

MATLAB ORTAMINDA DERİN ÖĞRENME UYGULAMALARI

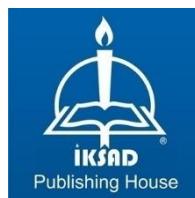
Dr. Öğr. Üyesi Sedat METLEK

Öğr. Gör. Dr. Halit ÇETİNER



MATLAB ORTAMINDA
DERİN ÖĞRENME
UYGULAMALARI

Dr. Öğr. Üyesi Sedat METLEK Öğr. Gör. Dr. Halit ÇETİNER



Copyright © 2021 by iksad publishing house
All rights reserved. No part of this publication may be reproduced, distributed or
transmitted in any form or by
any means, including photocopying, recording or other electronic or mechanical
methods, without the prior written permission of the publisher, except in the case of
brief quotations embodied in critical reviews and certain other noncommercial uses
permitted by copyright law. Institution of Economic Development and Social
Researches Publications®
(The Licence Number of Publicator: 2014/31220)
TURKEY TR: +90 342 606 06 75
USA: +1 631 685 0 853
E mail: ikssadyayinevi@gmail.com
www.ikssadyayinevi.com

It is responsibility of the author to abide by the publishing ethics rules.
Iksad Publications – 2021©

ISBN: 978-625-7562-08-9
Cover Design: İbrahim KAYA
June / 2021
Ankara / Turkey
Size = 16x24 cm

ÖNSÖZ

Günümüzde yazılım sektöründe sık sık kullanılan derin öğrenme kavramı televizyondan internete kadar birçok yazılı ve görsel platformda karşımıza çıkmaktadır. Literatürde de farklı tanımları olmakla beraber, yapılan tanımların genel olarak ortak noktası herhangi bir olay ile ilgili görüntüler üzerinden karar verme sürecinde kullanılacak parametrelerin otomatik olarak çıkartılarak, insan beynindeki anlama ve karar verme sürecinin taklit edilmesidir. Bu nedenle literatürde yapılan çalışmalar incelendiğinde birçoğunun görüntüler üzerinden uygulandığı görülmektedir. Akademik literatürün dışında genel olarak yazılım sektörüne de yeni bir soluk getirmiştir.

Google, Microsoft ve Facebook gibi Dünya genelinde önemli bir yere sahip olan yazılım şirketleri derin öğrenme kavramı üzerine daha da yoğunlaşarak metin işlemeden yüz tanıtmaya, otonom sürüsten görüntü işlemeye kadar birçok uygulamada CNN tabanlı derin öğrenme yapılarını kullanmaya başlamışlardır.

Günümüzde artık araştırmacıların dışında endüstri sektörü de doğrudan derin öğrenme ve yapay zeka kavramlarını kullanmaya başlamışlardır. Özellikle üretim sektöründe kalite kontrol aşamasında insanlar tarafından yapılan görevlerin yapay zeka tabanlı robotlar kullanılarak gerçekleştirildiği görülmektedir. Pandemi döneminde üretim sektöründe işletmelerde yaşanan bulaş riski gibi olumsuzlukları ortadan kaldırmak ve üretmeye aralıksız devam edebilmek için de derin öğrenme tabanlı robotlara ihtiyaç duyulduğu görülmektedir. Bu ihtiyaçtan

hareketle önmüzdeki yıllarda derin öğrenme kavramını daha sık duyacağımız aşikardır.

Derin öğrenmenin yazılım sektöründe bu kadar önemli bir noktaya gelmesinin yanında bazı dezavantajlarında bulunmaktadır. Burada önemli sayılabilen iki önemli konu bulunmaktadır. Bunlardan ilki derin öğrenme tabanlı algoritmaların yüksek donanım tabanlı bilgisayarlara ihtiyaç duymasıdır. Birkaç yıl öncesindeki kullanılan donanım seviyesi ile günümüzde kullanılan donanım seviyesi arasında açık ara bir fark bulunmaktadır. Bu nedenle gelinen teknolojik gelişmeler ışığında bu sorun tam anlamıyla olmasa da genel olarak çözülmeye başlandığı görülmektedir. Buna ek olarak konuya ilgilenen yazılım sektöründeki önemli bazı firmalarda internet ortamında derin öğrenme yazılımlarının geliştirilebilmesi için açık kaynak servisleri sağlamışlardır. Bu servislerin oluşturulmasındaki ana sebep yüksek donanım tabanlı bilgisayara duyulan ihtiyacın azaltılmak istenmesidir.

Konu ile ilgili yaşanan bir diğer dezavantajda derin öğrenme tabanlı algoritmaların iç yapısında kullanılan bazı katmanların üretmiş olduğu değerlerin tam olarak açıklanamamasıdır. Bu durum genel olarak birçok yapay zeka algoritmasında da görülmektedir. Bu nedenle kitabın ilk bölümünde derin öğrenme algoritmalarının içerisinde kullanılarak katmanlar ve bu katmanların görevleri kısaca anlatılmıştır. Bu anlatım sayesinde temel seviyedeki birçok araştırmacı, katman yapılarını özümseyerek kendine özgün derin öğrenme mimarisi oluşturabilecektir. Kitabın ikinci bölümünde ise yazılım sektöründeki firmaların gelişmiş birçok uygulamada kullanmış olduğu kompleks

yapıdaki derin öğrenme modelleri, örnek veriler ile MATLAB ortamında anlatılmıştır. Bu sayede hazırlanan kitap temelden ileri seviye bir araştırmacıya hitap edebilecek niteliktedir. Bu yönyle başta Bilgisayar Teknolojileri, Elektrik-Elektronik, Mekatronik, Makine olmak üzere, İktisat, Ekonomi gibi sosyal bilimler alanlarında da derin öğrenme tabanlı güncel algoritmaları kullanarak uygulama geliştirmek isteyen araştırmacılara kaynak bir kitap niteliğindedir.

Kitap aynı zamanda yabancı dil konusunda problem yaşayan tüm araştırmacılara hitap edebilecek yapıdadır. Bu nedenle hazırlanan kitabın sektörde, akademide ve derin öğrenme konularında meraklı okuyuculara da yararlı olmasını temenni ediyoruz.

Saygılarımla...

Dr. Öğr. Üyesi. Sedat METLEK

Öğr. Gör. Dr. Halit ÇETİNER

Haziran, 2021

İÇİNDEKİLER

| | |
|--|----|
| ÖNSÖZ..... | i |
| İÇİNDEKİLER..... | v |
| ŞEKİLLER DİZİNİ | i |
| TABLOLAR DİZİNİ | i |
| BÖLÜM 1 - GİRİŞ | 1 |
| 1.1. Yapay Zekâ | 2 |
| 1.2. Makine Öğrenmesi | 4 |
| 1.3. Derin Öğrenme | 6 |
| 1.3.1. Giriş katmanı | 8 |
| 1.3.2. Konvolüsyon katmanı | 8 |
| 1.3.3. Aktivasyon katmanı..... | 11 |
| 1.3.4. Havuzlama katmanı..... | 13 |
| 1.3.5. Ezberleme (Dropout) katmanı | 14 |
| 1.3.6. Tam bağlantı (FullConnected, FC) katmanı..... | 15 |
| 1.3.7. Sınıflandırma (Classification) katmanı | 16 |
| BÖLÜM 2 - DERİN ÖĞRENME MİMARİLERİ | 19 |
| 2.1. Derin Öğrenme Mimarilerinin Alt Yapısı | 19 |
| 2.2. SQUEEZE.NET Mimarisi..... | 21 |
| 2.3. MOBILENET V2 Mimarisi | 23 |
| 2.3.1. Derinlemesine konvolüsyon (Depthwise convolution, DC) | |
| | 25 |

