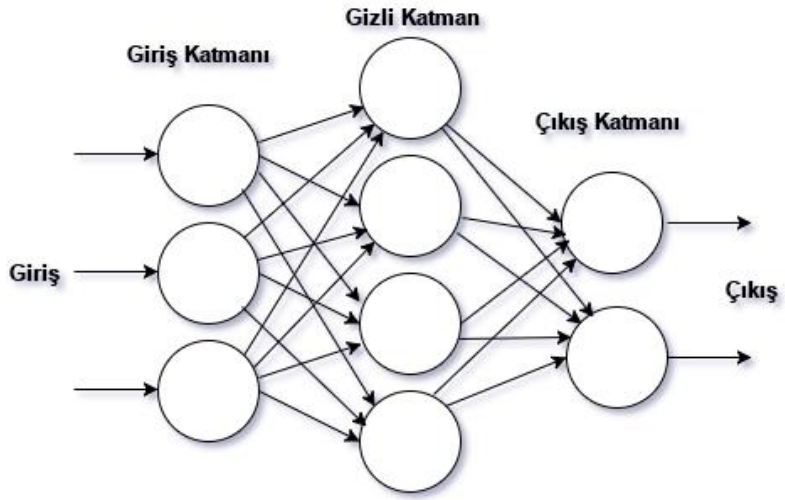
Tek katmanlı algılayıcılar gizli katmanı bulunmayan ileri beslemeli sinir ağı türüdür. Giriş katmanı çıkış katmanına bağlıdır. Doğrusal fonksiyonları temsil edebildikleri için karmaşık olmayan doğrusal problemlerin çözümünde kullanılırlar. Şekil 3.7' de tek katmanlı algılayıcı modeli görülmektedir.



Şekil 3.7 Tek katmanlı algılayıcı modeli

3.2.3.2. Çok Katmanlı Algılayıcılar

Çok katmanlı algılayıcılar en az bir gizli katmanı bulunan ileri beslemeli sinir ağı türüdür. Gizli katmanlar verilerdeki özelliklerin belirlenmesini sağlamaktadır. Gizli katman sayısında bulunan sinir hücreleri ne kadar fazla ise veriden o kadar fazla özellik çıkarımı yapılabilmektedir. Gizli katmanlar sayesinde doğrusal olmayan fonksiyonlar temsil edilebilmektedir. Çok katmanlı algılayıcılar tek katmanlı algılayıcıların çözebildiği problemlere kıyasla daha karmaşık olan problemlerin çözümünde kullanılırlar. Şekil 3.8' de 1 adet gizli katmana sahip çok katmanlı algılayıcı modeli yer almaktadır.



Şekil 3.8 Çok katmanlı algılayıcı modeli

Sigmoid aktivasyon fonksiyonu kullanılarak Şekil 3.8 'deki gibi ileri beslemeli bir ağda gizli katmanda bulunan bir nöronun hesaplaması aşağıdaki gibi yapılmaktadır.

$$\boldsymbol{\sigma(x) = \frac{1}{1 + e^{-x}}} \quad (3.6)$$

$$\boldsymbol{T_2(x) = \sum_{j=1}^{N=3} w_j x_j + b} \quad (3.7)$$

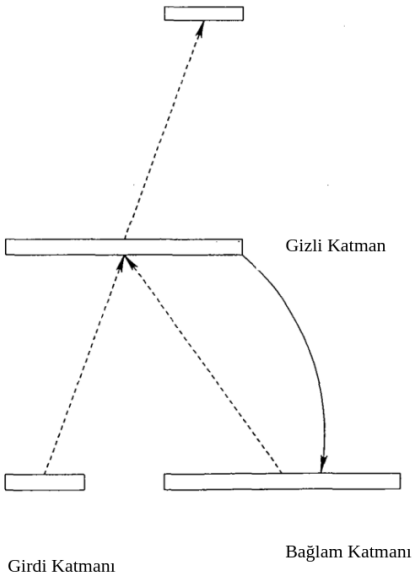
$$\boldsymbol{O_2(x) = \frac{1}{1 + \exp(-(w_1 x_1 + w_2 x_2 + w_3 x_3 + b))}} \quad (3.8)$$

Sigmoid fonksiyonu $\boldsymbol{\sigma(x)}$ olmak üzere, giriş katmanında bulunan üç nörondan alınan verilerin toplam fonksiyonunu belirten $\boldsymbol{T_2(x)}$ değeri sigmoid fonksiyonuna girdi olarak verilir. Elde edilen $\boldsymbol{O_2(x)}$ değeri 2 numaralı nöronun çıktı değerini ifade etmektedir. Bu değer sıfır olduğunda nöron aktive olmayacak 1 olduğunda ise aktive olup ateşlenecektir.

3.2.4. Geri Beslemeli Sinir Ağları

Geri beslemeli YSA, ileri beslemeli YSA gibi katmanlardaki tüm nöronların bir sonraki katmandaki nöronlara sıralı olarak bağlandığı bir yapıdan oluşmazlar. Bir nöronun çıktısı kendinden önceki katmanda bulunan bir nöronun girdisi olabilmektedir. Geri beslemeli ağlar tekrarlı ağlar olarak ta adlandırılmaktadır. Genellikle zaman serisi gibi sıralı problemlerinin çözümünde kullanılırlar. Yazılan metin girdilerine karşılık sonraki karakterin tahmin edilmesi gibi.

Geri beslemeli ağlar ağ içerisinde döngüler yaratarak iletilen sinyalleri her iki yönde de taşıyabilir. Bu ağlar oldukça güçlüdür ve son derece karmaşık bir yapıda olabilirler. Dinamiktirler ve durumları denge noktasına ulaşıncaya kadar sürekli değişmektedir. Giriş değişinceye kadar denge noktasında kalırlar [35]. Şekil 3.9' da geri beslemeli yapay sinir ağı görülmektedir.



Şekil 3.9 Geri beslemeli sinir ağı modeli

Elman ağı adı verilen basit tekrarlı ağ modelli ilk kez 1990 yılında Jeffrey L. Elman tarafından tasarlanmış ve kullanılmıştır. İleri beslemeli YSA' daki yapıya bir bağlam katmanı eklenmesi ile oluşan ağ yapısına sahiptir. Aktivasyonlar gizli katmandan bağlam katmanına, ağırlıklar bir olacak şekilde sabit olarak birebir kopyalanırlar. Şekil 3.9' da belirtilen noktalı kısmınlar öğrenilebilen bağlantıları ifade etmektedir. Gizli birimler, ağın belirli bir girdi için doğru çıktıyı üretmesini sağlayan kalıpları kodlayıp girdi kalıpları için içsel temsiller oluşturur. Bağlam birimleri sayesinde önceki içsel durum hatırlanır. Gizli katmandaki birimler hem harici bir girdi hem de istenen bazı çıktıların önceki halini haritalama görevine sahiptir. Gizli birimler üzerindeki örüntüler bağlam olarak kayıt edilir. Böylece içsel temsiller zamansal bağlama duyarlı hale gelir. Bu model doğal olarak sırasıyla sunulan girdilerin işlenmesini içerir [36, 37]. Bu tür modeller hava tahmini yapabilen uygulamalar içerisinde kullanılabilir.

3.3. Derin Öğrenme

Derin öğrenme yöntemi YSA yöntemlerinin derinleşmesi ile ortaya çıkan popüler bir makine öğrenmesi yaklaşımıdır. Bu yöntemde hesaplama yapısı YSA hesaplamalarına dayanmaktadır. Klasik YSA yaklaşımlarına kıyasla daha iyi başarılar

elde edilen bir yöntemdir. Daha önceki yıllarda donanım olanaklarının kısıtlı olmasından dolayı YSA' nın eğitilmesi zor olduğu için modellerdeki katman sayıları az sayıda tutulmaktaydı. İşlem gücü yüksek olan makinelerin hayatımıza girmesi ile birlikte daha derin modellerin eğitiminin yapılması mümkün olmuştur. Özellikle GİB' nin matris işlemlerini hızlı yapma özelliğinden dolayı modeller GİB'leri üzerinde eğitilmeye başlanmış ve daha derin modellerin MİB'ye kıyasla daha kısa sürede eğitilmesi sağlanmıştır.

Derin öğrenmenin popüler olmasında önemli etkenlerin başında büyük ölçüde artan işlemci yetenekleri(GİB gibi), eğitim işlemi için kullanılan veri sayısının artması sayılabilir [38]. Büyüyen veriye örnek olarak; Facebook platformunda her gün yaklaşık 350 milyon görüntü yüklenirken Youtube üzerinde ise her dakikada 100 saatlik video yüklenmektedir. Derin öğrenmede verilerden deneyim kazanıldığı için bilgisayarların problemleri daha iyi bir şekilde çözebilmesini sağlamada kuşkusuz gerçek dünyayı yansıtacak verilerin önemi büyüktür [39]. İnternet üzerinde görüntü, video, ses, metin gibi çeşitli türlerde artan veri gerçek dünyayı yansıtacak boyutlara ulaşmaya başlamıştır. Çeşitli verilerin, yeterli sayıda kullanılmasıyla daha etkin eğitim gerçekleştirme imkânı doğmuştur.

Derin öğrenmenin problem çözme yeteneği YSA' nın doğrusal olmayan problemleri çözebilme yeteneğinden gelmektedir. Modelin derinliğine paralel olarak artan katman sayısı karmaşık problemlerin çözümüne zemin hazırlamaktadır Ayrıca derin öğrenme modelleri az sayıda değişiklik ile farklı alanlardaki problemlerin çözümünde esneklik sağlarlar. Konuşma tanıma, doğal dil işlemenin yanı sıra derin öğrenme son yıllarda sıklıkla bilgisayar görüsü uygulamalarında görüntülerin sınıflandırılması, bölümlendirilmesi ve özellik çıkarımı için kullanılmaktadır.

3.3.1. Derin Öğrenme ile Görüntü Sınıflandırma

Bilgisayar görüsü dış ortamı belirli boyutlarda piksel matrisleri olarak algılamaktadır. Piksel matrisleri görüntünün bit sayısına göre belirlenen değerlerden oluşmaktadır. İnsan görüsünün karmaşık görüntüleri sınıflandırılması saniyeler sürerken, bilgisayarların bu işlemi temsili matrislerle saniyeler içinde yapabilmesi hem yeterli işlem gücü hem de bu işlemi en iyi şekilde gerçekleştirecek algoritma ihtiyacı doğurmaktadır.

Görüntü içerisindeki nesnenin piksel matrislerinin değeri ışığın geliş açısı, dış dünyada üç boyutlu olarak algılanabilen nesnelerin iki boyutlu olarak farklı açılardan değişik bir nesne gibi görünmesi, nesnelerin bir bölümünün başka bir cisim tarafından kapatılması gibi birçok faktöre bağlı olarak değişebilmektedir. Bunun için görüntü içerisindeki nesneye ait her özelliğin iyi bir şekilde analiz edilmesi gerekir. Veriden öğrenen bir yaklaşım kullanırken hangi özelliklerin hangi verilere ait olduğunun belirlenmesi gerekir. Ayrıca belirlenen özelliklerin nesneyi diğer nesnelerden farklı kılan ayırt edici özellikler olması gerekmektedir.

Andrew Ng 2011 yılında "Özelliklerle baş etmek zordur, zaman alıcıdır uzman bilgisi gerektirir. Öğrenme uygulamalarını çalışırken, özellikleri ayarlamak için çok zaman harcamaktayız." demiştir.

Derin öğrenmenin en önemli özelliklerinden biri klasik makine öğrenmesi tekniklerinden farklı olarak özellik mühendisliği çalışmasının yapılmasına gerek duyulmamasıdır. Derin öğrenmede veriler üzerinden otomatik olarak özellik çıkarımı yapılmaktadır. Görüntü sınıflandırma problemlerinde yaygın olarak konvolüzasyonel sinir ağları kullanılmaktadır. Bu ağlarda görüntü içerisindeki özelliklerin bulunması için eşleme yapılarak görüntü analiz edilir ve bu sayede eğitilmiş ağ karmaşık bir sahneyi analiz ederken nesnenin kısıtlı bir bölümünü görmüş olsa da özellik eşleme sayesinde ilgili parçanın hangi nesneye ait olduğunu algılayabilmektedir.

3.3.2. Konvolüzasyonel Sinir Ağları

KSA kendi içinde temel olarak iki bölümden oluşmaktadır. İlk bölüm bir veya daha fazla sayıda konvolüzasyon ve havuzlama katmanlarından oluşan özellik çıkarımının yapıldığı kısımdır. İkinci bölüm ise tam bağlı katmanlardan oluşan sınıflandırma işleminin yapıldığı kısımdır. Şekil 3.10' da KSA aşamaları görülmektedir.



Şekil 3.10 KSA aşamaları

Bu ağlar daha çok görüntü sınıflandırma alanında kullanılmaktadır. Ağda işleyiş olarak insanın görme mekanizması gibi belirli özellik çıkarımları yapılarak tahminler yürütülmektedir. Görüntü sınıflandırma problemlerinde zor olan görüntü piksellerinin içerisinden özellik çıkarımıdır. KSA zor olan bu süreci otomatik olarak ilk bölümde işleterek özellik haritalarını elde eder. Sonrasında ise elde edilen özellikler tam bağlı katmanlara iletilerek çok katmanlı algılayıcılar sayesinde belirli sayı değerleri elde edilerek görüntünün hangi sınıfa ait olduğu ile alakalı tahminler yapılır.

Bilgisayar görüsü uygulamalarında KSA' nın giriş katmanında görüntüye ait piksel değerleri girdi olarak kullanılır. Giriş katmanındaki nöron sayısı görüntünün içeriği renk kanal sayısına bağlı olarak değişmektedir. Tek kanallı gri tonlamalı bir görüntünün giriş katmanında tek bir sinir hücresi bulunurken, renkli bir görüntüde kırmızı, yeşil, mavi renk kanallarından oluşan üç adet sinir hücresi bulunur.

Görüntüye ait pikseller girdi olarak alınırken, tensör adı verilen çok boyutlu matrisler haline çevrilir. Genişlik ve yükseklik değerleri 200 olan bir renkli bir görüntü için 1x3x200x200 boyutunda bir tensör kullanılabilir. Bunun nedeni ağın belirli bir düzende matris işlemlerini yapmasını sağlamaktır.

3.3.2.1. Özellik Çıkarım Aşaması

Özellik çıkarım aşamasında sırayla bir veya birden fazla konvolüzasyon ve havuzlama işlemi uygulanmaktadır.

3.3.2.1.1. Konvolüzasyon Katmanı

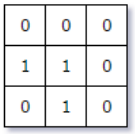
Konvolüzasyon katmanlarında amaç görüntünün piksel matrislerini kullanarak özellik haritalarının çıkarılmasını sağlamaktır. Bu katmanda görüntünün piksel matrislerine sol üstten başlayarak sağ alt noktaya inecek şekilde 3x3, 5x5 gibi farklı boyutlarda filtreler(ağırlıklar) kaydırma yapılarak uygulanır. Filtreler uygulanırken her bir kaydırma işleminde filtre ile aynı boyutta seçilen piksel matrisi ile çarpma işlemi yapılır ve elde edilen matris elemanlarının değerleri toplanarak sonuç elde edilir. İşlem en son piksele gelene kadar tekrarlandıktan sonra özellik haritaları oluşturulmuş olur.

Giriş katmanından sonraki katmanda uygulanan filtre sayısı görüntünün derinliği kadardır ve düşük seviye özellik çıkarımı yapılmasını sağlar. Sonraki konvolüzasyon

katmanlarında ise görüntünün belirli parçalarını içeren yüksek seviyeli özellikler belirlenmektedir. Görüntü üzerinde odaklanılacak özelliklerin ortalama boyutunun belirlenmesi ilk katmanda uygulanacak filtre boyutunun belirlenmesinde bir etken olarak kullanılabilmektedir. Filtreler sayesinde bir görüntünün çeşitli özelliklerine odaklanma sağlanmış birden farklı sürümü elde edilmektedir. Uygulanan filtreler ağırlıklardan oluşmaktadır. Ağırlıklar ise ağın eğitimi ile öğrenilmektedir. Özellik çıkarımının yapıldığı kısım KSA' yı diğer yapay sinir ağlarından farklı kılan kısımdır. Ağda ne kadar fazla konvolüzasyon katmanı bulursa o kadar fazla karmaşık özellikler belirlenebilmektedir.

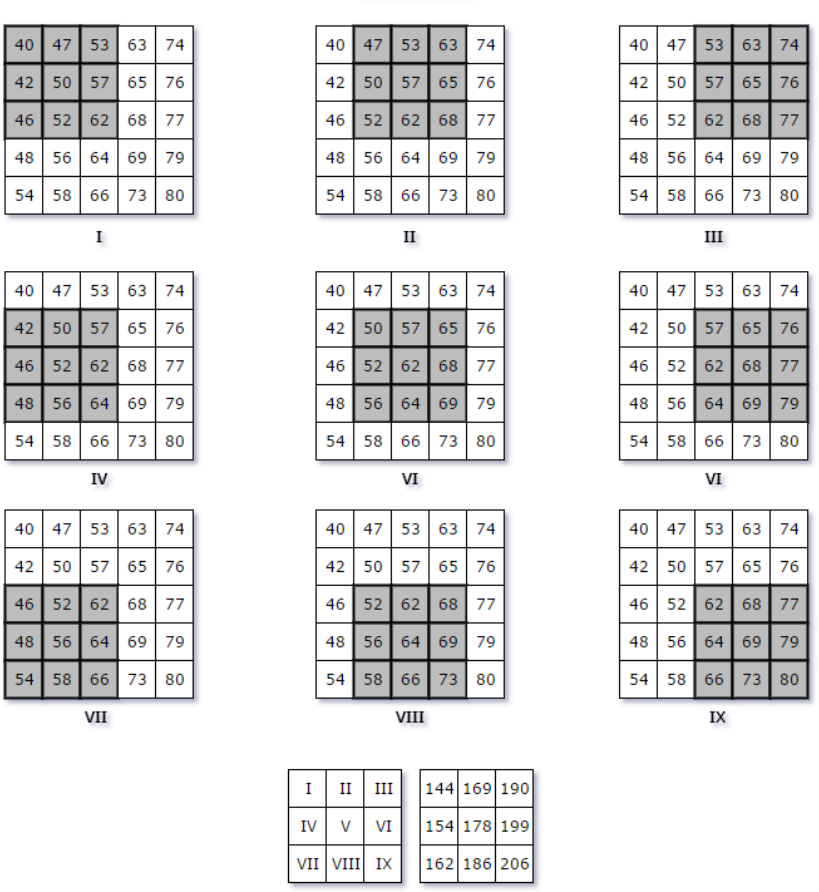
Örnek olarak üç kanallı 50x50 boyutunda bir görüntüye ilk konvolüzasyon katmanında 5x5 boyutunda filtreler uygulandığını varsayalım. Her kanala bu filtre uygulandıktan sonra 46x46 boyutunda özellik matrisleri elde edilecektir. Kaydırma işlemi uygulanırken filtre boyutunun dışında adım(stride) ve boşluk(padding) parametreleri de kullanılmaktadır. Adım parametresi, filtre uygulanırken filtrenin ne kadar aralıklarla kaydırılacağını belirtmektedir. Boşluk parametresi ise kenar değerlerinin işleme nasıl katılacağını ifade etmektedir. Konvolüzasyonel katmanın içerisinde filtre uygulama adımları aşağıda yer almaktadır.

1) Piksel matrislerine uygulanacak filtre boyutu belirlenir.
2) Adım ve boşluk değerleri belirlenir.
3) Piksel matrisinin en sol üst kısmına filtre uygulanır.
4) Filtre adım değerine göre eşleşme bitene kadar sağa kaydırılarak uygulanır.
5) En sol kısma gelinir ve adım değerine göre aşağıya doğru kaydırma yapılarak filtre uygulanır.
6) Eşleşme bitene kadar sırasıyla dördüncü ve beşinci adımdaki işlemler gerçekleştirilir.



Şekil 3.11 3x3 boyutunda filtre

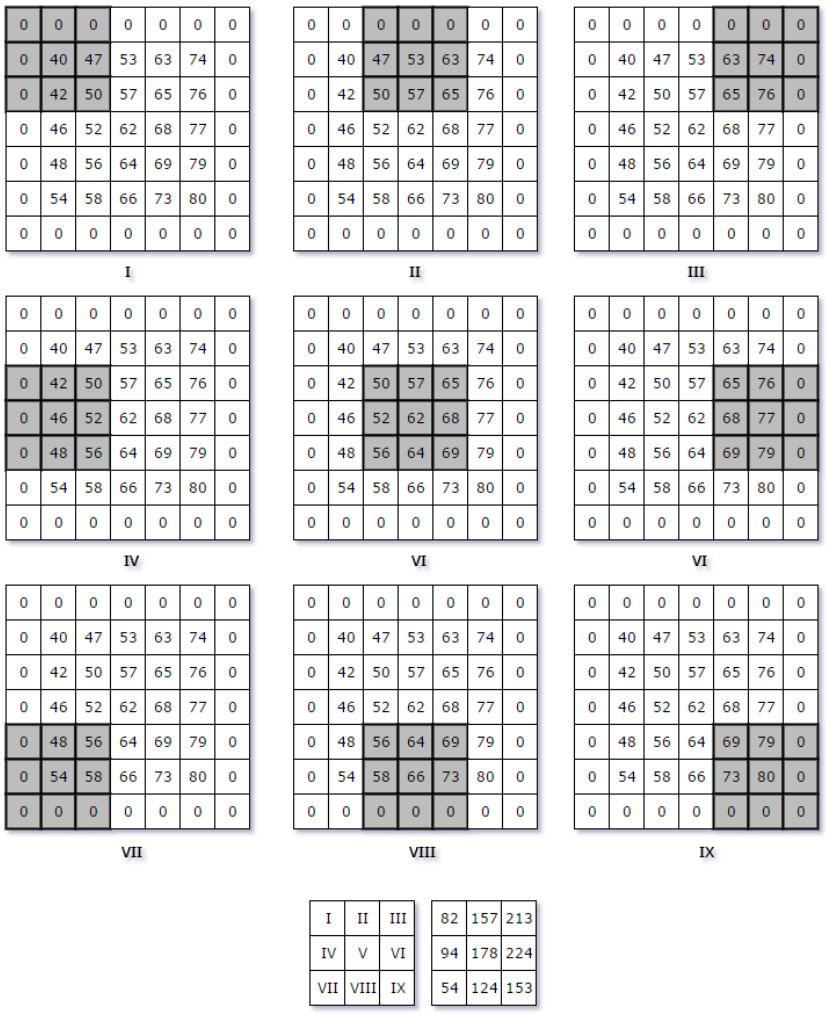
5x5 boyutunda bir piksel matrisine Şekil 3.11’ de görülen 3x3 boyutunda filtre bir adımlı olarak aşağıdaki gibi uygulanmaktadır. İşlemde kullanılan numaralandırılmış matrisler filtrelerin görüntü pikselleri ile eşleştiği noktaları göstermektedir. Çarpım yapılırken birbiri ile eşleşen noktaların sayı değerleri hesaba katılır.