| | |
|--|-----------|
| 2.3.2. Derinlemesine ayrılabılır konvolüsyon (Depthwise seperable convolution, DWC) | 27 |
| 2.3.3. Noktasal konvolüsyon (Pointwise convolution, PWC) 28 | |
| 2.3.4. MobileNetV2 hesaplama maliyeti..... | 30 |
| 2.4. RESNET50 Mimarisi | 31 |
| BÖLÜM 3 - MATLAB UYGULAMALARI..... | 36 |
| 3.1. SQUEEZE.NET | 36 |
| 3.1.1. Sistemin komut satırı ile test edilmesi..... | 48 |
| 3.1.2. Ön eğitilmiş ağdan öznitelik elde etme | 49 |
| 3.2. MOBILENET V2 | 51 |
| 3.2.1. Sistemin komut satırı ile test edilmesi..... | 61 |
| 3.2.2. Ön eğitilmiş ağdan öznitelik elde etme | 61 |
| 3.3. RESNET..... | 63 |
| 3.3.1. Sistemin komut satırı ile test edilmesi..... | 71 |
| 3.3.2. Ön eğitilmiş ağdan öznitelik elde etme | 72 |
| KAYNAKÇA | 74 |
| ÖZGEÇMIŞLER..... | 78 |

ŞEKİLLER DİZİNİ

| | |
|--|----|
| Şekil 1: Veri Madenciliği ve Sınıfları | 1 |
| Şekil 2: Klasik Öğrenmeye Karşın Makine Öğrenmesi | 5 |
| Şekil 3: Klasik Derin Öğrenme Mimarisi (Architecture of Deep Learning) | 7 |
| Şekil 4: Kernel ile Konvolüsyon İşlemi | 9 |
| Şekil 5: Konvolüsyon Katmanı (Metlek and Kayaalp 2020) | 10 |
| Şekil 6: Havuzlama Katmanı (Metlek and Kayaalp 2020)..... | 13 |
| Şekil 7: (a) Ezberleme Yapmış Ağ Modeli, (b) Ezberleme Katmanı Kullanılmış Ağ Modeli..... | 15 |
| Şekil 8: Tam Bağlantı Katmanı | 16 |
| Şekil 9: Squeeze.Net Mimarisi..... | 22 |
| Şekil 10: Fire Katmanı Yapısı | 22 |
| Şekil 11: Standart Konvolüsyon İşlemi (SC, (Bai, Zhao, and Huang 2018)) | 26 |
| Şekil 12: Derinlemesine Konvolüsyon (Depth-wise convolution, (Bai, Zhao, and Huang 2018))..... | 27 |
| Şekil 13: Derinlemesine Ayrılabilir Konvolüsyon (Depthwise Seperable Convolution)..... | 28 |
| Şekil 14: Noktasal Konvolüsyon (Pointwise Convolution) | 29 |
| Şekil 15: Darboğaz (bottleneck)..... | 29 |
| Şekil 16: RBB-1 Yapısı..... | 33 |
| Şekil 17: RBB2 Yapısı | 33 |
| Şekil 18: RESNET 50 Mimarisi | 35 |
| Şekil 19: Örnek Eğitim Görüntüleri | 38 |

| | |
|---|----|
| Şekil 20: Komutlar | 38 |
| Şekil 21: DeepNetworkDesigner Ekran Görüntüsü | 39 |
| Şekil 22: Mimari Seçimi | 40 |
| Şekil 23: Import Image Data | 40 |
| Şekil 24: Dataaugment | 41 |
| Şekil 25: Import Edilen Eğitim Verileri | 42 |
| Şekil 26: Mimariye Ait Varsayılan Ayarlar | 43 |
| Şekil 27: Konvolüsyon Katmanı Değişikliği | 43 |
| Şekil 28: Layer Library Classificationlayer | 44 |
| Şekil 29: Eğitim Ayarları | 45 |
| Şekil 30: Train Butonu | 46 |
| Şekil 31: Eğitim Sonucu | 46 |
| Şekil 32: Eğitilmiş Ağın Kayıt İşlemi | 47 |
| Şekil 33: İşlem Sonucu | 48 |
| Şekil 34: Mimari Kontrol Sonucu | 49 |
| Şekil 35: Kurulum Ekranı 1 | 52 |
| Şekil 36: Kurulum Ekranı 2 | 52 |
| Şekil 37: Kurulum Ekranı 3 | 53 |
| Şekil 38: Kurulum Ekranı 4 | 53 |
| Şekil 39: MobileNetV2 Tasarım Ekranı | 54 |
| Şekil 40: ImageInputLayer ‘in Düzenlenmesi | 55 |
| Şekil 41: Fullyconnectedlayer Katmanı Düzenlenmesi | 55 |
| Şekil 42: Classificationlayer‘in Yeniden Düzenlenmesi | 56 |
| Şekil 43: Veri Seti Ekleme | 57 |
| Şekil 44: Veri Seti Ekleme | 58 |
| Şekil 45: Dataaugment | 59 |

| | |
|--|----|
| Şekil 46: Import Edilen Eğitim Verileri | 59 |
| Şekil 47: Training Options Ayarları..... | 60 |
| Şekil 48: Simülasyon Sonucu..... | 60 |
| Şekil 49: MATLAB Araç Kutusu | 63 |
| Şekil 50: RESNET50 Kurulum Ekranı | 64 |
| Şekil 51: Kurulum Ekranı | 65 |
| Şekil 52: Çalışmada Kullanılacak Verilerin Yüklenmesi..... | 65 |
| Şekil 53: Örnek Veri Setindeki Görüntüleri Yükleme | 66 |
| Şekil 54: Yüklenen Veriler ve Özellikleri | 67 |
| Şekil 55: Imageinput Katmanını Revize Etmek | 68 |
| Şekil 56: Mimarının Revize Edilmesi | 69 |
| Şekil 57: Eğitim Ayarlamaları..... | 70 |
| Şekil 58: Sonuç Ekranı | 70 |
| Şekil 59: Eğitilmiş Ağın Çıkartılması | 71 |

TABLOLAR DİZİNİ

| | |
|--|----|
| Tablo 1: Makine Öğrenme Modelleri (Qiu et al. 2016) | 6 |
| Tablo 2: Aktivasyon Fonksiyonları..... | 12 |
| Tablo 3: Konvolüsyonel Sinir Ağındaki Katmanlar ve Özellikleri.... | 17 |
| Tablo 4: MobileNetV2 | 30 |

BÖLÜM 1

GİRİŞ

Çok büyük ve farklı sayısal veriler içeren veri yiğinlarının klasik algoritma ve teknikler kullanarak analiz edilmesi oldukça zordur (Najafabadi et al. 2015). Bu sorunun çözülebilmesi adına son yıllarda veri bilimi hızlı bir gelişim sağlamıştır. Özellikle her gün daha çok büyüyen dijital verilerde düşünüldüğünde, veri biliminin önemi bir adım daha öne çıkmaktadır. Veri biliminin bu şekilde hızlı gelişmesinin sonucunda veri madenciliği, yapay zekâ, makine öğrenmesi ve derin öğrenme kavramları ortaya çıkmıştır. Büyük miktardaki verilerden anlamlı örüntüler elde etmek için veri madenciliği ile sürekli farklı algoritmalar geliştirilmektedir. Bu anlamda yapay zekâ, makine öğrenmesi ve derin öğrenme kavramları da veri madenciliği ile paralel bir şekilde gelişmeye devam etmektedir.