Şekil 3.12 Bir adımlı konvolüzasyon işlemi

İşlem gerçekleştirilirken çarpım yapılan ilk piksel matrisinin hacmi küçülmektedir. Uygun sayıda sıfır boşluk (zero padding) yöntemi uygulandığında seçilen boşluk değerine göre kenarlara ve köşelere sıfır değerleri eklenerek hesap yapılır. Uygun adım sayısı, sıfır boşluk değeri, filtre boyutu belirlendiğinde hacim aynı kalabilir veya arttırılabilir. AxA boyutunda matrisi üzerinde N sayıda sıfır boşluk değeri seçilerek konvolüzasyon işlemine başlandığında görüntü piksellerinin kenarlarına ve köşe noktalarına sıfır değerleri eklenir ve bunun sonucunda (A+(2xN))x(A+(2xN))

boyutunda bir matris elde edilip bu matris üzerinde işlem yapılır. Yukarıdaki örnekte yapılan işlemi bir sıfır boşluk değeri ile gerçekleştirmek istediğimizde 5x5 boyutundaki matristen 7x7 boyutunda matris elde edilir, işlem iki adım şeklinde gerçekleştirildiğinde 3x3 boyutunda bir matris elde edilebilir. Eğer işlem bir adımlı olarak yapılırsa hacim değeri aynı kalacaktır.



Şekil 3.13 İki adımlı, sıfır boşluk konvolüzasyon işlemi

3.3.2.1.2. Havuzlama Katmanı

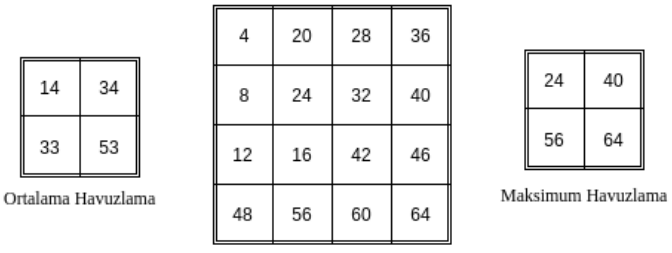
Havuzlama işlemi özellik çıkarım aşamasında konvolüzasyon katmanında gerçekleşen filtreleme işleminden sonraki aşamada uygulanmaktadır. Bu işlemin amacı

elde edilen özellik haritalarında boyutlarını indirgemektir. Ayrıca aşırı uyum (overfitting) probleminden kaçınmak için kullanılan bir yöntemdir.

Havuzlama işleminde konvolüzasyon katmanında olduğu gibi, çeşitli boyutlarda filtreler kullanılarak matrisler üzerinde hesaplamalar yapılmaktadır. Havuzlama çerçevesinin değeri seçilerek, belirlenen adım parametresi dikkate alınarak özellik matrisleri üzerinde maksimum ve ortalama havuzlama olmak üzere iki farklı işlem yapılabilir. Bu çalışmada maksimum havuzlama katmanları kullanılmıştır. Maksimum havuzlama katmanında havuzlama çerçevesi ile özellik matrisinde eşleşen değerlerin en büyüğü alınmaktadır. Ortalama havuzlama katmanında havuzlama çerçevesi ile özellik matrisinde eşleşen değerlerin ortalaması alınmaktadır. Havuzlama katmanında gerçekleştirilen filtre uygulamasının adımları aşağıda yer almaktadır.

1) Havuzlama çerçevesi boyutu belirlenir.
2) Adım parametresi değeri belirlenir.
3) Özellik matrisinin en üst sol köşesine belirlenen havuzlama çerçevesi kadar havuzlama işlemi uygulanır.
4) Çerçeve adım parametresi değerine göre en sona gelene kadar sağa kaydırılarak uygulanır.
5) En sol kısma gelinir ve çerçeve, adım parametresi değerine göre aşağıya doğru kaydırılarak uygulanır.
6) Eşleşme bitene kadar sırasıyla dördüncü ve beşinci adımdaki işlemler gerçekleştirilir.

Şekil 3.14' te 4x4 boyundaki matris üzerinde, 2x2 çerçeve boyutunda maksimum ve ortalama havuzlama katmanı uygulandığında elde edilen sonuçlar görülmektedir.



Şekil 3.14 Ortalama havuzlama ve maksimum havuzlama işlemleri

3.3.2.2. Sınıflandırma Aşaması

Sınıflandırma aşaması KSA' nın özellik çıkarımı aşamasından sonra yer almaktadır. Bu aşama özellik haritalarını girdi olarak almaktadır. Sınıflandırma yapılmasını sağlayan tam bağlı katman çok katmanlı algılayıcı yapısında tasarlanmaktadır. Ağın çıkışında elde edilecek değerler sayı değerleri olacağı için çok boyutlu veriler(matrisler) ile işlem yapıldıktan sonra bu katmanda boyut indirgeme yapılarak veri tek boyutlu hale getirilir.

İkili sınıflandırma ve çok sınıflı sınıflandırma olmak üzere iki şekilde gerçekleştirilebilir. İkili sınıflandırmada iki adet sınıf bulunur. Bir görüntü içerisinde bir nesne bulunduğunda pozitif işaretleme, nesne bulunmadığında ise negatif işaretleme yapılarak nesnenin görüntü içerisinde olup olmadığı anlaşılabilir. Çoklu sınıflandırmada ise n adet sınıf bulunur ve amaç bir verinin n adet sınıftan hangisine ait olduğunu belirlemektir. Yolcu kapasitesi, kapı sayısı ve arabanın yükseklik değerlerine göre spor ya da normal araba olduğunu tahmin eden bir YSA ikili sınıflandırma örnek olarak verilebilir. Çoklu sınıflandırmaya ise taç yaprak, çanak yaprak ve renk özelliklerine bakarak n adet çiçek sınıfından tahminler yürüten bir YSA örnek olarak verilebilir.

Sınıflandırma probleminde ağın çıktısı eğitimde kullanılan sınıf sayısına eşittir. Çıkış katmanında özellik haritasından görüntünün hangi sınıfa ait olduğunu belirtmek için hangi sınıfla daha çok ilişkili olduğunu belirten olasılık değerleri kullanılabilmektedir. Olasılık değerlerini elde etmek için softmax fonksiyonu kullanılmaktadır.

3.3.2.2.1. Softmax Fonksiyonu

Softmax fonksiyonu KSA' nın son katmanında kullanılmaktadır. Bunun nedeni elde edilen sayısal değerlerin hangi sınıflar ile ilişkili olduğunun bulunmasıdır.

$$\mathbf{f}(\mathbf{x_i}) = \frac{\mathbf{e}^{\mathbf{x_i}}}{\sum_{\mathbf{j}} \mathbf{e}^{\mathbf{x_j}}} \qquad (3.9)$$

Fonksiyon bir sayı dizisini girdi olarak alır ve çıktı olarak 0 ve 1 değerleri arasında sonuçlar üretir. Olasılık ifade ettiği için sonuç değerlerinin toplamı 1 olmalıdır. Çok

sınıflı sınıflandırma işleminde, sonuç olarak en büyük değere sahip sınıf doğru tahmin olarak kabul edilir. Bu çalışmada çok sınıflı sınıflandırma yapıldığı için son katmanda aktivasyon fonksiyonu olarak Softmax fonksiyonu kullanılmıştır.

3.3.3. Öğrenme Süreci

Makine öğrenmesindeki tekniklerde olduğu gibi derin öğrenmede de verilerden öğrenme durumu söz konusudur. Öğrenme süreçleri denetimli, denetimsiz ve takviyeli öğrenme olmak üzere üçe ayrılır. Tüm süreçlerde, veri sistemin en önemli parçalarından birini oluşturur. Veriler çözülmesi istenilen problem türüne bağlı olarak gerçek dünyayı yansıtacak çeşitlilikte olmalıdır.

Eğitim yapılırken ağırlık değerleri istenen sonuçları verecek şekilde güncellenir ve ağ üzerinde verileri belirli bir düzene göre ayırt edebilecek en uygun matematiksel fonksiyon elde edilmiş olur. Herhangi bir problemi çözmek için geliştirilip eğitilen bir model, farklı veriler ile tekrar eğitildiğinde farklı bir problemin çözümü için kullanılabilir.

3.3.3.1. Denetimli Öğrenme

Denetimli öğrenmede veriler makineye etiketli bir biçimde verilmektedir. Yani belirli girdi değerlerine göre çıktının ne olacağı söylenmektedir. Denetimli öğrenme sınıflandırma ve regresyon problemlerinde de kullanılmaktadır. Ağ modeli, denetimli öğrenme ile eğitilirken değişik koşulları barındıran çok sayıda veri ile öğrenme gerçekleştirilmektedir. Öğrenmedeki amaç sistemin hata oranının düşmesini sağlamaktır. Hata oranının düşmesi sağlanırken ağın ağırlık değerleri de istenen durumları sağlayacak şekilde güncellenmektedir. Belirli girdilere göre minimum hata ile tahmin yapılır. Hata değeri beklenilen değer ile gerçek değerin arasındaki fark olarak ifade edilmektedir.

$$\mathbf{Hata} = \mathbf{Tahmin} - \mathbf{Gerçek} \qquad (3.10)$$

Çıkış katmanındaki tüm düğümleri hesaba katarak toplam hatayı hesaplamak için maliyet(loss-cost) fonksiyonları kullanılmaktadır. Öğrenme aşamasında veriler

fazla sayıda olduğu için sisteme yığınlar halinde verilmektedir. A adet veri seti B adet yığınlar halinde sisteme verildiğinde sistem, C=A/B kadar tekrarda eğitilmektedir. C adet tekrar gerçekleştiğinde ise sistem bir epok(epoch) gerçekleştirmiş olur.

3.3.3.2. Denetimsiz Öğrenme

Denetimsiz öğrenme sürecinde makineye verilen veriler etiketsizdir. Denetimli öğrenmede olduğu gibi girdi değerlerine karşılık hangi çıktı değerlerinin elde edilmesi gerektiği söylenmemektedir. Bunun yerine sistem kendi kendine verilerden anlamlar çıkararak öğrenmektedir. Ağ güncellenirken ağın belirlediği kurallara göre ağırlıkları da güncellenmektedir. Bu yöntem çok sayıda veri içerisinden örüntülerin belirlenmesi için kullanılmaktadır.

Derin öğrenmedeki ana fikir sadece giriş ve çıkışlar arasındaki doğrusal olmayan haritalama değil, aynı zamanda girdi vektörlerinin altında yatan yapıyı öğrenmektir [40]. Denetimsiz öğrenmede YSA' da otomatik kodlayıcılar(autoencoder) kullanılmaktadır. Otomatik kodlayıcılar verileri etiketleri ile öğrenmezler ve girdi olarak aldığı verilerin yapıları hakkında bilgiler verirler. Görüntü verileri üzerinde bu yöntem kullanıldığında 50x50 boyutunda gri tonlamalı bir matris için giriş katmanında tüm pikselleri girdi olarak alan 2500 adet nöron bulunur. Tüm piksel değerleri kullanılarak gizli katmanlar sayesinde özellik belirlenme işlemi yapılır.

3.3.3.3. Takviyeli Öğrenme

Takviyeli öğrenme sürecinde denetimsiz öğrenmedeki gibi verilerin etiketleri bilinmemektedir. Fakat sonuç olarak elde edilen çıktının doğru ya da yanlış olduğu söylenerek sistemin kendini iyileştirmesine yardımcı olunmaktadır. Belirli sayıda adım sonrasında doğru bir sonuç elde edilirse ödül, yanlış sonuç elde edildiğinde ceza sinyali gönderilir. Küçük bir çocuğun soba gibi sıcak bir cisme dokunduğunda sinir sistemi sayesinde acıyı hissederek o cismin acı hissi verebilecek bir cisim olduğunu öğrenmesi ve aynı senaryoyu tekrar yaşadığında o cisme dokunmaması takviyeli öğrenme sürecine örnek olarak verilebilir. Derin öğrenmede genellikle oyun oynayabilecek bilgisayar zekâsı, oyun ajanları oluşturmak için kullanılmaktadır.

3.3.4. Veri Setlerinin Seçimi

Veri setlerinin seçiminde veriler temel olarak eğitim ve test verileri olarak ikiye ayrılmaktadır. Oluşturulan model eğitim verileri ile eğitilirken modelin başarısı daha önce hiç görmediği test verileri ile değerlendirilmektedir. Denetimli öğrenme sürecinde eğitim yapılırken maliyet fonksiyonunun değeri en aza indirgenerek eğitim verilerine göre verileri en iyi şekilde temsil edecek fonksiyon elde edilmeye çalışılır. Eğitilen model bazen sadece eğitim veri setini temsil etmektedir. Bu durumda yeni veriler ile test yapıldığında tahmin yeteneği azdır. Eğitim verilerini ezberlendiği için model sadece eğitim verisi setinde yer alan örnekleri başarı olarak yorumlayabilir.

Test veri setleri ile yapılan değerlendirme sonucunda elde edilen doğruluk oranının, eğitimde elde edilen doğruluk oranından çok daha düşük olması istenmeyen bir durumdur. Bu duruma aşırı uyum(overfitting) adı verilmektedir. Bazı durumlarda ise eğitilen model eğitim setinde de başarısız tahminler yapmaktadır. Bu probleme alçak uyum(underfitting) adı verilir. Arzu edilen durum ise modelin hem eğitim hem de test verilerine yakın sonuçlar vererek daha önce görmediği veriler üzerinde tahmin yapabildiği iyi uyumu yakalamaktır.

Derin öğrenmede aşırı uyum probleminden kaçınmak için yeterli sayıda, yeterli çeşitlilikte veri kullanılabilir. Karmaşık model basitleştirilebilir veya gereğinden fazla sayıda tekrar ile modelin eğitim verilerini ezberlemesini engellenebilir. Bunu yapabilmek için erken durdurma (early stopping) adı verilen yeterli tekrar sayısına ulaşıldığında eğitimi durdurma işlemi gerçekleştirilebilir. Bunların yanında regularizasyon teknikleri de kullanılmaktadır. Bu çalışmada bu sorunların önüne geçmek için regularizasyon tekniği olarak dropout tekniği kullanılmıştır.

Veri setleri eğitim, test ve doğrulama veri setleri olarak üçe ayrılabilmektedir. Model eğitilirken aşırı uyum problemi doğrulama veri setini kullanarak saptanabilmektedir. Doğrulama veri seti tasarlanan modelin eğitim verileri dışındaki veriler üzerinde nasıl sonuçlar verdiğini anlayarak modelin iyileştirilmesi için parametrelerin ayarlanmasında kullanılmaktadır. Bu yüzden doğrulama veri setinin kullanımı gerçek başarımı elde etmek için önemlidir.

3.3.5. Maliyet Fonksiyonları

Maliyet fonksiyonları hata fonksiyonları olarak da ifade edilmektedir. Bu fonksiyonlar ağın hatasını temsil ederler ve eğitim sürecinde minimize edilirler. Çeşitli maliyet fonksiyonları bulunmaktadır. Maliyet fonksiyonu seçimi probleme göre yapılmaktadır. Regresyon problemlerinde yaygın olarak ortalama hata karesi, ortalama mutlak hata, karekök ortalama kullanılırken, sınıflandırma problemlerinde ise çapraz entropi maliyet fonksiyonu kullanılmaktadır.

3.3.5.1. Ortalama Hata Karesi (Mean Squared Error)

Ortalama hata karesi, regresyon problemlerinde yaygın olarak kullanılmaktadır. Aşağıdaki gibi ifade edilir.

$$\mathbf{OHK}(\mathbf{w}) = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\mathbf{y}_{\boldsymbol{i}} - \hat{\mathbf{y}}_{\boldsymbol{i}})^{\mathbf{2}} \qquad (3.11)$$

$$\mathbf{OHK}(\mathbf{w}) = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\left(\mathbf{y}_{\boldsymbol{i}} - \mathbf{f}(\mathbf{x}_{\boldsymbol{i}}, \mathbf{w})\right)^{\mathbf{2}} \qquad (3.12)$$

$$\mathbf{OHK}(\mathbf{w}) = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\mathbf{e}_{\boldsymbol{i}}^{\mathbf{2}} \qquad (3.13)$$

Mevcut eğitim verileri kullanılarak ağdaki ağırlıkların performans ölçütü olarak kullanılabilmektedir. İşlem yapılırken eğitim verileri içerisinden her i örneği için $\mathbf{x}_{\boldsymbol{i}}$ girdisi $\mathbf{w}$ ağırlık değerleri ile birlikte alınır, ağın çıktı değeri olan $\hat{\mathbf{y}}_{\boldsymbol{i}}$ değeri elde edilir. Ağın beklenen çıktı değeri olan $\mathbf{y}_{\boldsymbol{i}}$, $\hat{\mathbf{y}}_{\boldsymbol{i}}$ değeri ile karşılaştırılarak fark hesaplanarak $\mathbf{e}$ değeri bulunur. Ortalama hata değerini bulmak için $\mathbf{e}$ 'nin karesi alınarak eğitim verilerinin sayısı kadar toplanır ve ortalaması alınır.

3.3.5.2. Ortalama Mutlak Hata (Mean Absolute Error)

Ortalama mutlak hata hesaplanırken veri sayısı kadar her bir hatanın ayrı ayrı mutlak değeri alınır ve ortalaması hesaplanır. Sonuç değeri aşağıdaki gibi elde edilir.

$$\mathbf{OMH} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} |\mathbf{y}_{\boldsymbol{i}} - \hat{\mathbf{y}}_{\boldsymbol{i}}| \qquad (3.14)$$

$$\mathbf{OMH} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} |\mathbf{e}_{\boldsymbol{i}}| \qquad (3.15)$$

Bu metotta da ortalama hata karesinde olduğu gibi elde edilen çıktı $\hat{\mathbf{y}}_{\boldsymbol{i}}$ değerinin $\mathbf{y}_{\boldsymbol{i}}$ değerine yakınsanması amaçlanır.

3.3.5.3. Karekök Ortalama (Root Mean Square)

Ortalama hata karesinin hesaplanırken veri sayısı kadar hatanın karesi alınır, toplanır ve sonucun karekökü alınır. Aşağıdaki gibi ifade edilebilir.

$$\mathbf{KO} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\mathbf{y}_{\boldsymbol{i}} - \hat{\mathbf{y}}_{\boldsymbol{i}})^{\mathbf{2}}} \qquad (3.16)$$

$$\mathbf{KO} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\mathbf{e}_{\boldsymbol{i}})^{\mathbf{2}}} \qquad (3.17)$$

Karekök ortalama ve ortalama mutlak hata sıklıkla birbiri yerine kullanılmaktadır [41]. Karekök ortalama, tüm ağın hata dağılımının tam olarak betimlenmesine yardımcı olmaktadır. Elimizde en az yüz adet ve daha fazla sayıda örnek olduğunda, büyük hata değerleri ile kullanılması daha uygundur [42].

3.3.5.4. Çapraz Entropi(Cross Entropy)

Çapraz Entropi fonksiyonu sınıflandırmada problemlerinde kullanılan bir maliyet fonksiyondur. Aşağıdaki gibi tanımlanmaktadır.

$$\mathrm{CE} = -\frac{1}{N}\sum_{i=1}^{N}[\mathrm{y}_i \ln \hat{\mathrm{y}}_i + (1-\mathrm{y}_i)\ln(1-\hat{\mathrm{y}}_i)] \quad (3.18)$$

Başında bulunan negatif işareti sayesinde öncesinde anlatılan maliyet fonksiyonları gibi negatif sonuçlar vermez. (CE > 0) Bir nöron çıktısı $\mathbf{y_i}$ değeri ile beklenen değer olan $\hat{\mathbf{y}}_i$ birbirine yakın olduğunda CE sonuç değeri sıfıra yakın bir değer alacaktır. Diğer maliyet fonksiyonlarından farklı olarak öğrenme yavaşlama problemine çözüm sunmaktadır. Bunun nedenini anlamak için kısmi türev alınarak inceleme yapılabilir. Sigmoid aktivasyon fonksiyonu kullandığımızı varsayarak, $\boldsymbol{\sigma'(z) = \sigma(z)(1-\sigma(z))}$, $\hat{\mathbf{y}}_i = \boldsymbol{\sigma(z)}$ eşitliklerini kullandığımızda $\boldsymbol{x}$ girdi, y çıktı değerine göre bir nöronun herhangi bir ağırlığının ve bias değerinin kısmı türevi aşağıdaki gibi ifade edilebilir [43].

$$\frac{\partial \mathbf{CE}}{\partial \boldsymbol{w_i}} = -\frac{1}{N}\sum_{x}\left(\frac{\boldsymbol{y}}{\boldsymbol{\sigma(z)}} - \frac{(1-\boldsymbol{y})}{1-\boldsymbol{\sigma(z)}}\right)\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{w_i}} \quad (3.19)$$

$$\frac{\partial \mathbf{CE}}{\partial \boldsymbol{w_i}} = -\frac{1}{N}\sum_{x}\left(\frac{\boldsymbol{y}}{\boldsymbol{\sigma(z)}} - \frac{(1-\boldsymbol{y})}{1-\boldsymbol{\sigma(z)}}\right)\boldsymbol{\sigma'(z)x_i} \quad (3.20)$$

$$\frac{\partial \mathbf{CE}}{\partial \boldsymbol{w_i}} = \frac{1}{N}\sum_{x}\left(\frac{\boldsymbol{\sigma'(z)x_i}}{\boldsymbol{\sigma(z)}(1-\boldsymbol{\sigma(z)})}\right)(\boldsymbol{\sigma(z)} - \boldsymbol{y}) \quad (3.21)$$

$$\frac{\partial \mathbf{CE}}{\partial \boldsymbol{w_i}} = \frac{1}{N}\sum_{x}\boldsymbol{x_i}(\boldsymbol{\sigma(z)} - \boldsymbol{y}) \quad (3.22)$$

Ağırlıkların ve bias değerlerinin öğrenme hızı beklenen çıktı ile elde edilen sigmoid çıktısı arasındaki farkla yani $(\boldsymbol{\sigma(z)} - \boldsymbol{y})$ ifadesi ile belirtilen çıkış hatası ile

kontrol edilmektedir. Hata değerine göre öğrenme hızı değişmektedir, büyük hata değerinde daha hızlı öğrenme sağlanacaktır. Bu çalışmada Çapraz Entropi yöntemi kullanılmıştır. Bu yöntemin kullanılmasının nedeni, yöntemin öğrenme yavaşlama problemini çözmesidir.