Şekil 1: Veri Madenciliği ve Sınıfları

Genel olarak veri madenciliğinden derin öğrenmeye doğru giden hiyerarşi Şekil 1'de gösterilmiştir.

Bu anlamda ilk olarak yapay zeka, makine öğrenmesi ve derin öğrenme kavramları detaylı olarak aşağıda sunulmuştur. Sonrasında veri bilimi alanındaki en son geliştirilen kavumlardan birisi olan derin öğrenme ile ilgili MATLAB ortamında örnek uygulamalar geliştirilmiştir.

1.1. Yapay Zekâ

Yapay zekâ kavramı ilk olarak 1950'li yıllarda bazı bilim insanlarının bilgisayarların insanlar gibi düşünüp düşünemeyeceği sorusunu ortaya atması ile çıkmıştır. Bu sorunun cevabı günümüzde hala araştırma konusudur. Bu nedenle yapay zekâ kavramı makine öğrenmesi ve derin öğrenmeyi de kapsayan bir yapıyı oluşturmaktadır (Chollet 2018).

Yapay zekâ terimi genel olarak literatürde bir makinenin insanların öğrenme ve problem çözme gibi bilişsel yapısını örnek alarak geliştirilen algoritmala verilen isimdir. İnsanların örnek alınan bilişsel becerileri ise sınıflandırma, konuşma, tanıma, akıl yürütme, planlama, öğrenme ve doğal dil işleme gibi becerileridir (Ongsulee 2017). Bu bilişsel becerilerin bazlarının örnek alınarak geliştirilmesi sonucunda 1950'li yıllarda yapay zekâ algoritmaları ile bilgisayarlar satranç oynamayı da başarmıştır.

Yapay zekâ, bilgisayar bilimi, mühendislik, biyoloji, psikoloji, matematik, istatistik, mantık, felsefe, işletme ve dilbilimi gibi birçok disiplini etkileyen ve bunlardan da doğrudan etkilenen bir yapıyı

oluşturmaktadır (Buchanan 2005; Kumar et al. 2016). Bunun sonucunda tiptan (Jiang et al. 2017; Macrae 2019; Rong et al. 2020) askeriye (Maas 2019; Morgan et al. 2020; Wang, Liu, et al. 2020), ekonomiden (Hajkowicz et al. 2019; Huang, Rust, and Maksimovic 2019; Sion 2018) meteorolojiye (Anandharajan et al. 2016; Kosovic et al. 2020; McGovern et al. 2017) kadar birçok alanda uygulamaları görülmektedir.

Bu kadar çok farklı alanda kullanılması etkileyici olmakla birlikte, bu teknolojilerin belirli alanlara uygun hale getirilmesi yıllarca süren araştırmalara ve uzmanlaşmış ekiplere bağlıdır. Uzun süren bu araştırmaların sonuçlarını genel olarak araştırmak için 2015 yılında bir rapor hazırlanmış ve kamuoyu ile paylaşılmıştır. Paylaşılan bu rapora göre yapay zekâ alanında yapılan çalışmalar aşağıda sunulan sekiz ana başlık altında toplanmış ve raporlanmıştır (Stone et al. 2016).

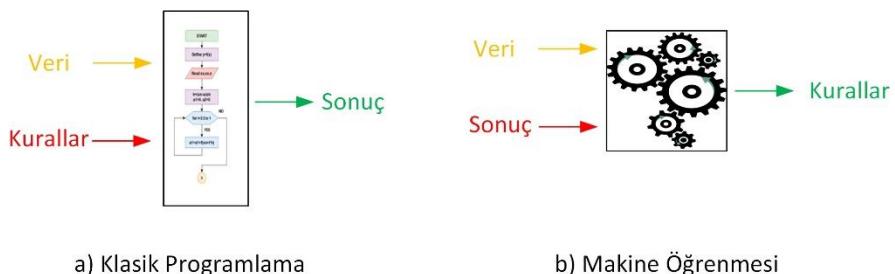
Yapay zeka alanındaki raporlanan çalışma alanları;

1. Ulaşım (Transportation);
2. Servis Robotları (Service robots);
3. Sağlık Hizmeti (Healthcare);
4. Eğitim (Education);
5. Düşük Gelirli Topluluklar (Low-resource communities);
6. Güvenlik ve Kamu Güvenliği (Public safety and security);
7. İstihdam ve İşyeri (Employment and workplace);
8. Eğlence (Entertainment).

Bu rapordan da anlaşılacağı üzere önmüzdeki yıllarda yapay zekâ uygulamalarını hayatımızın birçok noktasında sıkılıkla duyacağız. Özellikle yukarıda raporda belirtilen insan yaşamını etkileyen tüm kişisel ve sosyal alanlarda yapay zekâ uygulamalarının yaşamımızın bir parçası haline geleceği aşikardır.

1.2. Makine Öğrenmesi

Klasik programlama bilgisayarda herhangi bir konu ile alakalı veriler ve kurallar kullanılarak gerçekleştirilen programlama türüdür. Bu programlama türünde sonuç ile veriler arasında sıkı sıkıya bağlı bir ilişki mevcuttur. Çıkacak sonuç ve oluşabilecek her hangi bir durum ile ilgili algoritma geliştirmeden önce olabildiğince geniş bir perspektiften bakarak tüm durumların göz önünde bulundurulması gerekmektedir. Fakat bazı durumlarda veriler ile çıkan sonuç arasında doğrudan lineer ya da non-lineer bir ilişki kurulamamakta ve bunun sonucunda da kurallar oluşturulamamaktadır. Kurallar olmadığı için de böyle bir durumda klasik programlama ile soruna çözüm bulunamamaktadır. Bu tarz sorumlara çözüm getirebilmek adına geliştirilen yapay zekâ algoritmaları makine öğrenme algoritması olarak adlandırılmaktadır. Bu algoritmaların, klasik programlamadan farkı Şekil 2'de gösterilmiştir.



Şekil 2: Klasik Öğrenmeye Karşın Makine Öğrenmesi

Son yıllarda makine öğrenme algoritmalarının veri ile sonuç arasında kurabildiği bağıntıdan dolayı ticari ve askeri birçok alanda tercih nedeni olmuştur.

Literatürde genel olarak makine öğrenme algoritmaları denetimli öğrenme, denetimsiz öğrenme ve pekiştirmeli öğrenme olarak üç alt bölüme ayrılmıştır. Denetimli öğrenme modelinde, etiketli giriş ve çıkış verileri eğitim aşamasında birlikte kullanılmaktadır.

Denetimsiz öğrenme modelinde ise etiketli eğitim verilerinin sadece çıkış değeri kullanılarak sistemin doğru çalışıp çalışmadığı test edilmektedir. Pekiştirmeli öğrenmede ise gerçekleştirilen işlem sonucunda elde edilen değerin doğru ya da yanlış olması durumu geri bildirimler yoluyla sağlanmaktadır. Bu üç temel makine öğrenme modelinde dayanarak, farklı sorunlar için geliştirilmiş birçok makine öğrenme algoritması bulunmaktadır (Bekkerman et al. 2003; Jones 2014; Langford and Schapire 2005). Genel olarak bu algoritmalar Tablo 1'de kısaca kullanım alanlarına göre karşılaştırmalı olarak özetlenmiştir.

Tablo 1: Makine Öğrenme Modelleri (Qiu et al. 2016)

| Öğrenme Tipleri | Veri işleme yöntemi | Uzaklık normları | Öğrenme algoritmaları |
|--------------------|---|---------------------------------|------------------------|
| Danışmanlı öğrenme | <ul style="list-style-type: none"> -Sınıflandırma -Regresyon -Varsayımlı | Hesaplamlı sınıflandırıcılar | Destek vektör makine |
| | | İstatistiksel sınıflandırıcılar | Naive Bayes |
| | | | Gizli Markov model |
| | | | Bayes ağları |
| | <ul style="list-style-type: none"> -Kümemeleme -Tahmin | Bağlantılı sınıflandırıcılar | Sinir ağları |
| | | Parametrik | K - ortalamalar |
| | | | Gaussian model |
| | | Parametrik olmayan | Dirichlet işlem modeli |
| | | | X-ortalamalar |
| Pekürtmeli öğrenme | <ul style="list-style-type: none"> -Karar verme | Model-free | Q öğrenme |
| | | | R öğrenme |
| | | Model tabanlı | TD öğrenme |
| | | | Sarsa öğrenme |

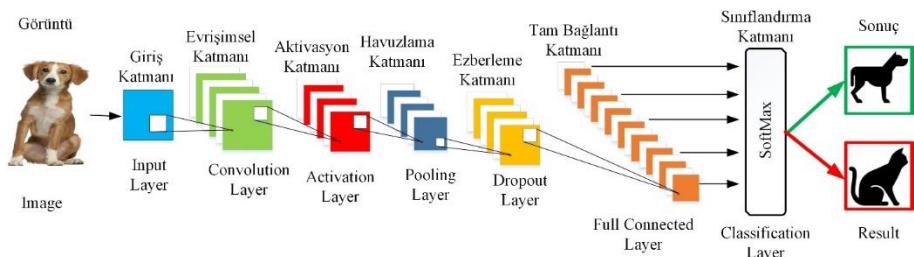
Tablo 1'den de anlaşılacağı üzere, denetimli ve denetimsiz öğrenme modelleri veri analizine odaklanırken, karar verme problemlerinde ise pekiştirmeli öğrenme modeli kullanılmaktadır.

1.3. Derin Öğrenme

Günümüzde makine öğrenme algoritmalarının ulaştığı en son nokta derin öğrenme algoritmalarıdır. Derin öğrenmede geleneksel öğrenme yöntemlerinin aksine hiyerarşik bir yapılanma söz konusudur. Bu yöntemde özniteliklerin otomatik olarak elde edilmesi için denetimli ve

denetimsiz öğrenme yöntemleri de kullanılabilmektedir (Yu and Deng 2010). Derin öğrenme algoritmalarının genel mantığı konvolüsyon tabanlı bir yapay sinir ağı modeline dayanmaktadır.

Şekil 3’de gösterilen derin öğrenme mimarisinin temelini oluşturan konvolüsyonel sinir ağı, farklı kombinasyonlar ile bir araya gelerek modern derin öğrenme mimarilerinin temelini oluşturmaktadır. Özellikle günümüzde donanım kapasitesinin artması sonucunda katman ve bağlantı sayısı değiştirilerek, farklı isimler altında yeni derin öğrenme mimarileri geliştirilmektedir.



Şekil 3: Klasik Derin Öğrenme Mimarisi (Architecture of Deep Learning)

Şekil 3’te gösterilen klasik bir derin öğrenme mimarisinde sırayla giriş, konvolüsyon, aktivasyon, havuzlama, ezberleme, tam bağlantı ve sınıflandırma katmanları bulunmaktadır. Bu katmanların görevleri aşağıda detaylı olarak sunulmakla beraber, bölüm sonunda Tablo 3’de özet olarak verilmiştir.

1.3.1. Giriş katmanı

Giriş katmanı birçok yapay zekâ algoritmasında olduğu gibi verilerin ağa sunulmak üzere standart hale getirilip, ağa sunulduğu katmandır. Genellikle bu katmanda kullanılan mimariye ve veri setine bağlı olmak üzere normalizasyon işlemi, Denklem 1 ile gerçekleştirilmektedir.

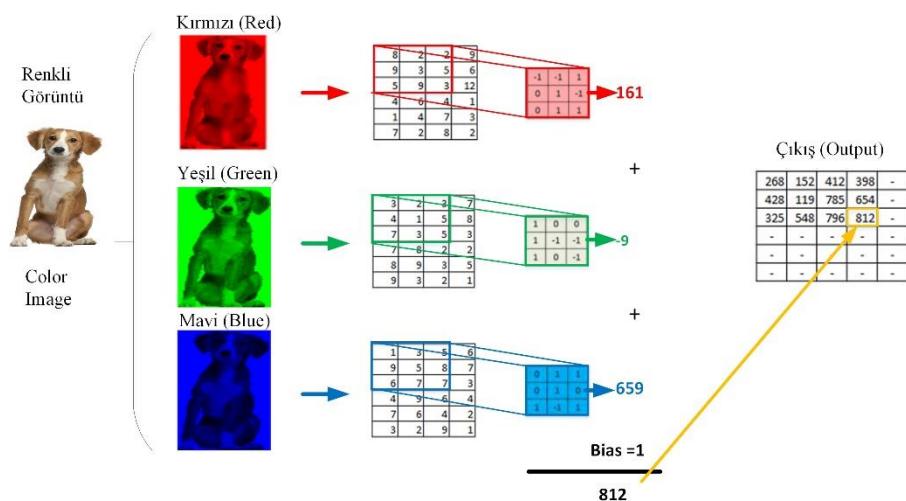
$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \quad (1)$$

Denklem 1'deki x_{norm} normalize edilmiş veriyi, $min(x)$ kullanılan veri seti içerisindeki en küçük değeri, $max(x)$ veri seti içerisindeki en büyük değeri ve x ise normalize edilecek veriyi ifade etmektedir.

1.3.2. Konvolüsyon katmanı

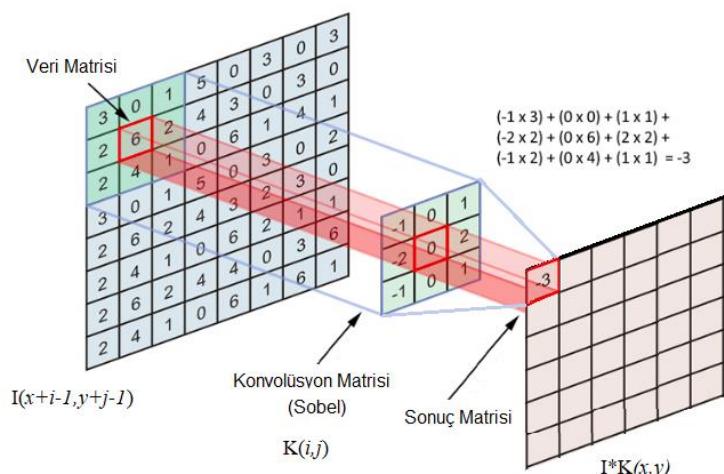
Konvolüsyon katmanında, giriş verileri üzerinde dolaştırılacak bir dizi filtre (kernel) kullanılmaktadır. Yapılacak uygulamaya bağlı olarak kullanılacak filtrelerin içerikleri, büyüklükleri ve sayısında değişmektedir. Uygulamada kullanılacak olan filtrelerin boyutu, kullanılan verinin boyutundan küçük olmak zorundadır. Veri üzerinde yatay ve dikey olarak dolaştırılan filtreler, bulunduğu yerdeki veri değerleri ile çarpılarak yeni veri seti elde edilmektedir. Bu şekilde, seçilen filtreler tüm veri üzerinde dolaştırılmaktadır.