3.3.6. Eğim Azalımı (Gradient Descent)

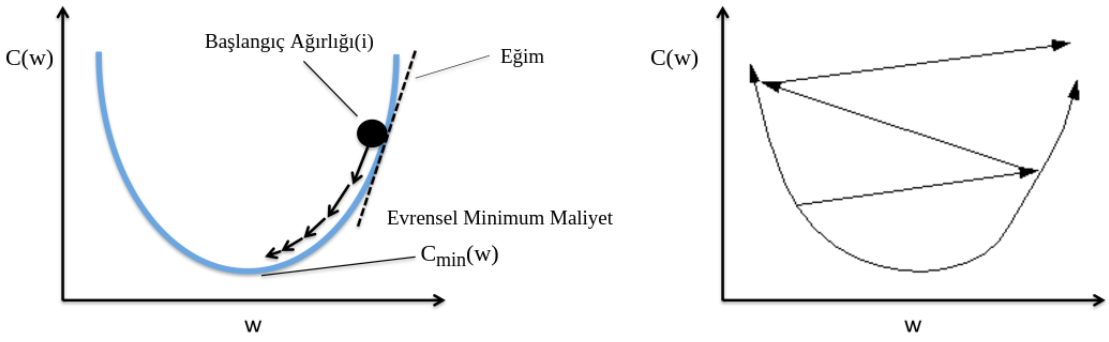
Eğim azalımı, YSA ve derin öğrenme yöntemlerinde ağın öğrenmesini sağlayan bir iyileştirme yöntemidir. Maliyet fonksiyonunun minimum olacak şekilde azaltılmasını sağlamak için ağın ağırlık ve bias değerlerinin bulunmasını sağlamaktadır. Bir ağ eğitilirken ağırlık değerlerinin güncellenmesinde aşağıdaki algoritma adımları kullanılmaktadır.

1) Ağırlık değeri(w_0) rastgele bir değer ile belirlenir.
2) Maliyet fonksiyonunu azaltmak için ağırlıklar Denklem 3.23'teki gibi eğitim verilerinin tümüne göre güncellenir.
3) Bir sonraki adıma geçmek için zaman değeri bir sonraki zaman değeri ile güncellenir ($t = t + 1$)
4) Minimum nokta bulunana kadar sırasıyla 2. ve 3. adım tekrar edilir.

$$\mathbf{w_{t+1}} = \mathbf{w_t} - \boldsymbol{\eta \nabla C}(\mathbf{w_t}) \qquad (3.23)$$

Her epoktan sonra ağırlık ve bias değerleri güncellenmektedir. Şekil 3.15' te bir maliyet fonksiyonunun değerinin tek bir ağırlığa bağlı olarak değişimi görülmektedir. Ağırlık değerinin ilk noktası i ile gösterilen noktadır. Bu noktadan başlanarak ağırlık aşamalı olarak güncellenir ve en düşük maliyet fonksiyonu değeri elde edilmeye çalışılır. Ağrılık güncellenirken eğitim dikliği yani maliyet fonksiyonunun ağırlığa göre türevi $\boldsymbol{\nabla C}(\mathbf{w_t})$ ve η öğrenme oranı ile çarpım işlemi gerçekleştirilir. Elde edilen değer $\mathbf{w_t}$ ağırlık değerinden çıkartılarak yeni ağırlık değeri $\mathbf{w_{t+1}}$ elde edilir. Öğrenme oranı, eğitim sırasında ağırlıkların ne büyüklükte güncelleneceğini, dolayısıyla her adım büyüklüğünü belirlemede kullanılmaktadır. Küçük değerli bir öğrenme oranı belirlendiğinde modelin doyma noktasına ulaşması iyi öğrenme oranına göre çok zaman alacaktır. Büyük değerli bir öğrenme oranı belirlendiğinde ise güncelleme işlemleri Şekil 3.15'te sağda yer alan görseldeki okların yönünde yapılır. Sağa ve sola doğru

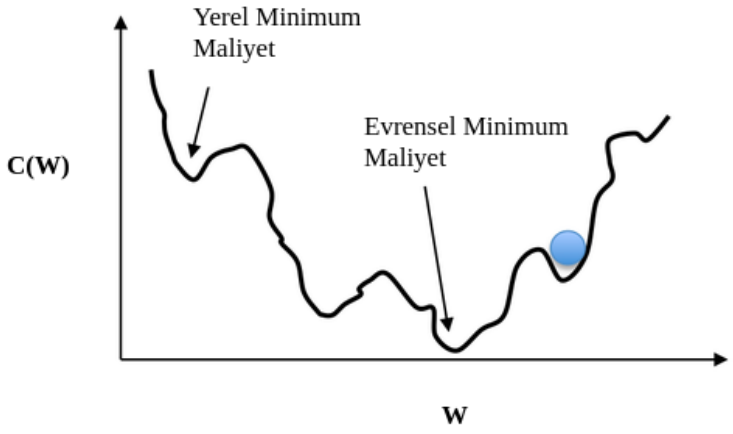
gerçekleşen sıçrayış hareketleri nedeni ile doyma noktasından uzaklaşılır. Uygun bir öğrenme oranı ile doyma noktasına ulaşma ise solda yer alan görseldeki gibidir.



Şekil 3.15 Doyma noktasına ulaşım

3.3.7. Momentum

Pratikte sadece bir ağırlık parametresinin ayarlanmasıyla, az sayıda adımla, hızlı bir şekilde en küçük noktanın bulunması zordur. Bunun için milyonlarca parametrenin ayarlanması ile çok sayıda tekrar yaparak doyma noktasına ulaşılabilir. Şekil 3.16' daki durumda belirli bir alanda, kendi içerisinde en düşük noktaları barındıran yerler bulunmaktadır. Buradaki en düşük noktalara yerel minimum adı verilir. Eğim azalımı algoritması güncellemeleri yaparken adımlar bir önceki noktadan daha düşük noktaya doğru atılır. Rastgele başlatılan ağırlık değeri yerel minimuma yakın bir yerde başlatılırsa en düşük nokta olarak yerel minimum seçilecektir. Asıl ulaşılmak istenen nokta ise yerel minimum noktaları değil, tüm minimum noktalara kıyasla en küçük olan noktadır. Bu nokta ise evrensel minimum olarak adlandırılmaktadır. Karmaşık durumlarda bu noktaya ulaşmak için sadece aşağıya doğru iniş yeterli değildir.



Şekil 3.16 Evrensel ve yerel minimum noktaları

Yerel minimum noktalarının aşılarak, evrensel minimum noktasına ulaşmaya yardımcı olan momentum olarak adlandırılan bir teknik bulunmaktadır. Denklem 3.24' deki gibi 0 ve 1 arasında değer alabilen $\boldsymbol{a}$ momentum değerinin ağırlık değeri ile çarpma işlemi yapılarak uygulanması ile yerel noktalara takılmadan en küçük noktaya ulaşabilmek için daha büyük adımlar atılarak güncelleme işlemi daha hızlı olarak yapılmaktadır, dolayısıyla öğrenme süresi kısalmaktadır.

$$\mathbf{w_{t+1}} = \boldsymbol{a}\mathbf{w_t} - \boldsymbol{\eta}\boldsymbol{\nabla C}(\mathbf{w_t}) \qquad (3.24)$$

Momentum değeri küçük olduğunda öğrenme süresi uzar ve istenen noktaya ulaşılamayıp yerel minimum noktasından çıkılamayabilir. Değer büyük olduğunda ise doyma noktası geçilebilmektedir. Öğrenme oranı ile momentum arasında ters ilişki vardır, küçük öğrenme oranı ile büyük momentum kullanılması daha uygundur.

3.3.8. Stokastik Eğim Azalımı (Stochastic Gradient Descent)

Stokastik eğim azalımı, eğim azalımı algoritmasının farklı bir sürümüdür. Eğim azalımı algoritmasında tüm eğitim verileri ile maliyet fonksiyonunun eğim hesaplaması yapılır ve ortalama alınarak parametrelerin güncellenmesi işlemi gerçekleştirilir. Stokastik olan algoritmada ise tüm veri seti üzerinde işlem yapmak yerine eğitim verilerinin içerisinden bir veya daha fazla sayıda rastgele örnek seçilir ve her örneğe

göre güncelleme yapılır. Seçilen örnekler bitene kadar işleme devam edilir. Büyük veri setlerinin kullanıldığı durumlarda eğim azalımı algoritmasında tüm veri setleri için tek seferde hesaplama yapılması maliyetlidir ve zaman almaktadır. Belirli örnekler üzerinde bu işlemleri gerçekleştirmek hız kazandırmaktadır. Eğitim seti içerisinden seçilen n sayıda x girdisi için bir ağırlığın güncellenmesi işlemi, n adet örneğin içerisindeki her x_i için aşağıdaki gibi ifade edilebilir.

$$w_{t+1} = w_t - \frac{\eta}{n}\sum_i \frac{\partial C_{x_i}}{\partial w_t} \quad (3.25)$$

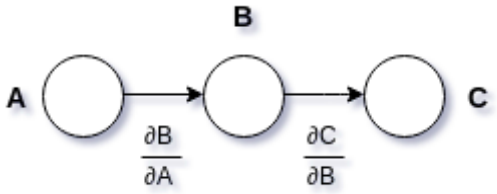
3.3.9. Geri Yayılım (Back propagation)

YSA ve derin öğrenme tekniklerinde ağın öğrenme sürecinde eğim azılımı algoritmasının yanında geri yayılım algoritması kullanılmaktadır. Geri yayılım algoritması türevlerin etkili bir şekilde hesaplanmasını sağlamaktadır. Ağın eğitiminde ağırlıkların ve bias değerlerinin güncelleme işlemi gerçekleştirilirken maliyet fonksiyonunun ağırlık veya bias değerlerine kısmi türevleri kullanılmaktadır. Kısmi türevlerin kullanılmasının amacı ise sistemde bulunan öğrenilebilen parametrelerin hata değerine etkisini hesaplamaktır. Ağın eğitiminde temel olarak geri yayılım algoritması her örnek için aşağıdaki adımlar ile gerçekleştirilmektedir.

1. Alınan örnek ile ağ üzerinde ileri hesaplama yapılır.
2. Çıkış katmanındaki birimlerin hata çıktısı hesaplanır.
3. Hesaplanan hata değerlerinin kısmı türevleri alınarak çıkış katmanından giriş katmanına doğru geri yayılımı yapılır.
4. Eğim azılım algoritması ile öğrenilebilen parametreler kısmı türevlere göre güncellenir.

Farklı sayıdaki gizli katmana sahip ağlarda her katman hatayı etkiler ve her katmanın farklı hatası vardır. Son katmanın hatası basitçe maliyet fonksiyonu ile elde edilen değerdir. Gizli katmanların hatası ise farklı bir biçimde tanımlanmaktadır. Geri yayılımda önceki katmanın ağırlıklarını güncellemek için bir sonraki katmanın hatası kullanmaktır. Hata, çıkış katmanından başlayarak geri doğru yayıldığı için geri yayılım adı verilmektedir. Ağırlıkların güncellenmesi çıkış katmanlarında ve gizli katmanlarda

farklılık göstermektedir [44]. Tüm gizli katmanlar için hata türevleri hesaplanmaktadır. Gizli katmanlar için hata türevlerini elde ettiğimizde gizli katmanlara bağlı ağırlıkların türevini hesaplamak kolaylaşır. İşlemler zincir kuralına göre yapılır [45].



Şekil 3.17 Basit zincir kuralı

A,B ve C katmanları bulunan küçük bir sinir ağı modelinde C çıkış katmanından elde edilen hatanın geriye yayılımını ifade eden basit bir zincir kuralı aşağıdaki gibi tanımlanabilir.

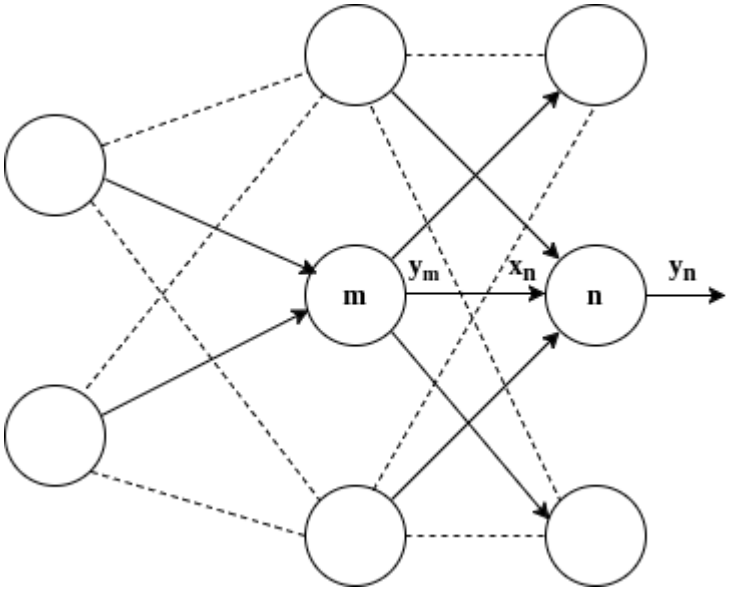
$$\Delta \boldsymbol{C} = \frac{\partial \mathbf{C}}{\partial \boldsymbol{B}} \Delta \boldsymbol{B} \qquad (3.26)$$

$$\Delta \boldsymbol{B} = \frac{\partial \mathbf{B}}{\partial \boldsymbol{A}} \Delta \boldsymbol{A} \qquad (3.27)$$

$$\Delta \boldsymbol{C} = \frac{\partial \mathbf{C}}{\partial \boldsymbol{B}} \frac{\partial \mathbf{B}}{\partial \boldsymbol{A}} \Delta \boldsymbol{A} \qquad (3.28)$$

$$\frac{\partial \mathbf{C}}{\partial \boldsymbol{A}} = \frac{\partial \mathbf{C}}{\partial \boldsymbol{B}} \frac{\partial \mathbf{B}}{\partial \boldsymbol{A}} \qquad (3.29)$$

Geri yayılım algoritmasında katmanlardaki sinir hücrelerini bağlayan ağırlık değerlerinin tümü hesaba katılır. Aşağıda üç adet sinir hücresi içeren bir gizli katmana sahip sinir ağı modeli yer almaktadır. Modelin gizli katman ve çıkış katmanından m ve n sinir hücreleri seçilmiştir. Bu sinir hücreleri arasında hata hesabı yapılırken işleme alınacak ağırlık değerlerine ait yollar düz çizgi, işleme alınmayacak yollar ise kesikli çizgiler ile belirtilmiştir.



Şekil 3.18 Sinir ağı üzerinde geri yayılım

Son katmandaki n sinir hücresinin girdisi gizli katmandaki hücrelerin çıkışına bağlı olan ağırlık yolların toplamından oluşmaktadır ve x_n ile ifade edilmiştir.

$$\mathbf{x_n} = \mathbf{b_n} + \sum_{m} \mathbf{y_m w_{mn}} \qquad (3.30)$$

Hatanın yayılımında kullanılacak kısmı türevler ise aşağıdaki gibi tanımlanabilir.

$$\frac{\partial \mathbf{x_n}}{\partial \mathbf{w_{mn}}} = \mathbf{y_m} \qquad (3.31)$$

$$\frac{\partial \mathbf{x_n}}{\partial \mathbf{y_m}} = \mathbf{w_{mn}} \qquad (3.32)$$

Aktivasyon işlemlerinde sigmoid fonksiyonunu kullandığımızda n sinir hücresinin çıkışı y_n aşağıdaki gibi ifade edilebilir.

$$\mathbf{y_n} = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-\mathbf{x}_n}} \quad (3.33)$$

Aktivasyon fonksiyonunun x_n ' e göre kısmi türevi ise aşağıdaki gibidir.

$$\frac{\partial \mathbf{y_n}}{\partial \mathbf{x_n}} = \mathbf{y_n}(\mathbf{1} - \mathbf{y_n}) \quad (3.34)$$

Hatanın geri yayılımında kısmı türevler ise zincir kuralına göre aşağıdaki gibi tanımlanabilir.

$$\frac{\partial \mathbf{C}}{\partial \mathbf{x_n}} = \frac{\partial \mathbf{y_n}}{\partial \mathbf{x_n}} \frac{\partial \mathbf{C}}{\partial \mathbf{y_n}} = \mathbf{y_n}(\mathbf{1} - \mathbf{y_n}) \frac{\partial \mathbf{C}}{\partial \mathbf{y_n}} \quad (3.35)$$

$$\frac{\partial \mathbf{C}}{\partial y_m} = \sum_n \frac{\partial \mathbf{x_n}}{\partial y_m} \frac{\partial \mathbf{C}}{\partial \mathbf{x_n}} = \sum_n w_{mn} \frac{\partial \mathbf{C}}{\partial \mathbf{x_n}} \quad (3.36)$$

$$\frac{\partial \mathbf{C}}{\partial w_{mn}} = \frac{\partial \mathbf{x_n}}{\partial w_{mn}} \frac{\partial \mathbf{C}}{\partial \mathbf{x_n}} \quad (3.37)$$

Hata fonksiyonun ağırlıklara göre kısmı türevinin alınmasından sonra elde edilen değerler $\frac{\partial \mathbf{C}}{\partial w_{mn}}$ eşitliğinde yerlerine konulursa aşağıdaki gibi bir eşitlik elde edilebilir.

$$\frac{\partial \mathbf{C}}{\partial w_{mn}} = \mathbf{y_m} \frac{\partial \mathbf{C}}{\partial \mathbf{x_n}} = \mathbf{y_m}\, \mathbf{y_n}(\mathbf{1} - \mathbf{y_n}) \frac{\partial \mathbf{C}}{\partial \mathbf{y_n}} \quad (3.38)$$

Hata fonksiyonu olarak ortalama hata karesi kullanıldığı varsayılırsa n sinir hücresinin hata çıktısı $\mathbf{y_n}$ olduğu için beklenen $\mathbf{o_n}$ değerine göre hata aşağıdaki gibi tanımlanabilir.

$$\mathbf{C} = \frac{\mathbf{1}}{\mathbf{2}} \sum_n (\mathbf{o_n} - \mathbf{y_n})^2 \quad (3.39)$$

Kısmi türev eşitlikleri yerine konulursa aşağıdaki gibi bir eşitlik elde edilir.

$$\frac{\partial \mathbf{C}}{\partial \mathbf{y}_n} = -(\mathbf{o}_n - \mathbf{y}_n) \tag{3.40}$$

$$\frac{\partial \mathbf{C}}{\partial \mathrm{w}_{mn}} = \mathbf{y}_m \mathbf{y}_n(\mathbf{1} - \mathbf{y}_n)\frac{\partial \mathbf{C}}{\partial \mathbf{y}_n} = -\mathbf{y}_m \mathbf{y}_n(\mathbf{1} - \mathbf{y}_n)(\mathbf{o}_n - \mathbf{y}_n) \tag{3.41}$$

Son olarak elde edilen Denklem 3.41'deki eşitlik eğim azalımı algoritması ile w_{mn} ağırlıklarının güncellenmesinde kullanılabilir.

$$\mathbf{w}_{mn} = \mathbf{w}_{mn} - \eta \frac{\partial \mathbf{C}}{\partial \mathrm{w}_{mn}} \tag{3.42}$$

$$\mathbf{w}_{mn} = \mathbf{w}_{mn} + \eta\, \mathbf{y}_m \mathbf{y}_n(\mathbf{1} - \mathbf{y}_n)(\mathbf{o}_n - \mathbf{y}_n) \tag{3.43}$$

3.3.10. Regularizayon Teknikleri

Derin öğrenme tekniklerinde kullanılan sinir ağı modelleri milyonlarca sayıda parametreye sahip olduğundan veya yeterli veri ile eğitilmediği için aşırı uyum problemi ortaya çıkabilmektedir. Bu problemin çözümü için kullanılan tekniklere regularizasyon teknikleri adı verilir. Yaygın olarak L1, L2 ve Dropout regularizasyon teknikleri kullanılmaktadır.

3.3.10.1. L1 Tekniği

L1 regularizasyon tekniğinde maliyet fonksiyonuna ağırlıkların ceza terimi eklenmektedir [46]. Büyük ağırlık değerleri cezalandırılarak sıfıra yakınsanmaktadır. Ceza terimi elde edilirken tüm ağırlıkların her birinin mutlak değeri alınır, bu değerlerin ortalaması hesaplanır ve sonuç regularizasyon sabiti (λ) ile çarpılır. Maliyet fonksiyonuna bu terimin eklenmesiyle elde edilecek eşitlik aşağıdaki gibi olacaktır.

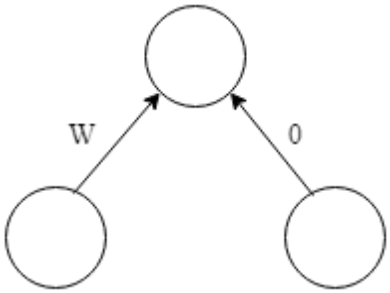
$$\mathbf{C}(\mathbf{w}) = \mathbf{C}(\mathbf{w}) + \frac{\lambda}{n}\sum_{i=0}^{n} |w_i| \quad (3.44)$$

3.3.10.2. L2 Tekniği

L1 tekniğinde olduğu gibi L2 regularizasyon tekniğinde de maliyet fonksiyonuna eklenen terim ile ağırlıkların küçültülmesi amaçlanır. Fakat ceza terimi değeri hesaplanırken tüm ağırlıkların her birinin kareleri alınır, ortalaması hesaplanır, elde edilen değer regularizasyon sabiti (λ) ile çarpılır ve sonuç ikiye bölünür. Maliyet fonksiyonu aşağıdaki gibi ifade edilebilir.

$$\mathbf{C}(\mathbf{w}) = \mathbf{C}(\mathbf{w}) + \frac{\lambda}{2n}\sum_{i=0}^{n} {w_i}^2 \quad (3.45)$$

L2 regularizasyon tekniğinde ağda gerekli olmayan ağırlıkların kullanılması engellenir. L1 tekniğinde daha fazla ağırlık sıfıra eşitlenerek verilerin ağ tarafından daha iyi temsil edilmesi sağlanabilir [47].