Örneğin renkli bir görüntü derin öğrenme mimarilerinde giriş verisi olarak kullanılacaksa, kullanılan verinin yapısından dolayı üç boyutlu bir matris giriş verisini oluşturmaktadır. Bu durumda konvolüsyon katmanı renkli görüntüyü oluşturan her bir matris üzerinde dolaştırılarak yeni iki boyutlu bir veri matrisi elde edilmektedir. Bu durum ayrıntılı olarak Şekil 4'de gösterilmiştir.



Şekil 4: Kernel ile Konvolüsyon İşlemi

Şekil 4'de görüldüğü üzere renkli görüntünün her bir boyutu üzerinde dolaştırılan filtrelerin toplamına ek olarak bias değeri de eklenmektedir. Bias değerinin eklenmesi kullanılan mimariye göre farklılık gösterebilmektedir. Matematiksel olarak aynı işlem Şekil 5 üzerinde Denklem 2 uygulanarak da gerçekleştirilmektedir. Şekil 5'de gösterildiği üzere, $(x+i-1), (y+j-1)$ boyutundaki I veri matrisi üzerinde $[i,j]$ boyutunda bir K matrisi (filtresi) dolaştırılarak, $[x,y]$ boyutunda yeni bir I^*K veri matrisi elde edilmektedir.



Şekil 5: Konvolüsyon Katmanı (Metlek and Kayaalp 2020)

$$(I * K)_{xy} = \sum_{i=1}^h \sum_{j=1}^w K_{ij} \cdot I_{x+i-1, y+j-1} \quad (2)$$

Görüntü üzerinde dolaştırılacak filtrenin, ağın başarısına doğrudan etkisi bulunmaktadır. Örneğin ağ üzerinde dolaştırılacak matris çok büyük boyutlu ve büyük sayılabilecek rakamlardan oluşması durumunda, ağın eğitilmesi çok uzun sürebilir, hatta tercih edilen rakamlara göre hata ile karşılaşılma durumu da söz konusu olabilmektedir (Pang et al. 2017).

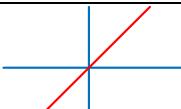
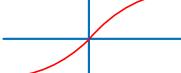
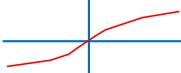
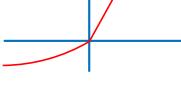
1.3.3. Aktivasyon katmanı

Aktivasyon katmanı edilen değerleri belirli aralıklara indirmek için kullanılmaktadır. Coğu zamanda elde edilen negatif değerleri pozitife çevirmek için kullanılmaktadır (Arora et al. 2016). Literatürde aktivasyon katmanında genellikle ReLu, Sigmoid, Adım (Step) gibi aktivasyon fonksiyonları kullanılmaktadır. Şekil 4'te gösterilen renkli görüntünün her bir boyutuna Denklem 2 uygulandıktan sonra elde edilen veri matrisine Denklem 3 uygulanarak aktivasyon işlemi gerçekleştirilmektedir. Birçok uygulamada aktivasyon katmanı, konvolüsyon katmanından sonra gelmesine rağmen, tasarımcının istedüğüne bağlı olarak farklı katmanlardan sonra uygulanabilmektedir.

$$conv(I, K)_{xy} = \alpha \left(b + \sum_{i=1}^h \sum_{j=1}^w \sum_{k=1}^d K_{ij} \cdot I_{x+i-1, y+j-1, k} \right) \quad (3)$$

Denklem 3'de, kullanılan d değeri görüntünün boyutlarını ifade etmektedir. b değeri ise biası ifade etmektedir. Bias değeri genellikle isteğe bağlı olarak kullanılmaktadır. α değeri ise aktivasyon fonksiyonunu ifade etmektedir. Aktivasyon fonksiyonu olarak da genellikle Tablo 2'deki TanH ve Relu fonksiyonları kullanılmaktadır.

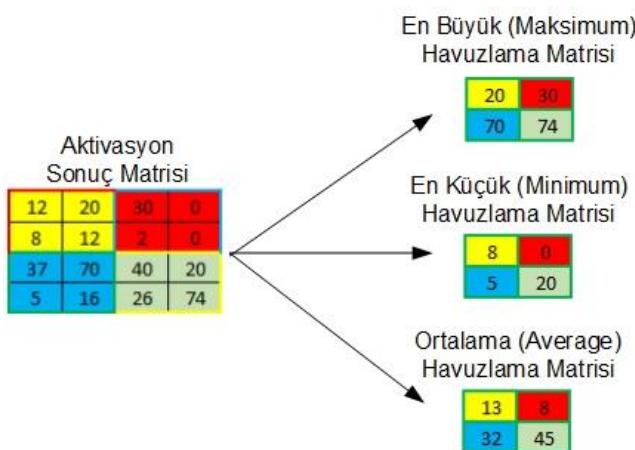
Tablo 2: Aktivasyon Fonksiyonları

| İsim | Grafik | Denklem $f(x)$ | Türev $f'(x)$ |
|-------------|---|---|---|
| Identity |  | $= x$ | $= 1$ |
| Binary Step |  | $= \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ | $= \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$ |
| Logistic |  | $= \frac{1}{1 + e^{-x}}$ | $= f(x)(1 - f(x))$ |
| TanH |  | $= \tanh(x)$ $= \frac{2}{1 + e^{-2x}} - 1$ | $= 1 - f(x)^2$ |
| ArcTan |  | $= \tan^{-1}(x)$ | $= \frac{1}{x^2 + 1}$ |
| ReLU |  | $= \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $= \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| PRelu |  | $= \begin{cases} ax & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $= \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| ELU |  | $= \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $= \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| SoftPlus |  | $= \log_e(1 + e^x)$ | $= \frac{1}{1 + e^{-x}}$ |

1.3.4. Havuzlama katmanı

Literatürde birçok uygulamada genellikle aktivasyon katmanından sonra havuzlama katmanı kullanılmaktadır. Havuzlama katmanının ana gayesi verileri daha küçük boyutlara indirmektir. Bu indirgeme işlemi esnasında kullanılan havuzlama fonksiyonlarına da bağlı olarak veri kayıpları yaşanabilmektedir (Hinton 2012). Bu nedenle havuzlama katmanı uygulandığında makul düzeyde veri kaybı yaşanacağı öngörülmelidir.

Literatürde birçok havuzlama fonksiyonu bulunmasına rağmen genellikle en büyük (max pooling) değer, en küçük (min pooling) değer ve ortalama (average pooling) değer fonksiyonları kullanılmaktadır. Kullanılan bu fonksiyonlar Şekil 6'da gösterildiği üzere aktivasyon katmanından elde edilen matris üzerinde dolaştırılarak yeni bir veri matrisi oluşturulmaktadır.