Şekil 3.19 Gerekli olmayan ağırlıkların kullanılmaması

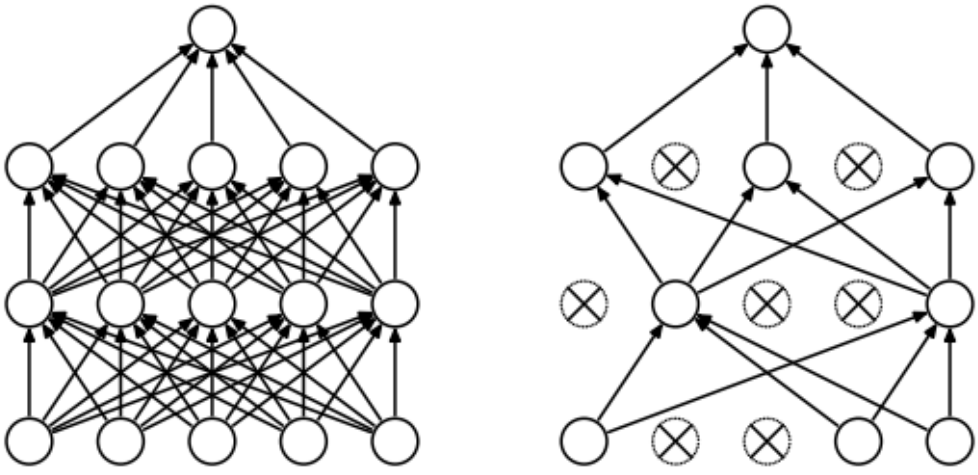
L1 ve L2 regularizasyon tekniklerinde ortak olarak ağırlık küçültme işlemi yapılsa da ağırlıkları küçültme şekli farklıdır. L2 tekniğinde ağırlıklar (w) ile orantılı olarak azaltılırken, L1 tekniğinde ağırlıklar sıfıra doğru sabit olarak azaltılır. Mutlak ağırlık (|w|) değeri büyük olduğunda L1, L2 tekniğinden daha az ağırlık azaltma işlem yapar, değer küçük olduğunda ise ters olarak L1, L2 tekniğinden daha fazla ağırlık

azalımı yapabilir. L1 tekniğinde yüksek öneme sahip ağırlıklar az değiştirilirken, önemsiz olan diğer ağırlıkların sıfıra indirgenmesi istenir [43].

3.3.10.3. Dropout

Dropout, derin öğrenme modellerinde en yaygın olarak kullanılan regularizasyon yöntemdir. Bu teknikte L1 ve L2 tekniklerinden farklı olarak, eğitim yapılırken ağ üzerindeki sinir hücrelerinin bazıları rastgele silinir. Bu şekilde sinir hücrelerinin aynı özellikleri temsil etmesi önlenmekte, her sinir hücresinin farklı özellikleri temsil etmesi sağlanarak ağın daha etkili biçimde çalışması sağlanmaktadır.

Makine öğrenmesi tekniklerinde çeşitli modellerin birleştirilmesi çoğu zaman performansı arttırmaktadır. Derin öğrenme tekniklerindeki birçok farklı modelin ayrı ayrı eğitilerek ortalamasının alınması maliyetlidir. Farklı derin ağların. her biri için optimum parametrelerin bulunması zordur ve her biri derin olmayan ağlara göre daha fazla hesaplama gücü gerektirir. Dropout tekniği birçok sinir ağı modelinin etkili bir biçimde birleştirilmesini sağlar. Görüntü ve konuşma tanıma, belge sınıflandırma ve hesaplamalı biyoloji problemlerinin çözümü için geliştirilen sinir ağlarının eğitiminde dropout regularizasyon tekniğinin kullanılması birçok veri seti üzerinde iyi sonuçlar alınmasını sağlayarak performans artışı gözlenmiştir [48].



Şekil 3.20 YSA modeli ve dropout tekniği uygulanmış hali

Bir katmandaki sinir ağlarının 0 ile 1 arasında seçilen olasılık değerine göre rastgele silinmesini sağlar. Şekil 3.20' de 0.5 olasılık değeri ile dropout tekniği kullanılarak regularize edilmiş bir ağ görülmektedir. Dropout kullanıldığında modelin doyma noktasına ulaşması daha uzun sürmektedir. Bu teknik sadece eğitim yapılırken gerçekleştirilir. Test sırasında ise silme işlemi gerçekleştirilmez. Eğitim sırasında silinmiş olan sinir hücresinin çıkış ağırlıkları olasılık değeri ile çarpılarak ileri hesap yapılır.

3.3.11. Derin Öğrenme Kütüphaneleri

Derin öğrenme uygulamalarının geliştirilebilmesi, çalışabilir duruma getirilmesi birçok konuda uzman bilgisi ya da birtakım donanım maliyetleri gerektirmektedir. Özellikle eğitim yapılırken sistem kaynaklarının iyi bir şekilde kullanılması performansı önemli ölçüde etkilemektedir. Örnek olarak MİB çekirdeklerinin paralel olarak kullanılabilmesi, MİB kümelerinin oluşturulması, GİB' in etkin kullanımı süreci hızlandırmaktadır. Böyle yapıların inşa edilmesi bilgi birimi gerektirmekte ve belirli zaman almaktadır. Derin öğrenme kütüphaneleri birçok farklı yapıyı temel alarak çalışmakta ve bundan dolayı hem eğitim sürecini kısaltmakta hem de geliştirme için zaman kazancı sağlamaktadır. Birçok derin öğrenme kütüphanesi olmasına karşın bu başlık altında popüler olan derin öğrenme kütüphaneleri açıklanmaktadır.

3.3.11.1. TensorFlow

TensorFlow, nümerik hesaplama yapılabilen, diğer makine öğrenmesi kütüphanelerine göre daha yeni olan Apache 2.0 lisanslı açık kaynaklı yazılım kütüphanesidir. Google beyin takımında yer alan araştırmacı ve mühendisler tarafından derin mimarileri desteklemek amacı ile geliştirilmiştir. Hesaplama sistemi veri akış çizgeleri kullanılarak tasarlanmıştır. Çizgelerin düğümleri matematiksel işlemleri temsil ederken, bağlantılar ise çok boyutlu dizileri temsil eden tensörlerden oluşmaktadır [35]. Python, C++, Java, Go programlama dillerinde uygulama programlama arayüzü mevcuttur. Python uygulama programlama arayüzü en eksiksiz ve en kolay kullanıma sahip olan arayüzdür [49].

TensorFlow mobil cihazlardan MİB destekli gömülü sistemlere kadar birçok sistem üzerinde çalışacak şekilde tasarlanmıştır. TensorFlow' un sağladığı esneklik sayesinde belirli bir sistemin çalışması için yapılan bir hesaplama farklı sistemler üzerinde çok az değişiklikle veya hiç değişiklik yapılmadan uygulanabilmektedir. Konuşma tanıma, bilgisayar görüsü, robotik, doğal dil işleme, coğrafik bilgi çıkarımı ve ilaç keşif hesaplaması gibi birçok alanda kullanılmaktadır. Hesaplamaların paralel olarak MİB, bir veya birden fazla GİB üzerinde yapılmasına olanak tanımaktadır [50]. GİB üzerinde paralel hesapların yapılabilmesini sağlayan CUDA platformunun desteği mevcuttur.

Tablo 3.1TenserFlow kütüphanesinin performansının karşılaştırılması

		Masaüstü MIB Iş Parçacığı Sayısı				Sunucu MIB Iş Parçacığı Sayısı					
		1	2	4	8	1	2	4	8	16	32
AlexNet-S	Caffe	1.606	0.999	**0.719**	-	1.533	1.045	**0.797**	0.850	0.903	1.124
	CNTK	3.761	1.974	**1.276**	-	3.852	2.600	1.567	1.347	**1.168**	1.579
	TF	6.525	2.936	1.749	**1.535**	5.741	4.216	2.202	1.160	**0.701**	0.962
	MXNet	2.977	2.340	2.250	**2.163**	3.518	3.203	2.926	2.828	**2.827**	2.887
	Torch	4.645	2.429	**1.424**	-	4.336	2.468	1.543	1.248	**1.090**	1.214
AlexNet-R	Caffe	3.558	2.587	**2.157**	2.963	4.270	3.514	3.381	**3.364**	4.139	4.930
	CNTK	9.956	7.263	**5.519**	6.015	9.381	6.078	4.984	**4.765**	6.256	6.199
	TF	4.535	3.225	1.911	**1.565**	6.124	4.229	2.200	1.396	1.036	**0.971**
	MXNet	13.401	12.305	12.278	**11.950**	17.994	17.128	16.764	**16.471**	17.471	17.770
	Torch	5.352	3.866	**3.162**	3.259	6.554	5.288	4.365	**3.940**	4.157	4.165

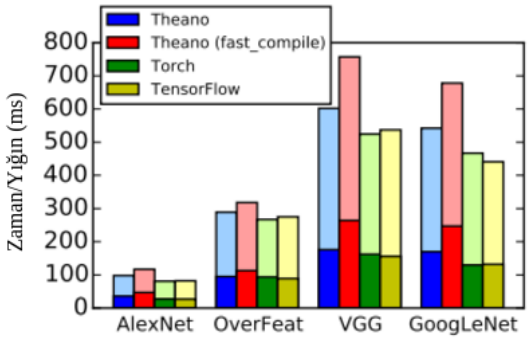
2017 yılında Shaohuai Shi ve arkadaşları tarafından yapılan bir çalışmada [51] AlexNet modelinin paralel olarak 16 MİB çekirdeği ile 16 ve 128 boyutunda yığınlar halinde popüler kütüphaneler olan Caffe, CNTK, TensorFlow, MXNet, Torch kullanılarak eğitimi yapılmıştır. Eğitim yapılırken her kütüphane için yığın başına saniye cinsinden süreler hesaplanmış ve bu süreler arasında karşılaştırma yapılmıştır. Bu çalışmada AlexNet üzerinde 16 çekirdekli MİB paralel olarak kullanılmıştır. Girdiler 16 boyutlu yığınlar halinde paralel olarak sınıflandırılmaya çalışılmıştır. TensorFlow kütüphanesinin diğer kütüphanelere göre en kısa sürede sınıflandırma yaptığı belirlenmiştir. 128 boyutlu yığın kullanıldığında ise en iyi sonuç 32 iş parçacığı kullanılarak paralel olarak yapılan teste alınmıştır.

3.3.11.2. Theano

Theano, çok boyutlu dizileri içeren matematiksel ifadelerin tanımlanmasına olanak tanıyan Montreal Üniversitesinde bir gurup araştırmacı tarafından geliştirilen BSD lisanslı açık kaynaklı Python kütüphanesidir. Bu kütüphanede birçok nümerik hesaplamanın etkin olarak yapılmasını sağlayan Numpy kütüphanesinin söz dizilim kolaylığı, optimize edilmiş doğal makine dilinin hızı ile birleştirilmiştir.

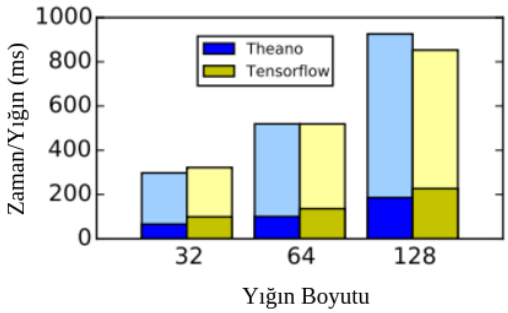
Numpy veri tipi olarak n boyutlu dizinin tanımlanmasını sağlar ve diziler üzerinde yeniden boyutlandırma, indeksleme gibi temel işlemlerin bir komut ile yapılmasını sağlayan fonksiyonlar içerir. Bu fonksiyonlar Python programları içerisinde kullanılmak üzere C programlama dili ile yazılmıştır. Bu kütüphane MİB ve GİB üzerinde hesaplama yapılmasına izin vermektedir. MİB üzerinde yapılan hesaplamalar için CUDA desteği bulunmaktadır. Hesaplama gerçekleştirilmeden önce Theano' da matematiksel ifadeler C++ veya CUDA(işlem GPU üzerinde ise) koduna çevrilir ve dinamik olarak yüklenen Python modülleri halinde derleme işlemi yapılır [52]. Matematiksel ifadeleri optimize ederek yinelenen veya gereksiz hesaplamaların yapılmasını engeller. Bu kütüphanede x – x ifadesinin yerine 0 değerinin atanması bu iyileştirmelere örnek olarak verilebilir [53].

Theano' nun üzerine inşa edilen makine öğrenmesi ve derin öğrenme tekniklerinin uygulanabilmesini sağlayan, yüksek seviyeli geliştirme yapılmasına izin veren çeşitli kütüphaneler bulunmaktadır. Derin öğrenme modellerinin mimarilerinin ve öğrenme algoritmalarının tanımlanmasını kolaylaştırma amacı ile geliştirilen Pylearn2 ve Keras kütüphaneleri örnek olarak verilebilir. TensorFlow, Torch Theano kullanılarak AlexNet, VGG, GoogLeNet derin öğrenme modelleri üzerinde ileri ve geri hesaplamalar yapılmış ve her yığın başına düşün milisaniye cinsinden süre hesaplaması yapılarak, kütüphanelerin performansı karşılaştırılmıştır. Test sonucunda Theano, Theano' nun hızlı derlenmiş sürümü, TensorFlow ve Torch' tan her yığın için az farkla uzun hesaplama süresine ihtiyaç duyduğu gözlemlenmiştir. Bu kütüphaneler için elde edilen sonuçlar Şekil 3.21' de yer almaktadır. Sonuçları gösteren grafikte koyu renkler ileri hesabı temsil ederken, açık renkler geri hesabı temsil etmektedir [54].



Şekil 3.21 Üç farklı kütüphane ile yapılan test sonucu

Aynı çalışmada video altyazısı oluşturmak için geliştirilen derin bir ağ üzerinde yığın boyutuna göre TensorFlow ve Theano kütüphaneleri arasında milisaniye cinsinden performans karşılaştırması yapılmıştır. İleri hesaplamalarda Theano' un, geri hesaplamada ise TensorFlow' un daha iyi sonuç verdiği gözlemlenmiştir. İleri ve geri hesap için toplam süre ele alındığında ise küçük yığın boyutu seçildiğinde(Yığın boyutu<64) Theano' un daha iyi sonuç verdiği gözlenirken, büyük yığın boyutlarında TensorFlow' un daha iyi performans gösterdiği gözlemlenmiştir. Şekil 3.22'de iki kütüphaneye ait test sonuçları görülmektedir.



Şekil 3.22 Yığın boyutuna göre yapılan test sonucu

3.3.11.3. Torch

Torch, basit betik diline sahip, cebirsel işlemlerin hızlı bir şekilde yapılmasını sağlayan, açık kaynaklı bilimsel hesaplama kütüphanesidir. Derin öğrenme alanındaki araştırmalarda sıklıkla kullanılır. MİB ve CUDA desteği ile GİB üzerinde paralel

hesaplama yapılmasına izin vermektedir [55]. Torch, diğer popüler kütüphanelerden farklı olarak Lua programlama dili ile geliştirme yapılmaktadır. Torch' un amacı esnek bir yapı inşası sağlamak ve öğrenen makinelerin eğitimini yapmaktır. Esneklik son derece hafif betik dili olan Lua ile sağlanmaktadır. Bunun yanında Lua en hızlı yorumlanan dildir.

Lua ile kolayca bir C uygulamasın içerisinde gömülü olarak kullanılabilecek şekilde tasarlanmıştır. Değerlerin C diline ve C dilinden iletimini sağlayan sanal yığına dayalı etkili bir C uygulama programlama uygulama programlama ara yüzü sunar. Lua' nın ana veri yapısı tablodur. İlişkisel bir dizi türü olan tablolar sadece sayı değerleri ile değil metin değerler ile de indekslenebilmektedir. Belirli boyutları olmayıp dinamik olarak boyutlandırılabilirler. Lua oyun programcıları tarafından iyi bilinse de birçok derin öğrenme kütüphanesinde kullanılan Python programlama diline göre daha az kullanılmaktadır. Python, çok sayıda kütüphaneye sahiptir. Lua' da çok sayıda kütüphane olmamasına rağmen sağladığı ara yüz sayesinde birkaç saatlik çalışma ile C veya C++ kütüphanesi Lua kütüphanesi olarak kullanılabilmektedir. LuaJIT adı verilen derleyici sayesinde birçok saf Lua dili ile yazılan kod C' ye yakın bir hızda çalışabilmektedir. Lua' nın diğer önemli özelliği ise gömülü sistemler üzerinde uygulanabilmesidir [56].

2016 Bosch araştırma ve teknoloji merkezi tarafından yapılan bir çalışmada [57] MİB üzerinde paralel olarak Caffe, Neon, Torch, GİB üzerinde Caffe, Theano, Torch kütüphaneleri kullanılmıştır. GİB üzerinde yapılan testlerde kütüphanelere ek olarak derin öğrenme için Nvidia tarafından geliştirilen yazılım geliştirme kiti CuDNN (Cuda Deep Neural Network Library) kullanılmıştır. Testlerde AlexNet modeli üzerinde ileri ve geri hesaplama yapılmış ve milisaniye cinsinden hesaplama süreleri karşılaştırılmıştır. MİB ve GİB üzerinde 256 yığın kullanılarak yapılan testlerin ikisinde de en iyi sonuç veren Torch' tur. Torch, MİB üzerinde 12 iş parçacığı ile paralel olarak yapılan işlemi bu testteki en yakın rakibi olan Caffe' ye göre yaklaşık 17 milisaniye önce bitirirken, geri hesabı Caffe' ye göre yaklaşık 35 milisaniye önce bitirmiştir. GİB üzerindeki sonuçlarda ise yine en yakın sonuç veren Caffe' ye göre ileri hesabı yaklaşık 19 milisaniye farkla, geri hesabı ise 32 milisaniye farkla tamamlamıştır.

	Kütüphane	Eğim Hesabı (ms)	İleri Hesap (ms)
MİB	Caffe (12 threads)	43 152	19 817
	Neon (1 thread)*†	100 987	28 828
	Torch (6 threads)*	11 977	4383
	Torch (12 threads)*	**8421**	**2746**
GİB	Caffe + cuDNN	422.4	111.7
	Theano + cuDNN	529.8	162.8
	Theano + cuconv	684.9	156.1
	Torch + cuDNN	**390.2**	**92.5**

Şekil 3.23 CPU ve GPU üzerinde ileri ve geri hesabın performansları

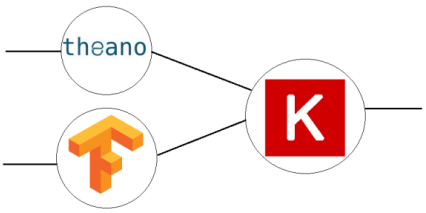
3.3.11.4. Caffe

Caffe BSD lisanslı açık kaynaklı bir C++ kütüphanesidir. Bu yönüyle araştırmacılara temiz ve istendiğinde değiştirilebilir bir ortam sunmaktadır. Bu kütüphane Berkeley Görü ve Öğrenme Merkezi(BVLC) tarafından geliştirilmiştir. Ara yüzler sayesinde Python ve Matlab ile geliştirme yapmaya olanak sağlar. MİB ve GİB üzerinde paralel geliştirme yapılabilmektedir. Birçok akademik çalışmada ve endüstriyel uygulamalarda kullanılmaktadır.

Bu kütüphane kullanılarak genel amaçlı KSA ve diğer derin modellerin etkili bir biçimde tanımlanabilir ve modellerin eğitimleri yapılabilmektedir. Bu kütüphane CUDA destekli tek K40 veya Titan GİB üzerinde yaklaşık 2.5 milisaniyede bir günde 40 milyonun üzerinde görüntü işleme yeteneği ile endüstri ve araştırma deneylerinin ihtiyaçlarını karşılamaktadır. Görüntü sınıflandırma, özellik çıkarımı gibi birçok uygulamanın örneklerini içeren detaylı olarak hazırlanmış doküman ile eğitim, test etme, modelin üzerinde hassas ayarların yapılması ve modelin dağıtımı için eksiksiz bir araç seti sunmaktadır. Ayrıca algoritmaların hızlı bir şekilde uygulanmasını sağlayarak, endüstri dağıtımlarının kısa bir zaman içerisinde yapılmasına olanak sağlar [58]. Geliştirilen bir öğrenme sisteminde tasarlanan modeller ve iyileştirme tanımlamaları, çok fazla kod yazmadan yapılandırma dosyalarıyla oluşturulabilmektedir.

3.3.11.5. Keras

Keras, derin öğrenme modellerinin kolayca tanımlanmasını sağlayan François Chollet tarafından geliştirilen yüksek seviyeli sinir ağı uygulama programlama ara yüzüdür. Python programlama dili ile yazılmıştır. TensorFlow, CNTK veya Theano kütüphanelerinin üzerinde çalışabilmektedir. Şekil 3.24' te Keras' ın işlev yapısı görülmektedir.



Şekil 3.24 Keras' ın işlev yapısı

Kullanıcı dostu, modüler ve genişletilebilir bir yapı sağladığı için deneylerin hızlı bir şekilde yapılmasına olanak tanımaktadır. Konvolüzasyonel ağların ve tekrarlı ağların veya ikisinin birlikte tanımlanmasına izin verir. MİB ve GİB üzerinde çalışabilmektedir. Sağladığı modülerlik sayesinde tasarlanan model, birbirine bağlanan, birbirinden bağımsız, isteğe bağlı olarak yapılandırılabilir modül dizisinden veya çizgesinden oluşmaktadır. Sinir ağlarının katmanları, maliyet fonksiyonları, iyileştirme yöntemleri, aktivasyon fonksiyonları, regularizasyon şemaları yeni model tasarlanırken birbirine bağlanabilen bağımsız modüllerdir. Modüllerin birbirine bağlanması ile genişletilebilir bir yapı sağlamaktadır [59]. Bunun yanında çeşitli örnekler barındıran detaylı hazırlanmış doküman ve Python paket yöneticisi ile sağlanan kurulum kolaylığı sayesinde kısa bir süre içerisinde geliştirme yapmaya izin vermektedir. Bu çalışmada bu esnek yapısından dolayı Keras kütüphanesi kullanılmıştır.

BÖLÜM 4

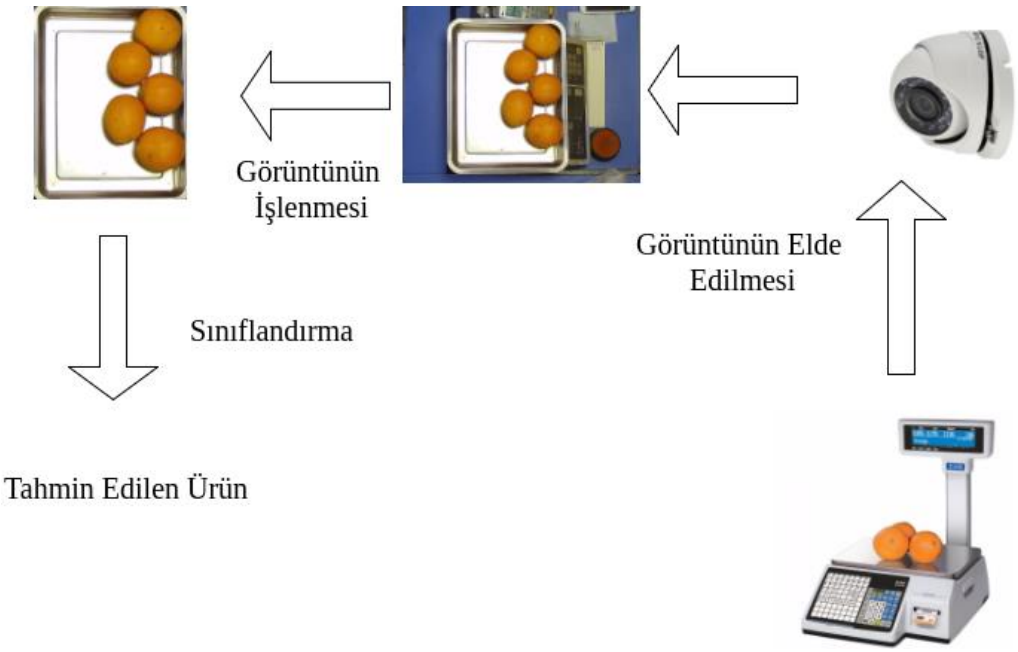
DENEYLER VE SONUÇLAR

4.1. Geliştirilen Derin Öğrenme Sistemi

Bu çalışmada, perakende satış noktalarında kullanılan barkotlu terazilere alternatif olarak kullanılabilecek, insan görüsünü derin öğrenme algoritmasıyla destekleyen bir sistem geliştirilmiştir. Çoğu zaman barkotlu terazilerin iş profesyonelleri tarafından kullanılması gerekmektedir. Aynı zamanda, iş profesyonellerinin tartı ile satılması gereken ürünlerin bulunduğu reyonlarda sürekli bulunmaları zorunludur. Geliştirilen sistem hızlı bir biçimde tartılan ürünün sınıfın otomatik olarak belirleyerek satış sürecinin hatasız ve bir iş profesyoneline ihtiyaç duymadan gerçekleşmesine imkân sağlayacaktır.

Yapılan çalışmada KSA kullanılarak çok sınıflı sınıflandırma işlemi yapabilen derin öğrenme modelleri oluşturulmuştur. Oluşturulan modellerin özgün etiketli veriler ile eğitimi yapılmıştır. Özgün veriler, satın alma sürecinde, ortamda bulunan kamera vasıtası ile elde edilen ve iş profesyoneli tarafından etiketlenen verilerinden oluşmaktadır. Farklı modeller, farklı veri setleri ile eğitilerek sonuçlar karşılaştırılmış ve en iyi model belirlenmeye çalışılmıştır.

Modellerin kodlama, geliştirme, eğitim ve test işlemleri Nvidia Jetson TX2 modülü üzerinde NVIDIA Pascal, 256 CUDA çekirdekli GİB üzerinde gerçekleştirilmiştir. Kit üzerinde 8 GB RAM bulunmaktadır. Geliştirme için Python programlama dili seçilmiştir. Derin öğrenme kütüphanesi olarak arka planında TensorFlow kütüphanesi ile birlikte çalışan Keras kullanılmıştır. Veri setleri oluşturulurken 2 mega piksel kalitesinde fotoğraf çekebilen IP kamera kullanılmıştır. Elde edilen görüntülerin sorunsuz bir biçimde sabit diskte kayıt edilebilmesi için POE (Power Over Ethernet) özellikli gigabit bir ağ anahtarı kullanılmıştır. Şekil 4.1' de sistemin genel çalışma modeli yer almaktadır.



Şekil 4.1 Sistemin çalışma modeli

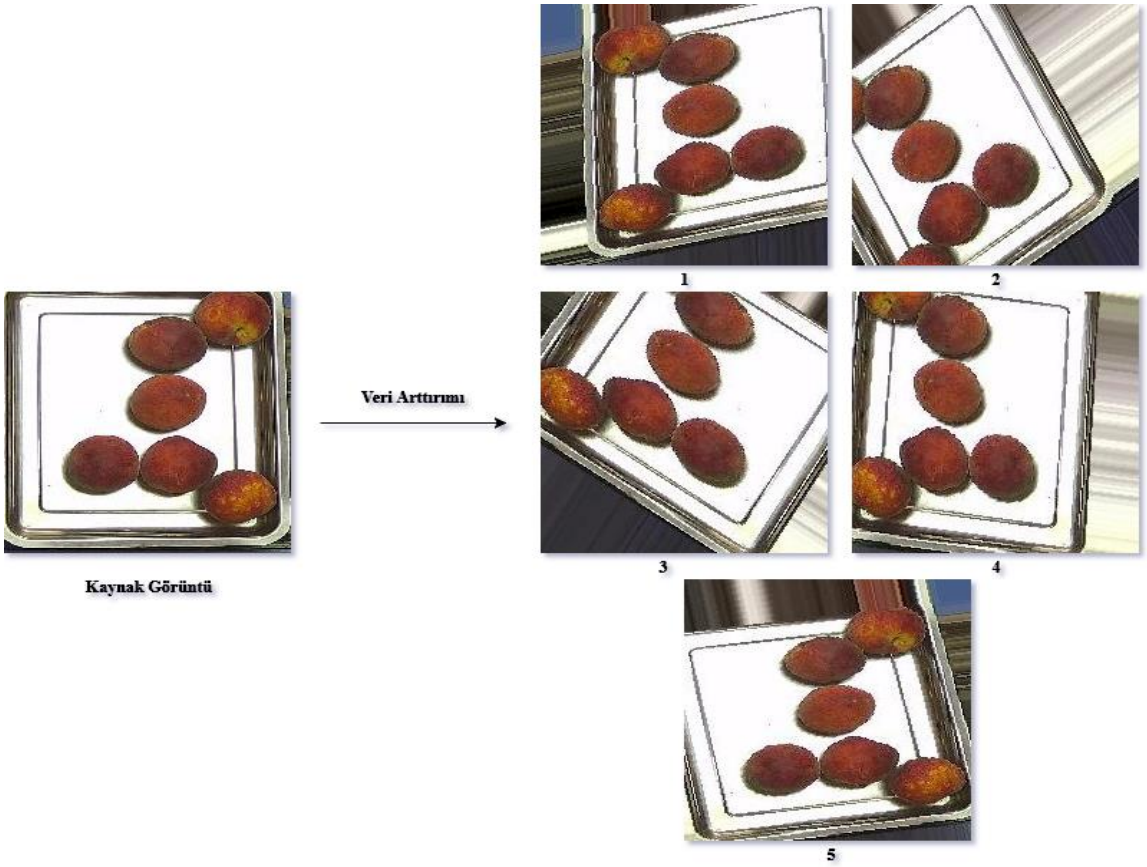
4.2. Verilerin Hazırlanması

Verilerin eğitim ve test süreçlerine hazır hale getirilmesi için yapılan işlemler genel olarak üç aşamadan oluşmaktadır. İlk aşama canlı ortamdan toplanan veriler sınıflarına göre ayrıştırılır ve veri üstünde istenilen bölgelere odaklanılmaktadır. İkinci aşama verilerin az sayıda olduğu durumlarda uygulanmaktadır. Son aşama ise görüntülerin piksel matrisi değerlerinin normalizasyon işlemidir.

İlk aşamada yapılan verilerin sınıflara göre ayrıştırılması işleminde Python programlama dili ile yazılan program sayesinde canlı otomasyondan alınan verilerin etiketlerine göre sınıfları belirlenmiştir. 1920x1080 boyutundaki tüm görüntüler terazinin olduğu bölümden otomatik olarak kırpılmış ve yeniden boyutlandırılmıştır. Otomasyondan elde edilen görüntü verileri sabit bir kameradan elde edilen verilerden oluştuğu için terazinin olduğu bölge manuel olarak belirlenmiştir ve her bir veri üzerinde kırpma işlemi uygulanmıştır. Kırpılan görüntüler 100x100 ve 200x200 boyutunda olacak şekilde yeniden boyutlandırılmıştır. İlk aşamada yapılan işlemler aşağıdaki gibi sıralanmıştır.

1- Tüm görüntü verilerinin etiketlerini incelenmesi ve etiket adına göre klasörleri oluşturulması
2- Tüm görüntü verilerinin içerisinden bir adet görüntü girdisinin alınması
3- Görüntünün terazinin olduğu bölgeden kırpılması
4- Kırpılan bölgenin 100x100 ve 200x200 boyutunda yeniden boyutlandırılması
5- Etiketlerden sınıfların belirlenmesi ve sınıfa ait klasörün içerisine dâhil edilmesi
6- 2, 3, 4, 5 numaralı işlemlerin sırası ile görüntü girdileri bitene kadar tekrar edilmesi

İkinci aşama yapılan verilerin arttırılması işlemi sadece verilerin az sayıda olduğu durumda, modelin başarısını arttırmak ve aşırı uyum probleminin giderilmesini sağlamak için kullanılmaktadır. Kaynak görüntü kullanılarak rastgele döndürme, aşağıya ve yukarıya kaydırma, yakınlaştırma işlemleri uygulanarak bir görüntüden çok sayıda örnek görüntü oluşturulmuştur. Şekil 4.2' de kaynak görüntüden beş farklı görüntü elde edilmiştir.



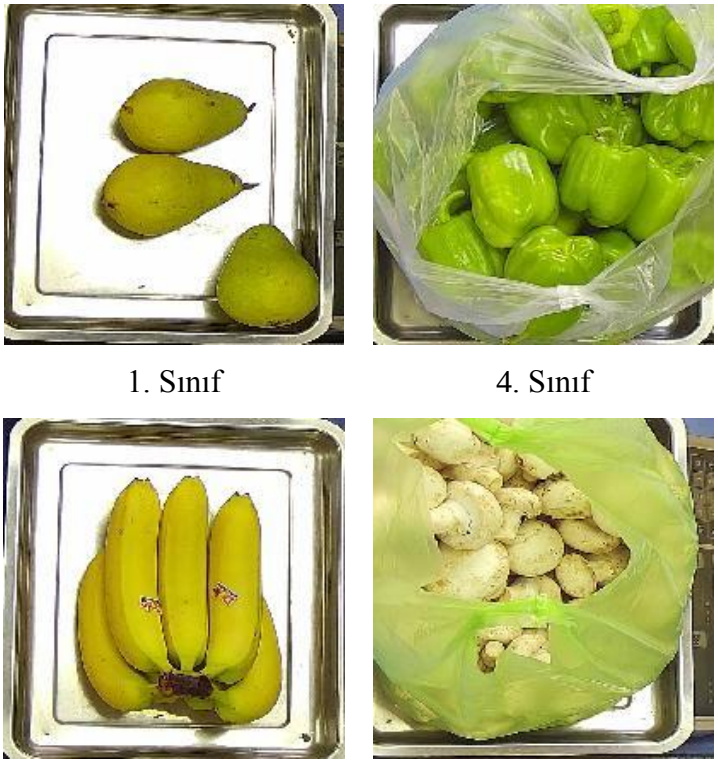
Şekil 4.2 Kaynak görüntüden veri arttırımı

Son aşamada her bir veriye ait sayı değerlerinin en büyük ve en küçük noktalarının değiştirilmesiyle değerlerin farklı bir aralıkta ifade edilmesi kuralına dayanan minimum-maksimum normalizasyon yöntemi kullanılmıştır. Bu yönteme göre 0 ile 255 arasında piksel değerleri alabilen 8 bit renkli görüntü verileri 0 ile 1 arasında değerler alabilecek şekilde normalize edilmiştir. Farklı boyutta birçok matris işlemi yapıldığı için büyük veriler ile çalışmak yerine 0 ile 1 arasında bulunan küçük veriler ile çalışarak tüm verilerin benzer ortalama, standart sapmaya sahip olması sağlanmakta ve eğitim süreci hızlandırılmaktadır.

4.3. Veri Setleri

Derin öğrenme uygulamalarında veri setlerinin seçimi oldukça önemlidir. Veri setleri eğitim, doğrulama ve test veri setleri olarak üçe ayrılmıştır. Bu veri setlerinin seçimi deneylerin önemli bir bölümünü oluşturmaktadır.

Bu çalışmada yapılan deneyler temel olarak iki veri seti oluşturulmuştur. Bu veri setleri farklı sınıf sayıları, farklı boyutlar, farklı eğitim veri sayısı, farklı doğrulama veri sayısı, farklı test veri sayıları kullanılarak test edilmiştir. Şekil 4.3' te kullanılan verilere ait görüntüler yer almaktadır. Tüm veri setleri meyve ve sebzelere ait görüntüler içermektedir.



Şekil 4.3 Terazi üzerindeki ürünlerin görüntüleri

İlk veri seti 20 adet sınıftan oluşmaktadır. Tablo 4.1' de veri setinde kullanılan sınıfların isimleri yer almaktadır.

Tablo 4.1 İlk veri setine ait sınıflar

Numara	Sınıf Adı
1.	Armut
2.	Baklagiller
3.	Çarliston ve Sivri Biber
4.	Dolma Biber
5.	Domates
6.	Erik
7.	Elma
8.	Havuç
9.	Kabak
10.	Kavun
11.	Kayısı
12.	Kiraz
13.	Karpuz
14.	Muz
15.	Mantar
16.	Patlıcan
17.	Patates
18.	Portakal
19.	Soğan
20.	Salatalık

Tablo 4.1' de yer alan ilk veri setinde her bir sınıfa ait 126 görüntü eğitim ve doğrulama için kullanılmıştır. Dolayısıyla eğitim ve doğrulama veri seti 2520 adet görüntüden oluşmaktadır. Test veri setinde ise her sınıfa ait 14 adet görüntü kullanılmıştır. Kullanılan görüntüler 200x200 boyutunda matrislerden oluşmaktadır.

İkinci veri seti ilk veri setinin üzerine verilerinin eklenmesi, bazı sınıflarda az sayıda bulunan verilerin ve sınıf sayılarının arttırılması ile oluşturulmuştur. İkinci veri seti 31 adet sınıftan oluşmaktadır. Tablo 4.2' de bu veri setine ait sınıflar listelenmiştir. İkinci veri setinde her sınıfa ait 1100 adet eğitim ve doğrulama verisi bulunmaktadır. Dolayısıyla eğitim ve doğrulama veri seti 34100 adet görüntüden oluşmaktadır. Test veri setinde ise her sınıfa ait 100 adet görüntü kullanılmıştır.

Tablo 4.2 İkinci veri setine ait sınıflar

Numara	Sınıf Adı	Numara	Sınıf Adı
1.	Armut	17.	Ispanak
2.	Bakla	18.	Kabak
3.	Dolma Biber	19.	Kavun
4.	Bezelye	20.	Kayısı
5.	Kırmızı Biber	21.	Kiraz
6.	Meksika Biber	22.	Kırmızı Lahana
7.	Sivri Biber	23.	Muz
8.	Çarliston Biber	24.	Mantar
9.	Yeşil Köy Biberi	25.	Patlıcan
10.	Domates	26.	Patates
11.	Elma	27.	Portakal
12.	Erik	28.	Salatalık
13.	Anjelik Erik	29.	Soğan
14.	Fasülye	30.	Çilek
15.	Havuç	31.	Şeftali
16.	Ispanak		

İkinci veri seti üzerinde yapılan testlerde 100x100 ve 200x200 olmak üzere iki farklı boyutta matrisler kullanılmıştır. Veri setleri, sınıf sayıları, eğitim, doğrulama, test veri sayıları ve içerdikleri görüntü verilerinin boyutları Tablo 4.3'teki gibidir.

Tablo 4.3 Kullanılan veri setleri ve özellikleri

Veri Seti	Sınıf Sayısı	Eğitim Verisi	Doğrulama Verisi	Test Verisi	Boyut
1.	20	2268	252	280	200x200
2.	31	31000	3100	3100	200x200
3.	31	31000	3100	3100	100x100

4.4. Performans Ölçütleri

Eğitilen modeller test edilirken doğruluk değerinin yanında karmaşıklık matrisleri kullanılarak kesinlik(precision), anma değeri(recall) ve F1 skor değerleri hesaplanmıştır. Şekil 4.4'te görüldüğü üzere iki sınıfa ait karmaşıklık matrisi yer almaktadır.

		Tahmin Değerleri		Toplam
		A	B	
Gerçek Değerler	A	DP (Doğru Pozitif)	YN (Yanlış Negatif)	N_A
	B	YP (Yanlış Pozitif)	DN (Doğru Negatif)	N_B
Toplam		N'_A	N'_B	N

Şekil 4.4 Karmaşıklık matrisi

Bu matris üzerinde N'_A, N'_B değerleri tahminlerin toplam değerini, N_A ve N_B değerleri sınıf bazında toplam örnek sayılarını ifade etmektedir. N değeri ise tüm örneklerin toplamını ifade eder. DP doğru tahmin edilen A sınıf sayısını, DN doğru tahmin edilen B sınıf sayısını, YN gerçekte A sınıfı olan fakat B sınıfı olarak tahmin edilen örneklerin sayısını, YP gerçekte B sınıfı olan fakat A sınıfı olarak tahmin edilen örneklerin sayısını belirtmektedir. Matris üzerinde doğruluk hesaplaması yapılırken doğru tahmin edilen değerler toplam örnek sayısına bölünür. 0, 1 veya 0 ve 1 arasında bir değer elde edilir.

$$\textbf{Doğruluk} = \frac{\textbf{DP} + \textbf{DN}}{\textbf{N}} \qquad (4.1)$$

Kesinlik değeri, bir sınıfına doğru tahmin edilen örnek sayısının, toplam tahmin sayısına oranı ile hesaplanmaktadır. A sınıfının kesinlik değerini elde ederken doğru tahminlerin sayısı, doğru tahminlerin sayısı ile gerçek değeri B olan fakat A olarak yanlış tahmin edilenlerin örnek sayılarının toplamına bölünerek bulunur. A ve B sınıflarının her ikisinin de A sınıfı olarak tahmin edildiği durumun başarısını göstermektedir.

$$\textbf{Kesinlik} = \frac{\textbf{DP}}{\textbf{N}'_{\boldsymbol{A}}} = \frac{\textbf{DP}}{\textbf{DP} + \textbf{YP}} \qquad (4.2)$$

Anma değeri, bir sınıfa ait doğru olarak sınıflandırılan örnek sayısının, doğru ve yanlış sınıflandırılan toplam örnek sayısına bölünmesi ile elde edilir. A sınıfının anma değeri elde edilirken doğru tahminlerin sayısının, doğru tahminlerin sayısı ve gerçek değeri A olan fakat B olarak adlandırılmış yanlış tahminlerin örnek sayılarının toplamına bölünmesi ile elde edilir. Denklem 4.3' te görüldüğü gibi A sınıfının ne kadar başarılı bir şekilde tahmin edildiğini belirtmektedir.

$$\textbf{Anma Değeri}(\textbf{AD}) = \frac{\textbf{DP}}{\textbf{N}_{\boldsymbol{A}}} = \frac{\textbf{DP}}{\textbf{DP} + \textbf{YN}} \qquad (4.3)$$

F1 skoru kesinlik ve anma değerinin harmonik ortalamasının hesaplanması ile bulunur. Harmonik ortalama sayesinde kesinlik ve anma değeri arasında büyük bir fark olduğunda F1 değeri küçük olan değere yakın bir değer alacaktır.

$$\textbf{F1 Skoru} = \frac{\mathbf{2 * Kesinlik * AD}}{\mathbf{Kesinlik + AD}} \quad (4.4)$$

4.5. Deneysel Sonuçlar

Bu bölümde farklı modellerle, farklı veri setleri ile yapılan testler incelenmektedir. Tüm deney modelleri KSA tipindedir ve çok sınıflı sınıflandırma problemini çözmek için tasarlanmışlardır. Maliyet fonksiyonu olarak Kategorik Çapraz Entropi kullanılırken, bu fonksiyonu minimize etmek için stokastik eğim azalımı yöntemi tercih edilmiştir. Çıkış katmanı hariç tüm katmanlarda Relu aktivasyon fonksiyonu kullanılmıştır. Son katmanda olasılıkların elde edilmesini sağlamak için Softmax fonksiyonu kullanılmıştır. Temelde 4 farklı test yapılmıştır.

Tablo 4.4 Testler ve özellikleri

Test No	Karşılaştırılan Model Sayısı	Kısıtları
Test 1	2	Farklı veri setleri ile eğitim yapılması
Test 2	3	Tam bağlı katmanların nöron sayılarının arttırılması
Test 3	3	100x100 görüntü girdileri ile konvolüzasyon katmanı sayıları farklı modellerin testi
Test 4	3	200x200 görüntü girdileri ile konvolüzasyon katmanı sayıları farklı modellerin testi

4.5.1. Test 1

Test 1 için ilk tasarlanan model beş konvolüzasyon, üç tam bağlı katmandan oluşmaktadır. Karşılaştırma yapmak için bu modelin çıkış nöronlarının sayısı değiştirilerek farklı bir sürümü oluşturulmuş ve iki farklı veri seti ile eğitim yapılmıştır. Test 1' de kullanılan modellerin konvolüzasyon katmanı, tam bağlı katman ve parametre sayıları Tablo 4.5' te görüldüğü gibidir.