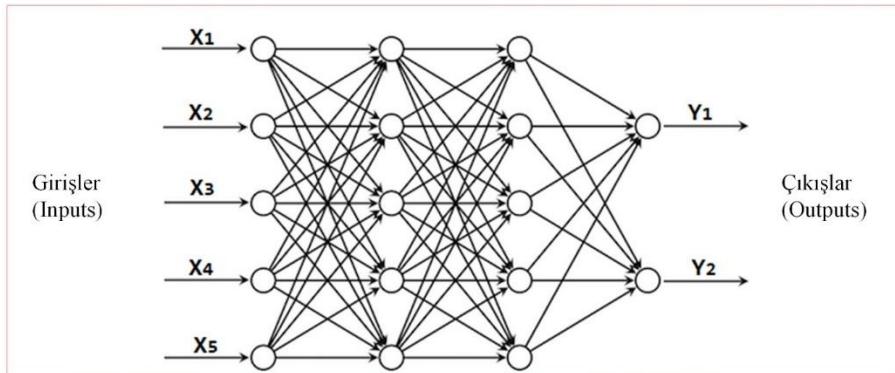


Şekil 6: Havuzlama Katmanı (Metlek and Kayaalp 2020)

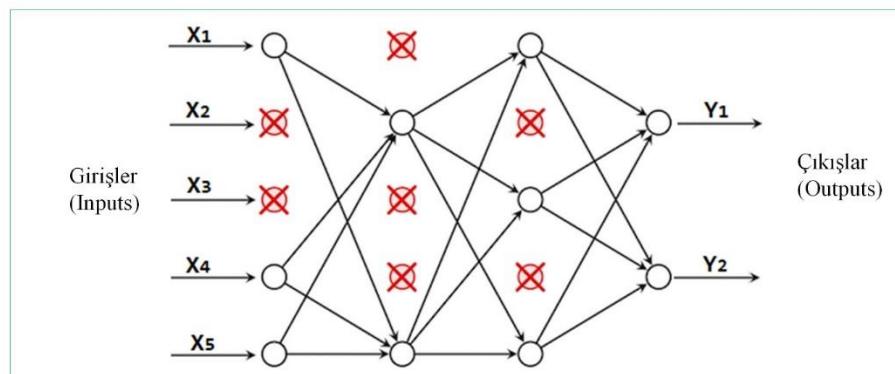
1.3.5. Ezberleme (Dropout) katmanı

Çok az veri temini ile gerçekleştirilmek istenen bazı yapay zekâ algoritmalarında eğitim esnasında sistem kullanılan verileri ezberleyemektedir. Bu nedenle tasarlanan ağ mimarisi içerisinde bu verilerin ağa unutturulması gerekmektedir. Yapay zekâ algoritmalarında bu işlem ezberleme katmanı ile gerçekleştirilmektedir. Bahse konu olan ezberleme katmanı bazı derin öğrenme modellerinde de kullanılabilmektedir. Bu nedenle Şekil 7'de çalışma prensibine kısaca değinilmiştir. Şekil 7(a)'da ezberleme yapmış bir ağ yapısı gösterilmiştir. Şekil 7(b)'de ise ezberleme katmanı kullanılan bir ağ yapısı gösterilmiştir. Literatürde genellikle derin öğrenme modellerinde ezberleme katmanı, havuzlama katmanı ile tam bağlantı katmanı arasında kullanılmaktadır.

Literatürdeki bazı uygulamalarda girişlerden bazlarının, ağıın çıkışını diğerlerine göre çok daha fazla etkilediği görülmüştür. Böyle bir durumda, ağıın girişindeki etki değeri küçük olan girişlerdeki değişimler algılanmamaktadır. Bunun sonucunda da ağıın çıkışını güçlü olan giriş parametresi şekillendirmektedir. Bu sorunu çözebilmek adına tasarlanan ağıın mimarisinin içerisinde ezberleme katmanı kullanılarak ağıın zayıf giriş parametrelerinin çıkışa etkisi de artırılabilmektedir.



a) unutma katmanı uygulanmamış (without dropout layer)

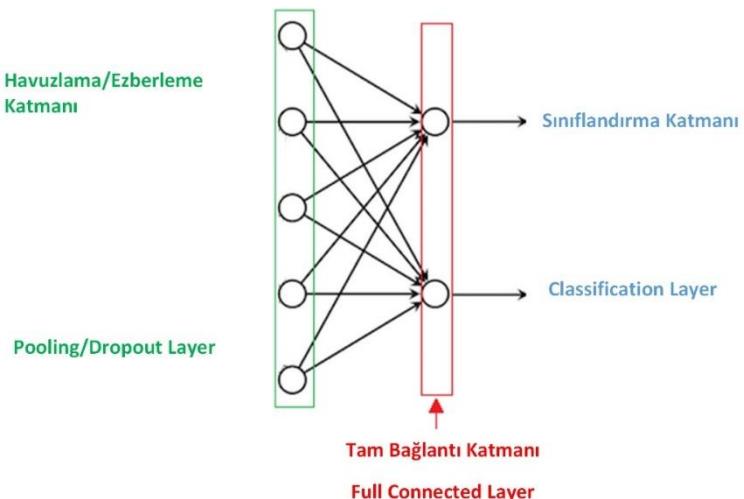


b) unutma katmanı uygulanmış (with dropout layer)

Şekil 7: (a) Ezberleme Yapmış Ağ Modeli, (b) Ezberleme Katmanı Kullandırılmış Ağ Modeli

1.3.6. Tam bağlantı (FullConnected, FC) katmanı

Tam bağlantı katmanı havuzlama veya ezberleme katmanından sonra kullanılabildeği gibi genellikle sınıflandırma katmanından önce de kullanılmaktadır. Tam bağlantı katmanına gelen veriler tek boyutlu bir vektöre dönüştürülerek sınıflandırma katmanına gönderilmektedir. Bu durum Şekil 8'de gösterilmiştir.



Şekil 8: Tam Bağlantı Katmanı

1.3.7. Sınıflandırma (Classification) katmanı

Genel olarak yapay zekâ algoritmalarında olduğu gibi derin öğrenme algoritmalarında da son katman sınıflandırma katmanıdır. Bu katman kendinden önceki tam bağlantı katmanından gelen verileri değerlendirerek ağıın çıkış değerini oluşturan katmandır.

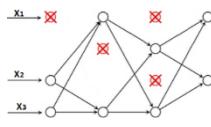
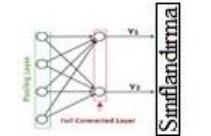
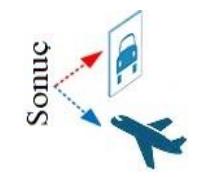
Tasarlanan ağıın çıkışında üretilen sonuç, sınıflandırma bilgisi olabileceği gibi eğri uydurma yöntemlerinde kullanılan bir polinomun katsayıları da olabilmektedir. Sınıflandırma katmanından elde edilecek değer sınıf bilgisi ise birçok farklı sınıflandırma fonksiyonu, bu katmanda kullanılabilmektedir. Literatürde genelde derin öğrenme uygulamalarında, olasılıksal bir hesaplama yöntemi olan SoftMax sınıflandırıcısı kullanılmaktadır. SoftMax sınıflandırıcısı ile olası her sınıf için 0-1 aralığında değerler üretilmektedir. Üretilen bu sınıf

değerlerinden 1'e en yakın olan değer, o girişin sınıf bilgisini oluşturmaktadır.

Bazı durumlarda bağlantı katmanından gelen öznitelikler, kullanılan algoritmaya bağlı olarak farklı bir sınıflandırma algoritmaları ile de sınıflandırılabilir olmaktadır. Yukarıda anlatılan katman yapısı kısaca Tablo 3'de özetlenmiştir.