Tablo 4.5 Test 1'de kullanılan modellerinin özellikleri

	Konvolüzasyon Katmanı Sayısı	Havuzlama Katmanı Sayısı	Tam Bağlı Katman Sayısı	Parametre Sayısı
Model 1	5	3	3	8.888,340
Model 2	5	3	3	8.899,615

Model 1' in katmanlara göre parametre sayıları ve çıkış şekillerini içeren model yapısı Tablo 4.6' da yer almaktadır. Tablo 4.7' de Model 2' nin yapısı yer almaktadır. İki modelin de konvolüzasyon ve tam bağlı katman sayıları aynı olsa da iki numaralı veri setinde sınıf sayısı farklı olduğu için çıkış katmanındaki nöron sayısı, parametre sayısı ve dolayısıyla toplam parametre sayısı fazladır.

Tablo 4.6 Model 1’ in yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 50, 50, 64)	9472
1. Maksimum Havuzlama	(x, 25, 25, 64)	0
2. Konvolüzasyon	(x, 25, 25, 128)	204928
2. Maksimum Havuzlama	(x, 13, 13, 128)	0
3. Konvolüzasyon	(x, 13, 13, 256)	295168
4. Konvolüzasyon	(x, 13, 13, 256)	590080
5. Konvolüzasyon	(x, 13, 13, 128)	295040
3. Maksimum Havuzlama	(x, 7, 7, 128)	0
Düzleştirme Katmanı	(x, 6272)	0
1. Tam Bağlı Katman	(x, 1024)	6423552
1. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 1024)	1049600
2. Dropout(%50)	(x, 1024)	0
3. Tam Bağlı Katman	(x, 20)	20500

Tablo 4.7 Model 2' nin yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 50, 50, 64)	9472
1. Maksimum Havuzlama	(x, 25, 25, 64)	0
2. Konvolüzasyon	(x, 25, 25, 128)	204928
2. Maksimum Havuzlama	(x, 13, 13, 128)	0
3. Konvolüzasyon	(x, 13, 13, 256)	295168
4. Konvolüzasyon	(x, 13, 13, 256)	590080
5. Konvolüzasyon	(x, 13, 13, 128)	295040
3. Maksimum Havuzlama	(x, 7, 7, 128)	0
Düzleştirme Katmanı	(x, 6272)	0
1. Tam Bağlı Katman	(x, 1024)	6423552
1. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 1024)	1049600
2. Dropout(%50)	(x, 1024)	0
3. Tam Bağlı Katman	(x, 31)	31775

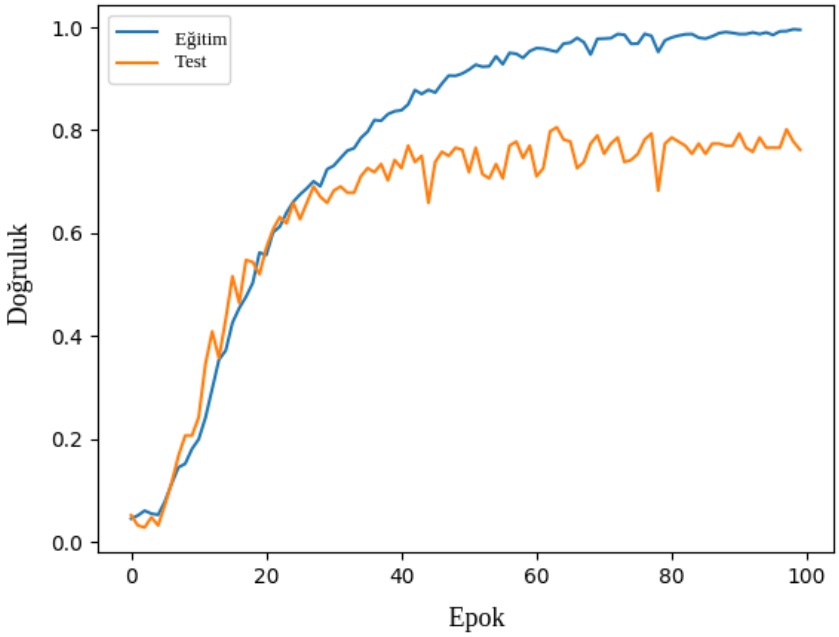
Tablo 4.3' te gösterilen veri setlerinden ilk eğitim bir numaralı az sayıda veri içeren veri setiyle, ikinci eğitim ise daha fazla sayıda veri içeren iki numaralı veri seti ile gerçekleştirilmiştir.

Tablo 4.8 Test 1' deki modellerin performansı

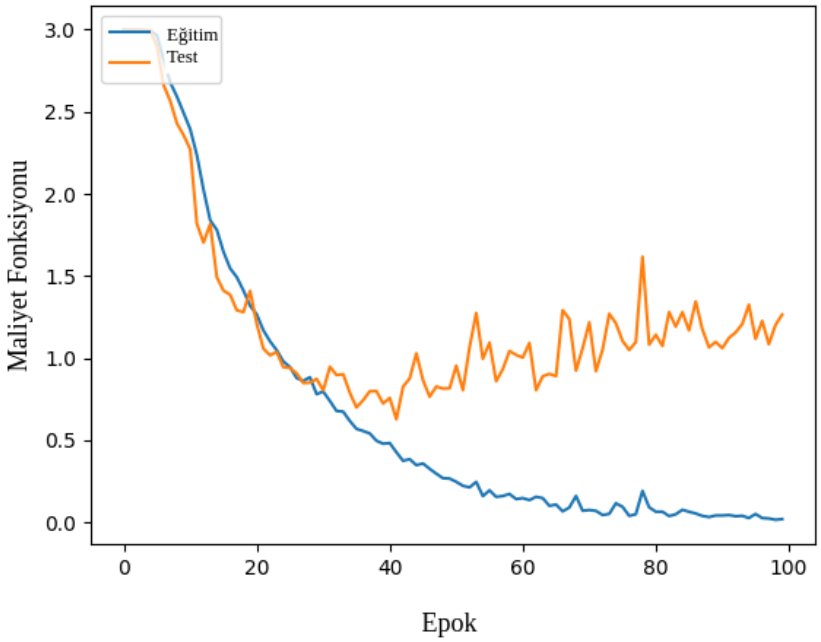
	Veri Seti	Görüntü Boyutu	Epok Sayısı	Başarı(%)	Kesinlik	Anma Değeri	F1 Skoru
Model 1	1	200x200	100	78.21	0.80	0.78	0.78
Model 2	2	200x200	100	91.26	0.92	0.91	0.91

Yapılan eğitimler sonucunda sınıf sayısı artmasına rağmen veri sayısı ve çeşitliliği arttığı için başarı oranı da artmıştır. Model 1 ve bu model ile çok benzer bir yapıya sahip olan Model 2 üzerinde aynı epok sayısı ile farklı veri seti kullanarak yapılan eğitimde başarı %78.21'den %91.26' ya çıkarılmıştır. Bunun yanında kesinlik değeri 0.80' den 0.92' ye, anma değeri ve f1 skoru 0.78' den 0.91 değerine çıkartılarak daha başarılı bir model oluşturulmuştur.

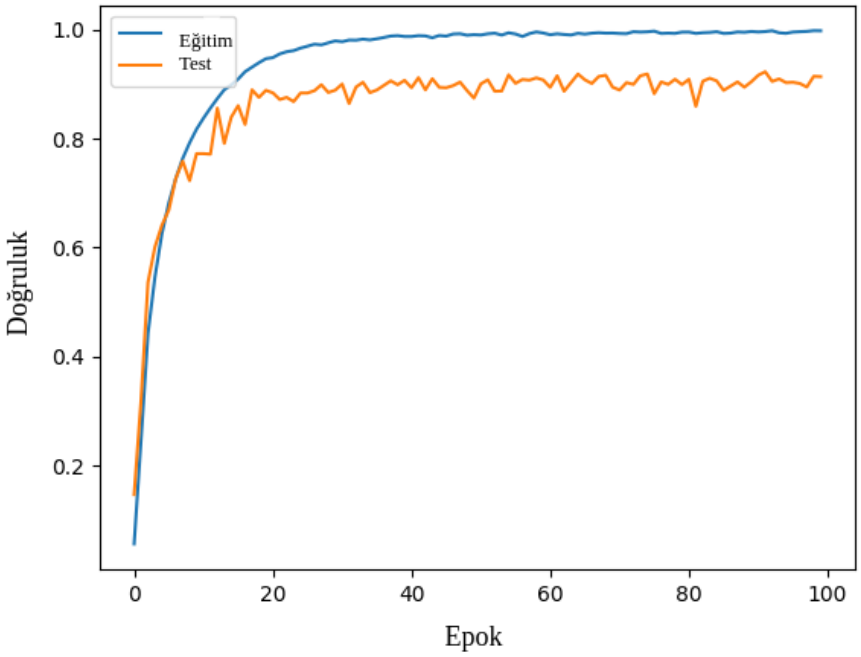
Model 1 ve Model 2' nin doğruluklarının epok sayılarına göre değişimini gösteren grafikler sırasıyla Şekil 4.5, Şekil 4.7' de görülmektedir. Model 1 için Şekil 4.6, Model 2 için ise Şekil 4.8' de ise maliyet fonksiyonlarının minimize edilmesini temsil eden grafikler yer almaktadır.



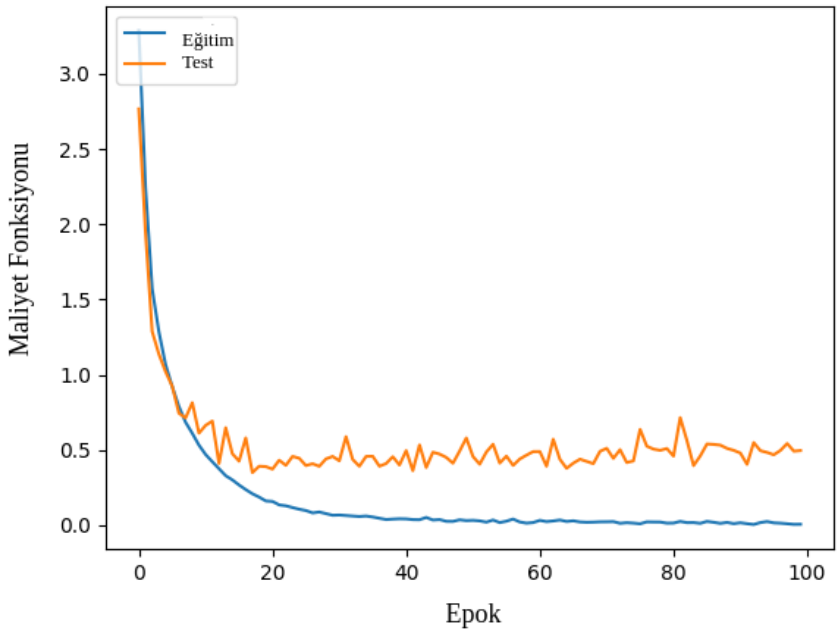
Şekil 4.5 Model 1’ in doğruluk değişim grafiği



Şekil 4.6 Model 1’ in maliyet fonksiyonu değişim grafiği



Şekil 4.7 Model 2’ nin doğruluk değişim grafiği



Şekil 4.8 Model 2’ nin maliyet fonksiyonu değişim grafiği

Şekil 4.5’ te Model 1’ e ait doğruluğu ve maliyet fonksiyonunun değişimini gösteren grafikte yaklaşık 30 epoktan sonra eğitim ve doğrulama verileri arasındaki fark artmaya başlamıştır. Eğitim başarısı artarken doğrulama başarısı artıp azalarak aynı seviyede kalmaya başlamıştır. Maliyet fonksiyonunun değişimde ise eğitimin maliyet fonksiyon çıktısı giderek azalırken, doğrulama verisinin maliyet fonksiyonu azalmak yerine artmaya başlamıştır. Tam bağlı katmanlarda dropout kullanılmış olsa da bu durumlar aşırı uyum problemi olduğunun göstergesidir.

Model 2’ nin doğruluk değerinin fazla olmasının yanında doğruluğu ve maliyet fonksiyonunun değişimini gösteren grafiklerde eğitim ile doğrulama arasındaki fark ilk modele göre oldukça azaltılmıştır. Bunun nedeni verilerin diğer veri setine kıyasla daha çok sayıda ve çeşitli sürümleri içeren verileri barındırmasıdır.

4.5.2. Test 2

Test 2’ de tam bağlı katmanların nöron sayılarının değiştirilmesi ile üç farklı model elde edilmiştir. Modellerin özelliklerini içeren Şekil 4.10‘ da görüldüğü gibidir. Modellerin tam bağlı katmanlarındaki nöron sayılarının artmasıyla birlikte öğrenilebilen parametrelerinin de arttığı görülmektedir.

Tablo 4.9 Test 2’de kullanılan modellerinin özellikleri

	Konvolüzasyon Katmanı Sayısı	Havuzlama Katmanı Sayısı	Tam Bağlı Katman Sayısı	Tam Bağlı Katmanların Nöron Sayıları	Parametre Sayısı
Model 3	5	3	3	512	4.885,023
Model 4	5	3	3	1024	8.899,615
Model 5	5	3	3	2048	18.501,663

Modellerin yapılarını temsil eden tablolarda görüldüğü gibi Test 1’ de kullanılan modellerin sadece tam bağlı katmanlarında dropout kullanılmıştır. Test 2’ de ise tam bağlı katmanların yanı sıra modellerin tümünde konvolüzasyon veya havuzlama katmanlarından sonra dropout tekniği uygulanmış ve artan parametreler baz alınarak

etkinliği ölçülmüştür. Test 2’ de kullanılan tüm modellerin modelin katmanlara göre parametre sayıları ve çıkış şekillerini gösteren model yapıları Tablo 4.10, Tablo 4.11 ve Tablo 4.12’ de görüldüğü gibidir.

Tablo 4.10 Model 3’ ün yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 50, 50, 64)	9472
1. Maksimum Havuzlama	(x, 25, 25, 64)	0
1. Dropout(%20)	(x, 25, 25, 64)	0
2. Konvolüzasyon	(x, 25, 25, 128)	204928
2. Maksimum Havuzlama	(x, 13, 13, 128)	0
2. Dropout(%20)	(x, 13, 13, 128)	0
3. Konvolüzasyon	(x, 13, 13, 256)	295168
3. Dropout(%20)	(x, 13, 13, 256)	0
4. Konvolüzasyon	(x, 13, 13, 256)	590080
4. Dropout(%20)	(x, 13, 13, 256)	0
5. Konvolüzasyon	(x, 13, 13, 128)	295040
3. Maksimum Havuzlama	(x, 7, 7, 128)	0
5. Dropout(%20)	(x, 7, 7, 128)	0
Düzleştirme Katmanı	(x, 6272)	0
1. Tam Bağlı Katman	(x, 512)	3211776
6. Dropout(%50)	(x, 512)	0
2. Tam Bağlı Katman	(x, 512)	262656
7. Dropout(%50)	(x, 512)	0
3. Tam Bağlı Katman	(x, 31)	15903

Tablo 4.11 Model 4’ ün yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 50, 50, 64)	9472
1. Maksimum Havuzlama	(x, 25, 25, 64)	0
1. Dropout(%20)	(x, 25, 25, 64)	0
2. Konvolüzasyon	(x, 25, 25, 128)	204928
2. Maksimum Havuzlama	(x, 13, 13, 128)	0
2. Dropout(%20)	(x, 13, 13, 128)	0
3. Konvolüzasyon	(x, 13, 13, 256)	295168
3. Dropout(%20)	(x, 13, 13, 256)	0
4. Konvolüzasyon	(x, 13, 13, 256)	590080
4. Dropout(%20)	(x, 13, 13, 256)	0
5. Konvolüzasyon	(x, 13, 13, 128)	295040
3. Maksimum Havuzlama	(x, 7, 7, 128)	0
5. Dropout(%20)	(x, 7, 7, 128)	0
Düzleştirme Katmanı	(x, 6272)	0
1. Tam Bağlı Katman	(x, 1024)	6423552
6. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 1024)	1049600
7. Dropout(%50)	(x, 1024)	0
3. Tam Bağlı Katman	(x, 31)	31775

Tablo 4.12 Model 5’ in yapısı

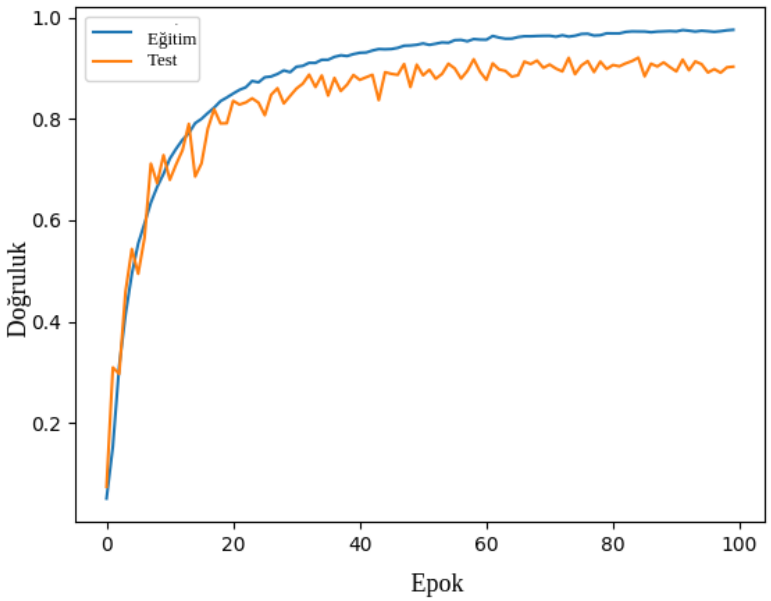
Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 50, 50, 64)	9472
1. Maksimum Havuzlama	(x, 25, 25, 64)	0
1. Dropout(%20)	(x, 25, 25, 64)	0
2. Konvolüzasyon	(x, 25, 25, 128)	204928
2. Maksimum Havuzlama	(x, 13, 13, 128)	0
2. Dropout(%20)	(x, 13, 13, 128)	0
3. Konvolüzasyon	(x, 13, 13, 256)	295168
3. Dropout(%20)	(x, 13, 13, 256)	0
4. Konvolüzasyon	(x, 13, 13, 256)	590080
4. Dropout(%20)	(x, 13, 13, 256)	0
5. Konvolüzasyon	(x, 13, 13, 128)	295040
3. Maksimum Havuzlama	(x, 7, 7, 128)	0
5. Dropout(%20)	(x, 7, 7, 128)	0
Düzleştirme Katmanı	(x, 6272)	0
1. Tam Bağlı Katman	(x, 2048)	12847104
6. Dropout(%50)	(x, 2048)	0
2. Tam Bağlı Katman	(x, 2048)	4196352
7. Dropout(%50)	(x, 2048)	0
3. Tam Bağlı Katman	(x, 31)	63519

Test edilen modellerin başarıları Tablo 4.13' te görüldüğü gibi tam bağlı katmanlardaki nöron sayısının belirli bir değere kadar artması başarı oranını arttırırken çok fazla sayıda artmasıyla birlikte parametre sayısının da artmasıyla modelin başarısını düşürdüğü gözlemlenmiştir. Tam bağlı katmanlarının tümünde 512 nöron bulunan Model 3 ile 1024 nöron bulunan Model 4' e yakın sonuçlar elde edilmiştir.

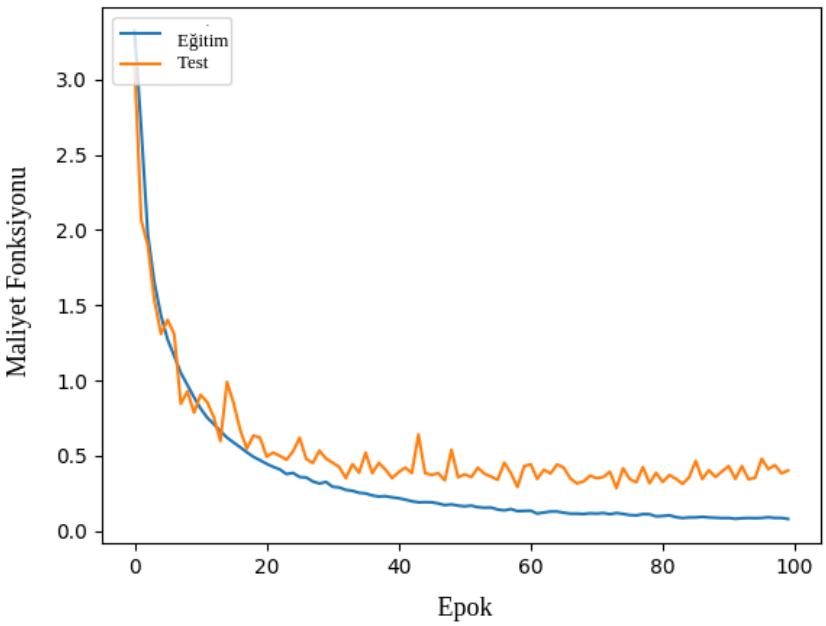
Tablo 4.13 Test 2' deki modellerin performansı

	Veri Seti	Görüntü Boyutu	Epok Sayısı	Başarı(%)	Kesinlik	Anma Değeri	F1 Skoru
Model 3	2	200x200	100	90.00	0.91	0.90	0.90
Model 4	2	200x200	100	91.42	0.92	0.91	0.91
Model 5	2	200x200	100	77.54	0.81	0.78	0.77

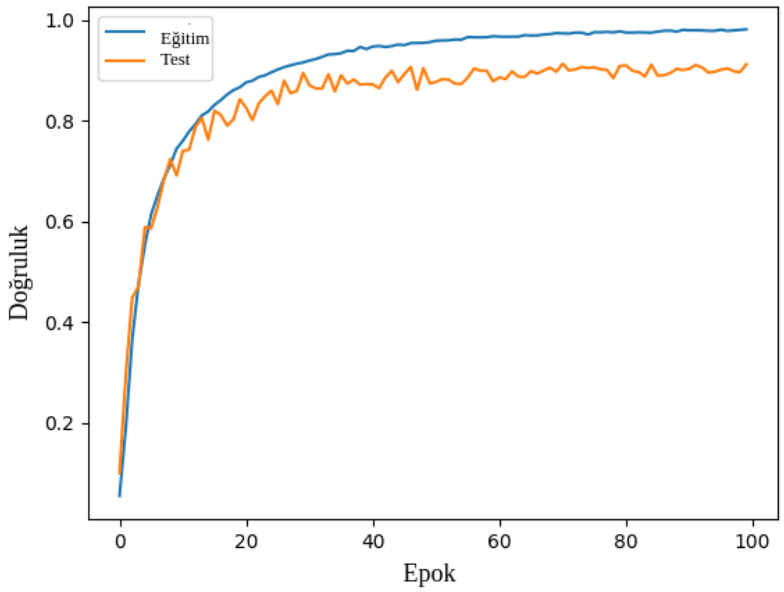
Model 3, Model 4 ve Model 5' in doğruluk değişimlerini temsil eden grafikler, Şekil 4.9, Şekil 4.11 ve Şekil 4.13' te görülmektedir. Bunun yanında maliyet fonksiyonlarındaki değişimi gösteren grafikler Şekil 4.10,Şekil 4.12 ve Şekil 4.14' teki gibidir.



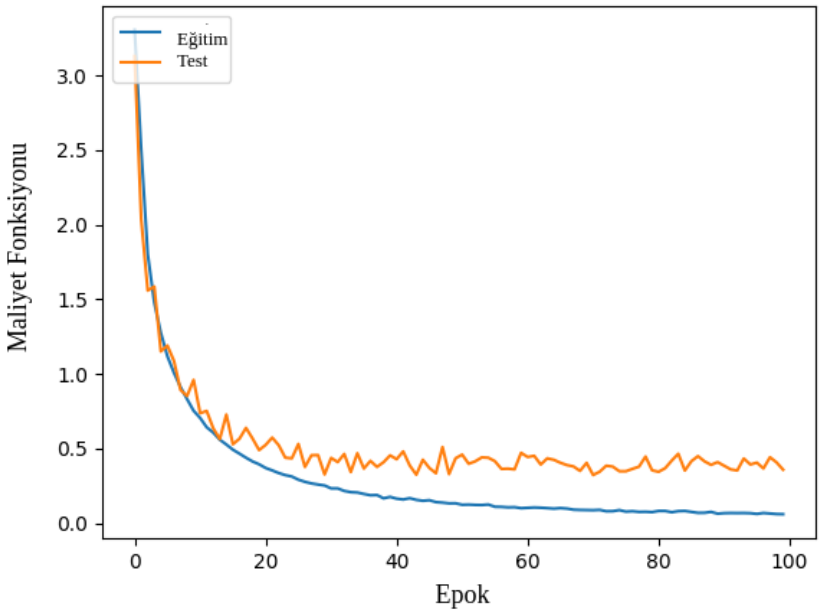
Şekil 4.9 Model 3’ ün doğruluk grafiği



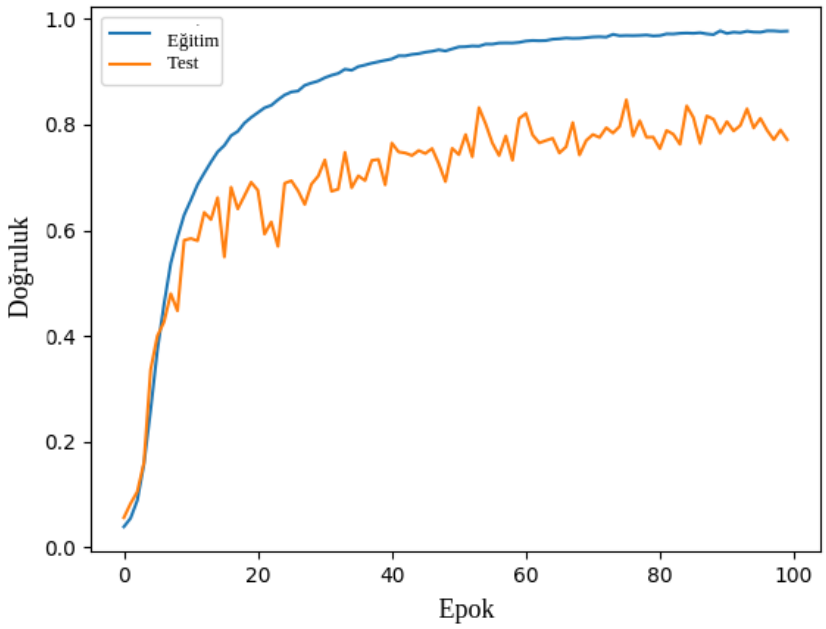
Şekil 4.10 Model 3’ ün maliyet fonksiyonu değişim grafiği



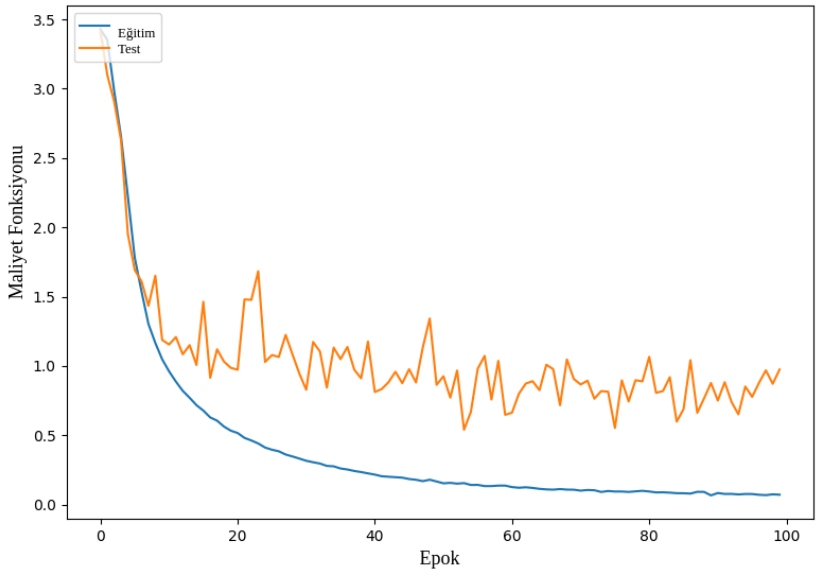
Şekil 4.11 Model 4' ün doğruluk grafiği



Şekil 4.12 Model 4' ün maliyet fonksiyonu değişim grafiği



Şekil 4.13 Model 5’ in doğruluk grafiği



Şekil 4.14 Model 5’ in maliyet fonksiyonu değişim grafiği

Grafiklerden de görüldüğü gibi Model 3 ve Model 4' ün eğitim ve test başarısı arasındaki fark Model 5' e ait olan doğruluk oranlarının farkından daha azdır. Model 5' e ait grafiklerde yaklaşık 15 epoktan sonra test doğruluğun çok az arttığı, maliyet fonksiyonunun da azalması gerekirken bazı noktalarda arttığı gözlemlenmiştir.

4.5.3. Test 3

Test 3'te 100x100 boyutunda görüntüler kullanılarak, modellerin derinliklerinin değiştirilmiş ve performanslar ölçülmüştür. Konvolüzasyon ve tam bağlı katman sayılarına göre aynı veri setini kullanarak farklı modeller elde edilmiştir. Tablo 4.14' te modellerin özellikleri görülmektedir. Model 6' nın 3 adet konvolüzasyon katmanı ve 3 adet havuzlama katmanı bulunmaktadır. Model 8' de de 3 adet havuzlama katmanı bulunurken Model 6'dan farklı olarak 5 adet konvolüzasyon katmanı bulunmaktadır. Bu durumun parametre artışına neden olduğu görülmektedir. Bunun yanında Model 6' nın konvolüzasyon ve havuzlama katmanı sayılarının arttırılmasıyla oluşturulan Model 7' nin parametre sayısında düşüş sağlanmıştır.

Tablo 4.14 Test 3' te kullanılan modellerin özellikleri

	Konvolüzasyon Katmanı Sayısı	Havuzlama Katmanı Sayısı	Tam Bağlı Katman Sayısı	Parametre Sayısı
Model 6	3	3	2	2.491,935
Model 7	4	4	2	890.463
Model 8	5	3	3	4.574,239

Test 3' te kullanılan modellerin yapıları Tablo 4.15, Tablo 4.16 ve Tablo 4.17'de görüldüğü gibidir.

Tablo 4.15 Model 6’ nın yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 25, 25, 64)	9472
1. Maksimum Havuzlama	(x, 13, 13, 64)	0
2. Konvolüzasyon	(x, 13, 13, 128)	204928
2. Maksimum Havuzlama	(x, 7, 7, 128)	0
3. Konvolüzasyon	(x, 7, 7, 128)	147584
3. Maksimum Havuzlama	(x, 4, 4, 128)	0
Düzleştirme Katmanı	(x, 2048)	0
1. Tam Bağlı Katman	(x, 1024)	2098176
1. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 31)	31775

Tablo 4.16 Model 7’ nin yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 25, 25, 64)	9472
1. Maksimum Havuzlama	(x, 13, 13, 64)	0
2. Konvolüzasyon	(x, 13, 13, 64)	102464
2. Maksimum Havuzlama	(x, 7, 7, 64)	0
3. Konvolüzasyon	(x, 7, 7, 128)	73856
3. Maksimum Havuzlama	(x, 4, 4, 128)	0
4. Konvolüzasyon	(x, 4, 4, 128)	147584
4. Maksimum Havuzlama	(x, 2, 2, 128)	0
Düzleştirme Katmanı	(x, 512)	0
1. Tam Bağlı Katman	(x, 1024)	525312
1. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 31)	31775

Tablo 4.17 Model 8’ in yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 25, 25, 64)	9472
1. Maksimum Havuzlama	(x, 13, 13, 64)	0
1. Dropout(%20)	(x, 13,, 13, 64)	0
2. Konvolüzasyon	(x, 13, 13, 128)	204928
2. Maksimum Havuzlama	(x, 7, 7, 128)	0
2. Dropout(%20)	(x, 7, 7, 128)	0
3. Konvolüzasyon	(x, 7, 7, 256)	295168
3. Dropout(%20)	(x, 7, 7, 256)	0
4. Konvolüzasyon	(x, 7, 7, 256)	590080
4. Dropout(%20)	(x, 7, 7, 256)	0
5. Konvolüzasyon	(x, 7, 7, 128)	295040
3. Maksimum Havuzlama	(x, 4, 4, 128)	0
5. Dropout(%20)	(x, 4, 4, 128)	0
Düzleştirme Katmanı	(x, 2048)	0
1. Tam Bağlı Katman	(x, 1024)	2098176
6. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 1024)	1049600
7. Dropout(%50)	(x, 1024)	0
3. Tam Bağlı Katman	(x, 31)	31775

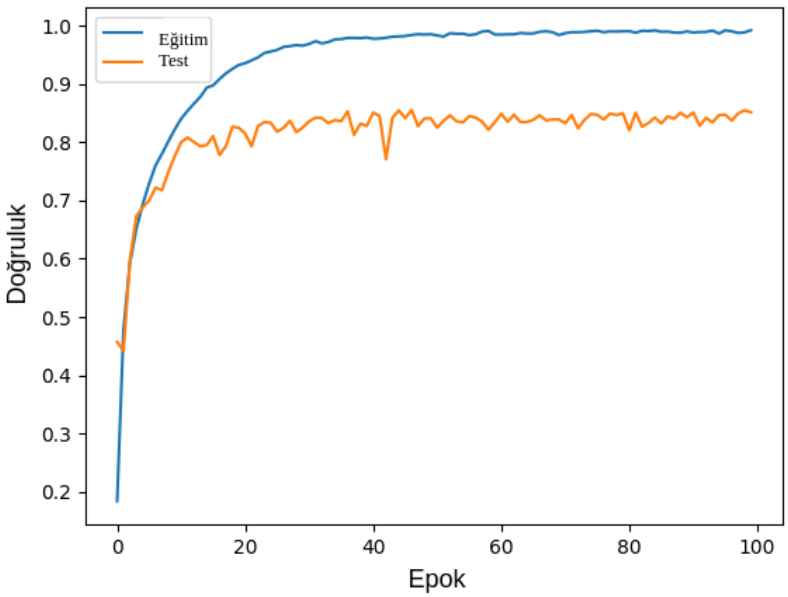
Tablo 4.18’ de görüldüğü gibi Model 6’ nın konvolüzasyon katman sayısını arttırdığımda elde edilen Model 8 daha başarılıdır. Bunun yanında Model 6’ nın hem

konvolüzasyon hem de havuzlama katmanı sayısı arttırıldığında başarı oranı ve diğer performans metriklerinde düşüş gözlemlenmektedir.

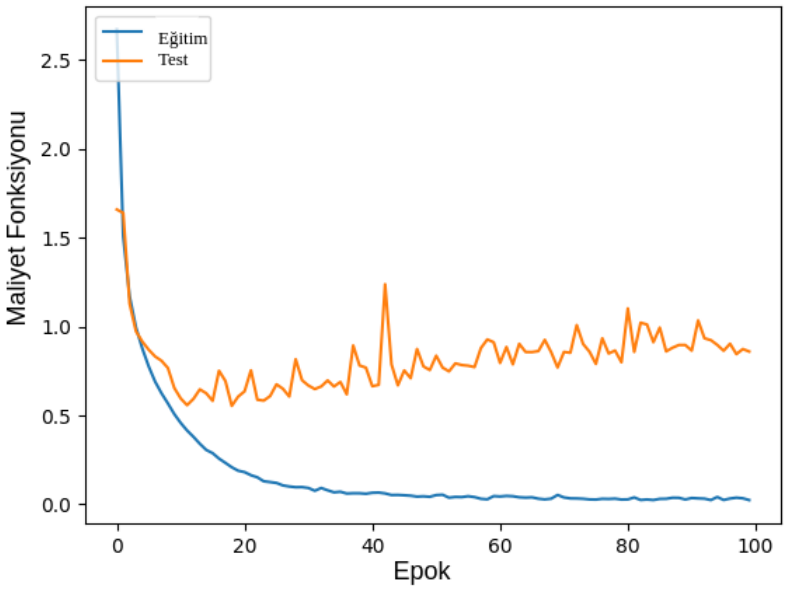
Tablo 4.18 Test 3' teki modellerin performansı

	Veri Seti	Görüntü Boyutu	Epok Sayısı	Başarı(%)	Kesinlik	Anma Değeri	F1 Skoru
Model 6	2	100x100	100	85.22	0.86	0.85	0.85
Model 7	2	100x100	100	84.52	0.85	0.85	0.84
Model 8	2	100x100	100	86.16	0.87	0.86	0.86

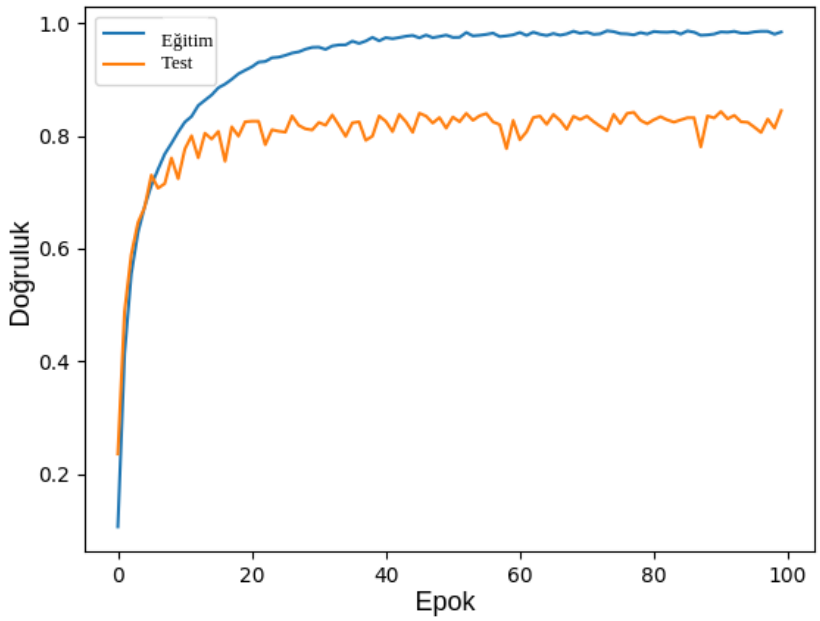
Test 3' teki tüm modellerin doğruluk artım grafiği Şekil 4.15, Şekil 4.17 ve Şekil 4.19' da, maliyet fonksiyonu değişim grafikleri Şekil 4.16, Şekil 4.18 ve Şekil 4.20' de yer almaktadır.



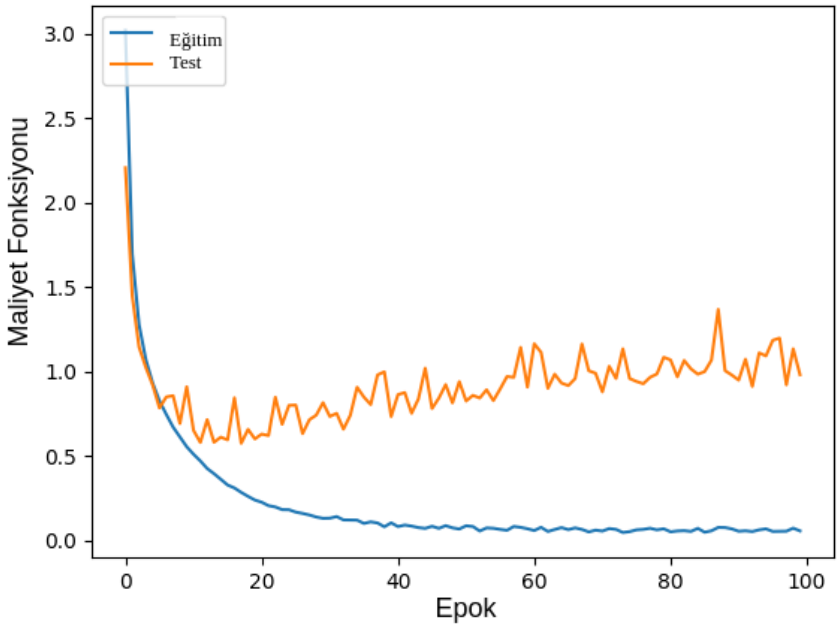
Şekil 4.15 Model 6’ nın doğruluk grafiği



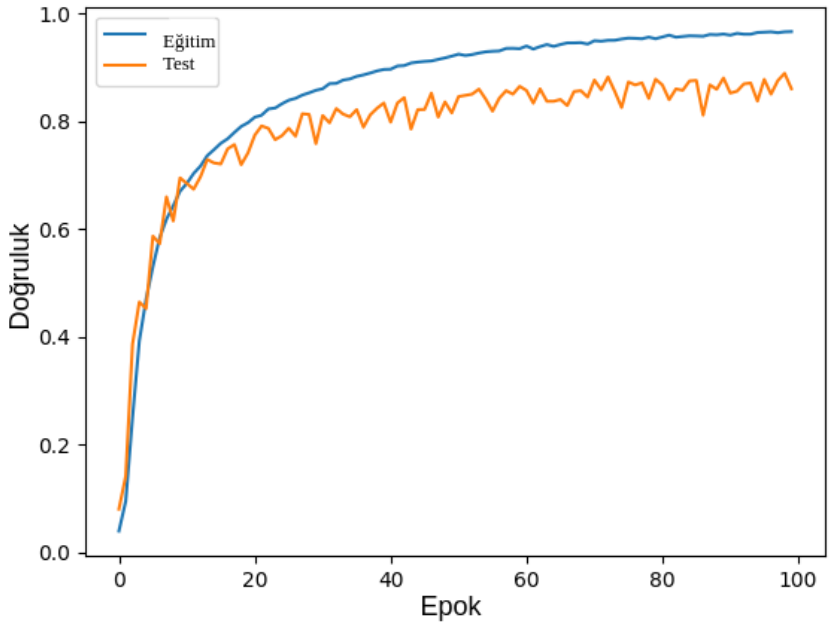
Şekil 4.16 Model 6’ nın maliyet fonksiyonu değişim grafiği



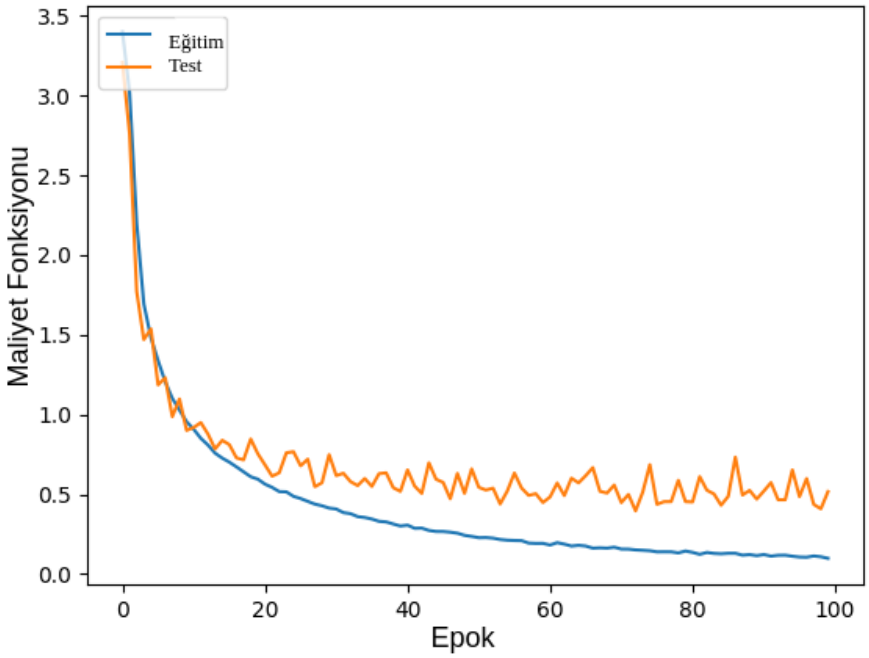
Şekil 4.17 Model 7’ nin doğruluk grafiği



Şekil 4.18 Model 7’ nin maliyet fonksiyonu değişim grafiği



Şekil 4.19 Model 8’ in doğruluk grafiği



Şekil 4.20 Model 8’ in maliyet fonksiyonu değişim grafiği

4.5.4. Test 4

Test 3’ te 100x100 boyutunda görüntülerin kullanılmasıyla elde edilen performans değerlerinin diğer testlere göre daha düşük olduğu gözlemlenmiştir. Bunun üzerine derinlik faktörünün 200x200 boyutunda görüntüler ile aynı değişikliklerin nasıl sonuç vereceği gözlemlenmiştir. Tablo 4.19’ da Test 3’ teki gibi parametre sayısının konvolüzasyon katmanı sayısına bağlı arttığı ve havuzlama katmanı sayısına bağlı olarak azaldığı görülmektedir. Fakat parametre sayı değerlerinin Test 3’ e göre daha fazladır. Bu testte doğruluk değeri yüksek, diğer modellerle karşılaştırma yapmaya uygun olan, Test 2’ de de kullanılan Model 4 seçilmiştir.