Tablo 3: Konvolüsyonel Sinir Ağındaki Katmanlar ve Özellikleri

| Katmanlar ve Özellikleri | Görüntü | | | | | | | | | | | | | | | | | | | | |
|--|--|----|----|----|---|---|----|---|---|----|----|----|----|---|----|----|----|----|---|----|----|
| Giriş Katmanı (Input Layer); Konvolüsyonel sinir ağlarındaki ilk katmandır. Ağa sunulan veriler ilk olarak bu katmandan sunulur. Veriler daha öncesinde herhangi bir boyutlandırma işlemeye tabi tutulmuşsa bu katman olduğu gibi ağa verilebilir. Eğer verilerin boyutlarında belli bir standart yoksa ya da ağın girişinde uyulması gereken bir boyut varsa, bu katmanda boyutlandırma işlemi de gerçekleştirilir. | | | | | | | | | | | | | | | | | | | | | |
| Konvolüsyon Katmanı (Convolution Layer); Kendisinden önceki katmandan gelen veriler üzerinde, boyutlarını ve özelliklerini geliştiricilerin belirlediği $m \times n$ boyutlarındaki filtrelerin dolaştırılarak yeni öznitelik matrislerinin elde edildiği katmandır. | | | | | | | | | | | | | | | | | | | | | |
| Aktivasyon Katmanı (Activation Layer); Konvolüsyon katmanından elde edilen matris değerlerini, kullanılan algoritmaya bağlı olarak belirlenen aralığa getiren katmandır. Bunun için sigmoid, hiperbolik tanjant sigmoid, adım ve ReLU gibi farklı aktivasyon fonksiyonları kullanılmaktadır. | | | | | | | | | | | | | | | | | | | | | |
| Havuzlama Katmanı (Pooling Layer); Kendisinden önce gelen verileri daha küçük boytlardaki matrislere indirmek için kullanılan katmandır. Bu veri indirgeme işlemi ağına daha hızlı çalışmasını sağlamasına rağmen, bazı durumlar için veri kaybına neden olabilmektedir. Veri indirgeme için ortalama, minimum ve maksimum değer matrisleri gibi farklı birçok matris de kullanılabilmektedir. | <table border="1"> <tr><td>12</td><td>20</td><td>30</td><td>0</td></tr> <tr><td>8</td><td>12</td><td>2</td><td>0</td></tr> <tr><td>37</td><td>70</td><td>40</td><td>20</td></tr> <tr><td>5</td><td>16</td><td>26</td><td>74</td></tr> </table> <table border="1"> <tr><td>13</td><td>8</td></tr> <tr><td>32</td><td>45</td></tr> </table> | 12 | 20 | 30 | 0 | 8 | 12 | 2 | 0 | 37 | 70 | 40 | 20 | 5 | 16 | 26 | 74 | 13 | 8 | 32 | 45 |
| 12 | 20 | 30 | 0 | | | | | | | | | | | | | | | | | | |
| 8 | 12 | 2 | 0 | | | | | | | | | | | | | | | | | | |
| 37 | 70 | 40 | 20 | | | | | | | | | | | | | | | | | | |
| 5 | 16 | 26 | 74 | | | | | | | | | | | | | | | | | | |
| 13 | 8 | | | | | | | | | | | | | | | | | | | | |
| 32 | 45 | | | | | | | | | | | | | | | | | | | | |

| | |
|---|--|
| <p>Ezberleme Katmanı (Dropout Layer); Ağın az sayıda veriye sahip olduğu durumlarda aşırı öğrenme riski bulunmaktadır. Ağın eğitim setinde, ezberleme yapmasının önüne geçmek için kullanılan katmandır.</p> |  |
| <p>Tam Bağlantı Katmanı (Full Connected Layer); Kendisinden önceki katmanlardan gelen tüm verileri alarak tek boyutlu bir dizi matrisine dönüştüren katmandır. Bu katmandan çıkan veriler doğrudan sınıflandırma katmanına giriş olarak aktarılır.</p> |  |
| <p>Sınıflandırma Katmanı (Classification Layer); Derin öğrenme mimarisinin son katmanıdır. Tam bağlantı katmanından gelen veriler değerlendirilerek ağın çıkışının oluşturduğu katmandır. Burada, sınıflandırma bilgisinin yanında gerçekleştirilen uygulamaya bağlı olarak herhangi bir eğrinin katsayıları da hesaplanabilmektedir. Genellikle derin öğrenme algoritmalarının sınıflandırma katmanında SoftMax sınıflandırıcısı kullanılmaktadır.</p> |  |

BÖLÜM 2

DERİN ÖĞRENME MİMARİLERİ

Kitabın bu bölümünde derin öğrenme mimarilerinin teorik altyapısı anlatılmıştır. Uygulamalara geçilmeden önce, konuya hâkimiyet sağlanması açısından bu bölümün detaylı olarak incelenmesinde fayda vardır.

2.1. Derin Öğrenme Mimarilerinin Alt Yapısı

Klasik programlama ile büyük miktardaki verilerin analizini gerçekleştirebilmek zor ve uzun bir süreçtir. Bu nedenle günümüzde büyük miktardaki verilerin analizini yaparak doğru sonuçlar üretebilen birçok farklı yapay zekâ algoritması geliştirilmiştir. Geliştirilen bu sistemlerin ortak özelliği, veri ile sonuç arasında bir bağıntı oluşturarak kurallar üretmesidir. Bu kural kavramı çoğu zaman eğitilen ağın ağırlık katsayılarını ve ağıın yapısını ifade etmektedir.

Günümüzde derin öğrenme bilgisayarlı görme, doğal dil işleme, konuşma tanıma, medikal görüntü işleme, borsa tahmini, enerji tüketim tahmini gibi birçok alanda uygulanmaktadır. Birinci bölümde de dephinildiği üzere, makine öğrenme kendi içerisinde danışmanlı, danışmansız gibi birçok farklı türü bulunmaktadır. Bu türlerin genel özelliği herhangi bir konu ile ilgili veriler elde edildiğinde ilk olarak veriler içerisinde konuyu en iyi ifade edebilecek özniteliklerin çıkartılmasıdır. Bu öznitelik çıkarma işleminde tasarımcılar bazen veri

elde edilen konu ile ilgili uzman kişilerden yardım isterken, bazende öznitelik çıkarmak için geliştirilmiş özel algoritmaları kullanmaktadır.

Bu noktadan yola çıkarak konvolüsyonel sinir ağları mantığı ile örüntüden otomatik olarak öznitelik elde edebilecek algoritmalarla geliştirilmeye başlanmıştır. Bu bakış açısı klasik makine öğrenme kavramına yeni bir soluk getirerek derin öğrenme kavramının ortayamasına neden olmuştur.

Derin öğrenme, herhangi bir konu ile ilgili bir örüntüyü kullanarak doğrudan sınıflandırma, eğri uydurma gibi işlemleri gerçekleştiren bir tür makine öğrenme algoritmasıdır. Derin öğrenmede genellikle bir konvolüsyonel sinir ağı mimarisi kullanılır. Derin terimi, ağıdaki katmanların sayısını ifade etmektedir. Bu ağıda ne kadar çok katman olursa, ağıın o kadar derin olduğu söylenebilmektedir. Klasik yapay sinir ağları ya da konvolüsyonel sinir ağında yalnızca iki veya üç katman bulunurken, derin sinir ağlarında yüzlerce alt katman bulunabilmektedir. Klasik bir konvolüsyonel sinir ağında giriş katmanı konvolüsyon katmanı, havuzlama katmanı, aktivasyon katmanı, unutma katmanı, tam bağlantı katmanı ve sınıflandırma katmanı bulunmaktadır. Bu katmanların genel bağlantı yapısı birinci bölümde detaylı olarak anlatılmıştır.