Tablo 4.19 Test 4’te kullanılan modellerin özellikleri

	Konvolüzasyon Katmanı Sayısı	Havuzlama Katmanı Sayısı	Tam Bağlı Katman Sayısı	Parametre Sayısı
Model 9	3	3	2	6.817,311
Model 10	4	4	2	2.463,327
Model 4	5	3	3	8.899,615

Tablo 4.20 Model 9’ un yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 50, 50, 64)	9472
1. Maksimum Havuzlama	(x, 25, 25, 64)	0
2. Konvolüzasyon	(x, 25, 25, 128)	204928
2. Maksimum Havuzlama	(x, 13, 13, 128)	0
3. Konvolüzasyon	(x, 13, 13, 128)	147584
3. Maksimum Havuzlama	(x, 7, 7, 128)	0
Düzleştirme Katmanı	(x, 6272)	0
1. Tam Bağlı Katman	(x, 1024)	6423552
1. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 31)	31775

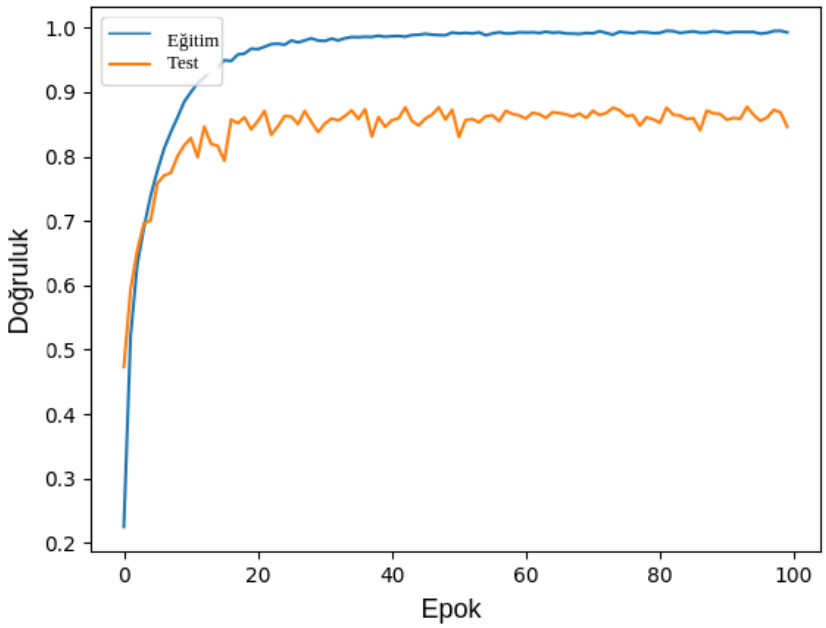
Tablo 4.21 Model 10’ un yapısı

Katman Tipi	Çıkış Şekli	Parametre Sayısı
1. Konvolüzasyon	(x, 50, 50, 64)	9472
1. Maksimum Havuzlama	(x, 25, 25, 64)	0
2. Konvolüzasyon	(x, 25, 25, 64)	102464
2. Maksimum Havuzlama	(x, 13, 13, 64)	0
3. Konvolüzasyon	(x, 13, 13, 128)	73856
3. Maksimum Havuzlama	(x, 7, 7, 128)	0
4. Konvolüzasyon	(x, 7, 7, 128)	147584
4. Maksimum Havuzlama	(x, 4, 4, 128)	0
Düzleştirme Katmanı	(x, 2048)	0
1. Tam Bağlı Katman	(x, 1024)	2098176
1. Dropout(%50)	(x, 1024)	0
2. Tam Bağlı Katman	(x, 31)	31775

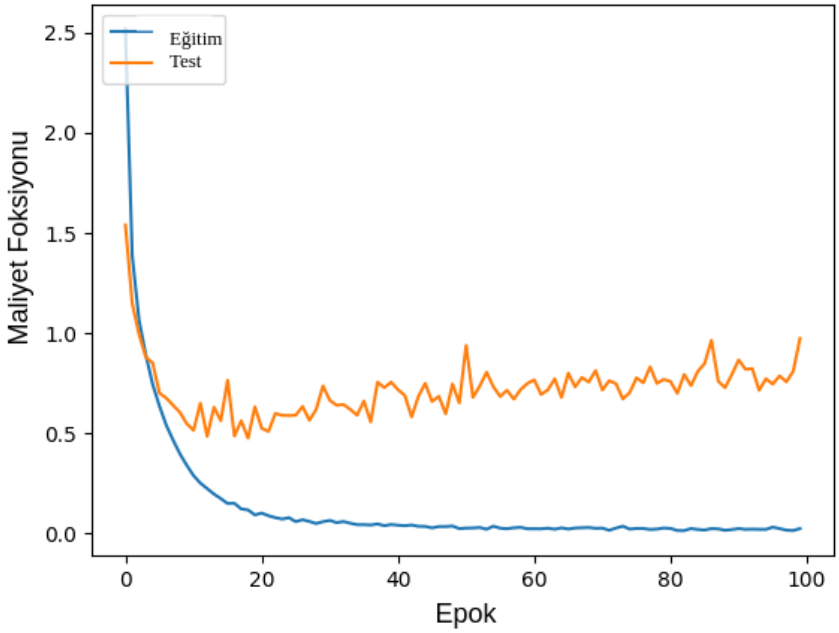
Tablo 4.22’ de yer alan test sonuçlarında Test 3’ ten farklı olarak konvolüzasyon katmanı sayısı, havuzlama katmanı sayısı ile aynı oranda arttığında başarı oranında ve diğer performans metriklerinde artış gözlemlenmektedir. Ayrıca diğer modellere göre daha derin olan Model 4 en iyi sonucu göstermiştir.

Tablo 4.22 Test 4’teki modellerin performansı

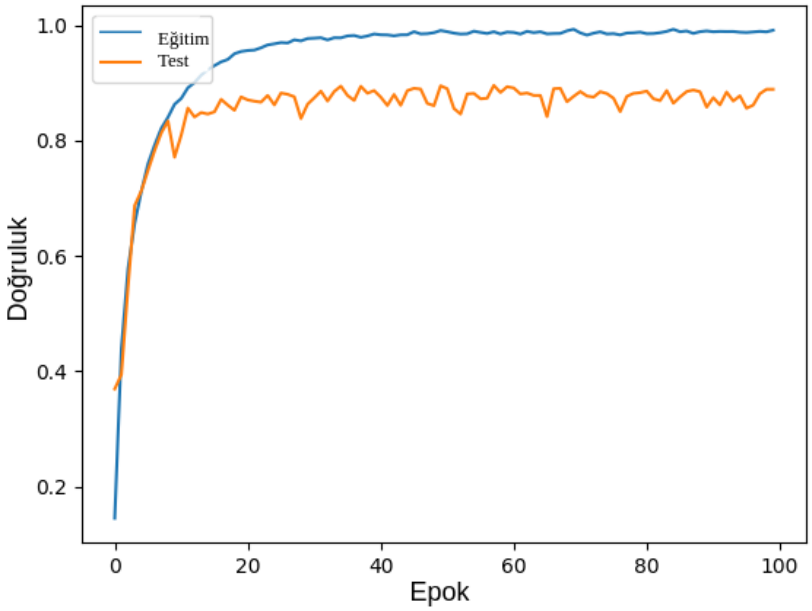
	Veri Seti	Görüntü Boyutu	Epok Sayısı	Başarı(%)	Kesinlik	Anma Değeri	F1 Skoru
Model 9	2	200x200	100	84.55	0.86	0.85	0.84
Model 10	2	200x200	100	89.16	0.89	0.89	0.89
Model 4	2	200x200	100	91.42	0.92	0.92	0.91



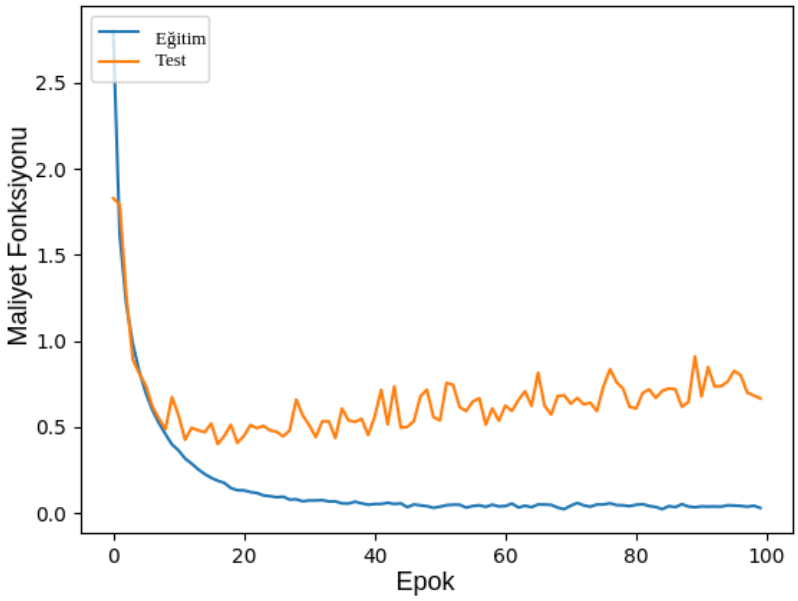
Şekil 4.21 Model 9’ un doğruluk grafiği



Şekil 4.22 Model 9’ un maliyet fonksiyonu değişim grafiği



Şekil 4.23 Model 10’ un doğruluk grafiği



Şekil 4.24 Model 10’ un maliyet fonksiyonu değişim grafiği

Şekil 4.21 ve Şekil 4.23' te yer alan doğruluk değişim grafiklerinin ikisinde de eğitim ve test başarısı arasında fark olmasına rağmen artan havuzlama katmanı sayısı ile birlikte parametrelerin azalmasıyla Model 10' un test başarısı, Model 9' a göre eğitim başarısına daha yakınsanmıştır. Model 9' da daha büyük bir şekilde olmak üzere Şekil 4.22 ve Şekil 4.24' te bulunan maliyet fonksiyonu değerinin azalması gerekirken artmaya başlamıştır. Şekil 4.12' de bulunan Model 4' e ait maliyet fonksiyonunda eğitim sonuçlarına daha yakın değerler elde edilmiştir. Bu modellere kıyasla Model 4 en tutarlı sonucu göstermiştir.

BÖLÜM 5

SONUÇ

Yapılan testler sonucunda KSA' nın görüntüleri sınıflandırmada oldukça başarılı olduğu görülmüştür. Çok sayıda veri ve çeşitli denemeler ile birlikte başarı artımı gözlenebilmektedir. Bununla birlikte çok sayıda, çeşitli hesaplamalar yapıldığı için derin öğrenme tekniğinin GİB 'leri üzerinde daha iyi bir şekilde eğitileceği gözlemlenmiştir. Yapay zekânın kullanımı ile birlikte birçok alanda verimli bir şekilde süreç kontrolü yapılacak ve insanların iş süreçlerinde yaptığı yanlışlar insan görüsüne yardımcı olarak kullanılan akıllı sistemler ile birlikte hataların daha aza indirgenmesi sağlanabilecektir. Geliştirilen derin öğrenme modelleri ile bu durumun desteklendiği görülmektedir.

Yapılan testler sonucunda 31 farklı sınıf üzerinde %91.42 oranında sınıflandırma başarısı elde edilmiş olup, bu oran insan görüsüne yakındır. Bu çalışmada elde edilen tecrübe kullanılarak, farklı çalışma alanlarında farklı ürünlerin sınıflandırılması ve iş süreçlerinde otonom olarak sınıflandırma yapabilen sistemlerin geliştirilmesi hedeflenmektedir.

KAYNAKLAR

[1] Bo Zhang, ***Computer Vision vs. Human Vision***, Proc. 9th IEEE Int. Conf. on Cognitive Informatics (ICCI'10), (2010).

[2] A. Krizhevsky, I. Sutskever, and Geoffrey E. Hinton, ***ImageNet classification with deep convolutional neural networks***, Advances in Neural Information Processing Systems 25, (2012).

[3] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, Brian Kingsbury, ***Deep neural networks for acoustic modeling in speech recognition***, IEEE Signal Processing Magazine, cilt 29, sayı no 29, sy. 82–97, (2012).

[4] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky, ***Strategies for training large scale neural network language models***, Proc. Autom. Speech Recognit. Understand, sy. 196–201, (2011).

[5] Andre Luckow, Matthew Cook, Nathan Ashcraft, Edwin Weill, Emil Djerekarov, Bennie Vorster, ***Deep Learning in the Automotive Industry***, IEEE International Conference on Big Data (Big Data), (2016).

[6] James Reason, Michael Maddox, ***Human Error***, Federal Aviation Administration, (2015).

[7] J Lee, B Bagheri, H.A. Kao, ***Recent Advances and Trends of Cyber-Physical Systems and Big Data Analytics in Industrial Informatics***, International Proceeding of Int Conference on Industrial Informatics, (2014).

[8] Jay Lee, ***Trends and Recent Advances of Industrial Big Data Predictive Analytics and Industrial 4.0 for Future Manufacturing***, Sirris, (2014)

[9] Tushar Jain, Meenu, ***Automation and Integration of Industries through Computer Vision Systems***, International Journal of Information and Computation Technology, (2013).

[10] Mital, A., Govindaraju, M., and Subramani, B., ***Comparison between Manual and Hybrid Methods in Parts Inspection***, Integrated Manufacturing Systems, sy. 344-349, (1998).

[11] Matthew D. ZeilerRob Fergus, ***Visualizing and Understanding Convolutional Networks***, European Conference on Computer Vision(ECCV), (2014).

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, ***Going Deeper with Convolutions***, Computer Vision and Pattern Recognition (CVPR) IEEE Conference, (2015).

[13] Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner, ***Gradient-based learning applied to document recognition***, Proceedings of the IEEE, sy. 2278-2324, (1998).

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei, ***ImageNet Large Scale Visual Recognition Challenge***, International Journal of Computer Vision (IJCV), sy. 211-252, (2015).

[15] Karen Simonyan, Andrew Zisserman, ***Very Deep Convolutıonal Networks For Large -Scale Image Recognition***, ICLR, (2015).

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, ***Deep Residual Learning for Image Recognition***, Computer Vision and Pattern Recognition (CVPR) IEEE Conference, (2016).

[17] Andrej Karpathy, Li Fei-Fei, ***Deep Visual-Semantic Alignments for Generating Image Descriptions***, IEEE Conference on Computer Vision and Pattern Recognition, (2015).

[18] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, ***Rich feature hierarchies for accurate object detection and semantic segmentation***, Computer Vision and Pattern Recognition (CVPR) IEEE Conference, sy. 580-587, (2014).

[19] J.R. Uijlings, K.E.Sande, T. Gevers , A.W. Smeulders, ***Selective Search for Object Recognition***, International Journal of Computer Vision, sy. 154-171 ,(2013).

[20] Ross Girshick, ***Fast R-CNN***, ICCV '15 Proceedings of the IEEE International Conference on Computer Vision (ICCV), sy. 1440-1448, (2015).

[21] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, ***Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks***, Advances in Neural Information Processing Systems 28 (NIPS), (2015).

[22] Hokuto Kagaya, Kiyoharu Aizawa, Makoto Ogawa, ***Food Detection and Recognition Using Convolutional Neural Network***, Proceedings of the 22nd ACM international conference on Multimedia, Sy. 1085-1088, (2014).

[23] Yuzhen Lu, ***Food Image Recognition by Using Convolutional Neural Networks (CNNs)***, arXiv preprint arXiv:1612.00983, (2016).

[24] Keiji Yanai, Yoshiyuki Kawano, ***Food image recognition using deep convolutional network with pre-training and fine-tuning***, Multimedia & Expo Workshops (ICMEW) IEEE International Conference, (2015).

[25] Suchet Bargoti, James Underwood, ***Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards***, The Journal of Field Robotics, (2016).

[26] Kaz Sato, ***How a Japanese cucumber farmer is using deep learning and TensorFlow***, https://cloud.google.com/blog/big-data, (2016).

[27] Patricia Constante, A.Gordon, Oscar Chang , E. Pruna, Fausto Acuna, I. Escobar, ***Artificial Vision Techniques for Strawberry's Industrial Classification***, IEEE Latin America Transactions, (2016).

[28] Yezhou Yang, Yi Li, Cornelia Fermüller, Yiannis Aloimonos, ***Robot Learning Manipulation Action Plans by "Watching" Unconstrained Videos from the World Wide Web***, The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15), (2015).

[29] Tonya Hines , ***Anatomy of The Brain***, Mayfield Clinic, (2016).

[30] Richard Morris, Marianne Fillenz, ***Neuroscience Science Of The Brain An Introduction For Young Students***, (British Neuroscience Association, UK, 2003)

[31] Mona Miller, Todd Bentsen, DeeDee Clendenning, Sara Harris, Debra Speert, Connie Binder, Lydia V. Kibiuk, Baltimore, Maryland, ***Brain Facts A Primer on the Brain and Nervous System***, (Society for Neuroscience, USA, 2008).

[32] Simon Haykin, ***Neural Networks and Learning Machines***, (Pearson Education, United States of America, 2009).

[33] David Kriesel, ***A Brief Introduction to Neural Networks***, (2005).

[34] Ian Goodfellow, Yoshua Bengio, Aaron Courville, ***Deep Learning(Adaptive Computation and Machine Learning series)***, (MIT Press, Cambridge, 2016).

[35] Christos Stergiou, Dimitrios Sigano, ***Neural Networks***, Imperial College London CS11 Notes, (1996).

[36] James L. McClelland, ***Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises***, (Standford Üniversitesi, 2015).

[37] Jeffrey L. Elman, ***Finding Structure in Time***, Cognitive Scıence 14, (1990).

[38] Li Deng, Dong Yu Now, ***Deep Learning: Methods and Applications, (Now, Publisher, Boston USA, 2014).***

[39] Larry Brown, ***Deep Learning With Gpus***, Geoint Sunum, (2015).

[40] Juha Karhunen, Tapani Raiko, Kyung Hyun Cho, ***Unsupervised Deep Learning: A Short Review***, Advances in Independent Component Analysis and Learning Machines, (Academic Press, 2015).

[41] Gary Brassington, ***Mean absolute error and root mean square error: which is the better metric for assessing model performance?***, Geophysical Research Abstracts, (2017).

[42] T. Chai, R. R. Draxler***, Root mean square error (RMSE) or mean absolute error (MAE)? –Arguments against avoiding RMSE in the literatüre***, Geosci. Model Dev., (2014).

[43] Michael A. Nielsen, ***Neural Networks and Deep Learning***, (Determination Press, 2015).

[44] Javier Bejar, ***Artificial Neural Networks***, UPC Üniversitesi Ders Notları, (2012).

[45] Urtasun, Zemel,***: Lecture 10: Neural Networks I***, Toronto Üniversitesi CSC 411 Ders Notları, (2015).

[46] Elad H., Nir A., Ran E., ***Introductıon To Deep Learning***, İsrail Teknoloji Enstitüsü Ders Notları, (2016).

[47] Geoffrey Hinton, ***Lecture 9a: Overview of ways to improve generalization***, Toronto Üniversitesi CSC321 Ders Notları, (2014).

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, ***Dropout: A Simple Way to Prevent Neural Networks from Overfitting***, Journal of Machine Learning Research 15, (2014).

[49] API Documentation, https://www.tensorflow.org/api_docs/

[50] TensorFlow Geliştirme Takımı, ***TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems***, arXiv preprint arXiv:1603.04467, (2016).

[51] Shaohuai Shi, Qiang Wang, Pengfei Xu, Xiaowen Chu, ***Benchmarking State-of-the-Art Deep Learning Software Tools***, arXiv preprint arXiv: 1608.07249,(2017).

[52] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, Yoshua Bengio, ***Theano: A CPU and GPU Math Compiler in Python***, Proc. of the 9th Python in Science Conf., (2010).

[53] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, Yoshua Bengio, ***Theano: new features and speed improvements***, Presented at the Deep Learning Workshop NIPS, (2012).

[54] Theano Geliştirme Takımı, ***Theano: A Python framework for fast computation of mathematical expressions***, arXiv preprint arXiv: 1605.02688, (2016).

[55] R. Collobert, L.J.P. van der Maaten, and A. Joulin. ***Torchnet: An Open-Source Platform for (Deep) Learning Research***. ICML Machine Learning Systems Workshop, (2016).

[56] Ronan Collobert, Koray Kavukcuoglu, Clement Farabet, ***Torch7: A Matlab-like Environment for Machine Learning***, Nıps Workshop, (2011).

[57] Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, Mohak Shah, ***Comparative Study Of Caffe, Neon, Theano, And Torch For Deep Learning***, ICLR, (2016).

[58] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell, ***Caffe: Convolutional Architecture for Fast Feature Embedding***, MM '14 Proceedings of the 22nd ACM international conference on Multimedia, sy. 675-678, (2014).

[59] Keras: Deep Learning for Python, https://github.com/fchollet/keras

ÖZGEÇMİŞ

29.06.1993 yılında Kırklareli ilinin Lüleburgaz ilçesinde doğdum. Orta ve lise öğrenimini Lüleburgaz' da tamamladım. 2011-2015 yılları arasında Trakya Üniversitesi Mühendislik Fakültesi, Bilgisayar Mühendisliği bölümünde öğrenim gördüm. 2 yıl süreyle Trakya Teknopark' da faaliyet gösteren firmalarda Ar-Ge personeli olarak çalıştım.

TEZ ÖĞRENCİSİNE AİT TEZ İLE İLGİLİ BİLİMSEL FAALİYETLER

C. Taşkın, A. Yiğit, D. Taşkın, A.C. Taşkın, ***Image Processing On Mobile Devices With OpenCV***, International Conference on Engineering, Technology and Systems – TechSys 2017, sy.151-155, (2017).