Kitapta anlatılan tüm uygulamalar, Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz (4 CPUs), ~2.9GHz işlemci, 8 Gb Ram ve Intel(R) HD Graphics 620 ekran kartına sahip bir donanım üzerinde Windows 10

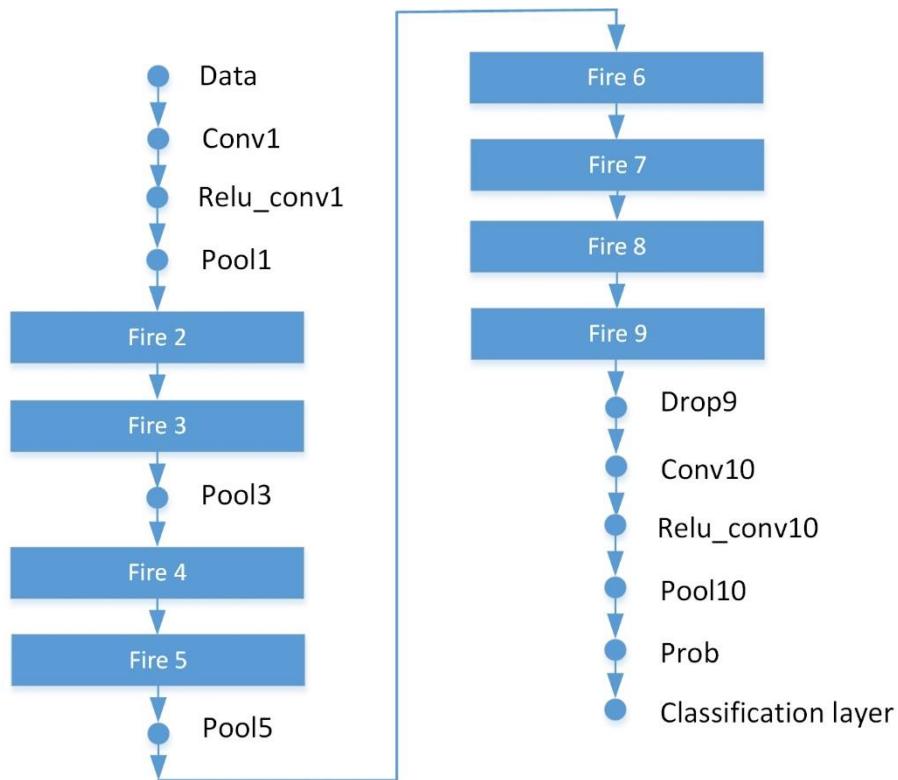
Pro 64-bit işletim sistemi ile MATLAB 2020b ortamında geliştirilmiştir.

Bu donanımdan da anlaşılacağı üzere çok yüksek düzeyde donanıma sahip olmayan bilgisayarlar üzerinde de derin öğrenme mimarileri çalıştırılabilir ve bu da internet ortamındaki sanal labaratuvarlarda uygulama geliştirme imkanı bulunmaktadır. Son kullanıcılar için de ön eğitme (PreTrained) işlemi tamamlanmış derin öğrenme mimarileri hazır olarak bulunmaktadır. Uygulayıcılar sadece bu ön eğitme işlemi tamamlanmış mimarileri kullanarak da sonuçlar üretmekte eder (Stone et al. 2016).

2.2. SQUEEZE.NET Mimarisi

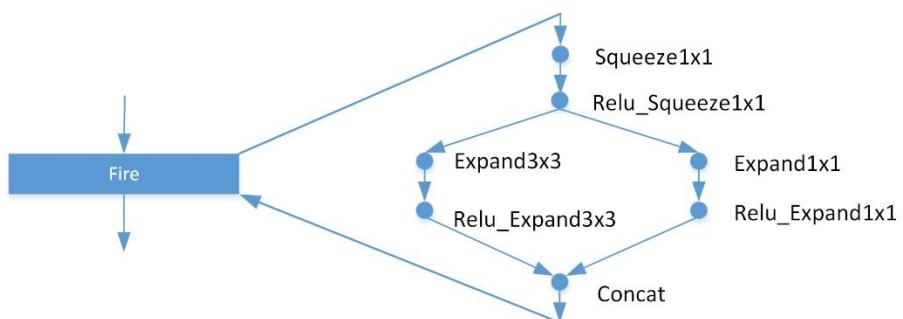
Squeeze.net mimarisi Tablo 3'te sunulan katmanların blok halinde bir araya getirilmesi ile oluşturulan bir yapıdır. Bu yapı Şekil 9'da gösterildiği üzere;

Data>>Conv1>>Relu_conv1>>Pool1>>Fire2>>Fire3>>Pool3>>Fire4>>Fire5>>Pool5>>Fire6>>Fire7>>Fire8>>Fire9>>Drop9>>Conv10>>Relu_conv10>>Pool10>>Prob>>Classification Layer yapılarından oluşmaktadır.



Şekil 9: SqueezeNet Mimarisi

Bu yapı içerisinde bulunan Fire katmanlarının içerikleri Şekil 10'da detaylı olarak sunulmuştur.



Şekil 10: Fire Katmanı Yapısı

Şekil 10'da bulunan Concat katmanı Fire yapısının iki farklı kolundan gelen değerleri birleştiren katmandır. Squeeze.net mimarisinde toplamda 68 tane katman bulunmaktadır. Bunlardan 18 adeti öğrenebilen katmandır (Matlab 2021). Bu mimari içerisinde bulunan fire katmanın tam bağlantı ve dense katmanına benzer bir görevi bulunmaktadır. Bu mimarinin en temel özelliği, boyut kapasitesini ve parametre sayısını azaltarak başarılı bir şekilde analiz yapabilmesidir. Squeeze.net mimari modeli, AlexNet mimarisinden yaklaşık 50 kat daha az parametre sayısına sahip bir model olmak ile beraber özellikle öğrenme süresi AlexNet mimarisine göre çok daha azdır (Iandola et al. 2016). Günümüzde bu nedenle çok fazla tercih edilmektedir.

2.3. MOBILENET V2 Mimarisi

Dünya genelinde gelişen iletişim teknolojilerinin bir sonucu olarak, birçok teknoloji hızlı bir şekilde yaygınlaşmaktadır. Bu duruma en güzel örnek mobil iletişim cihazlarıdır. Dünya genelinde yediden yetmişe birçok kişi bu hızlı gelişen teknolojiye ayak uydurmaya başlamıştır. Yazılım sektöründe bu gelişmeye hızlı bir şekilde adaptasyon sağlanmıştır. Önceleri sunucu ya da dağıtık sistemler üzerinde geliştirilip kullanılan bazı yazılımlar, şuan mobil cihazlar üzerinde de kullanılabilir duruma gelmiştir. Bu yazılımlardan ilk akla gelenleri de derin öğrenme yazılımlarıdır. Şuan birçok mobil uygulama üzerinde farklı derin öğrenme mimarileri kullanılarak geliştirilmiş uygulamalar bulunmaktadır. Bunlara örnek vermek gerekirse;

- Kullanıcının kişisel tercihlerinden elde edilen verileri analiz edilerek, uygun satın alma tercihlerini tespit edilebilen yazılımlar,
- Arama robotlarındaki arama kriterlerine ilişkili reklamları kullanıcıya otomatik gösteren yazılımlar,
- Güvenlik için kullanılan yüz ve parmak izi tanıyan yazılımlar,
- Ses analizi yapılabilen yazılımlar,

Mobil ortamda kullanılan derin öğrenme tabanlı yazılımlardır.

Özellikle mobil cihazlar üzerinde derin öğrenme yazılımlarının çalıştırılabilmesi için tasarlanan mimarilerin de bu ortama uygun olması gerekmektedir. Mobil ortamda kullanılan derin öğrenme mimarilerinin en önemlilerinden biriside MobileNetV2 mimarisidir. Her ne kadar mobil ortamlar düşünülerek tasarlanmış olsada, masaüstü ortamlarda da tercih edilen bir derin öğrenme mimarisidir.

MobileNetV2 görüntüleri sınıflandırmak için yaygın olarak kullanılan CNN tabanlı bir modeldir. MobileNet mimarisinin temel avantajı, geleneksel CNN modeline göre nispeten daha az hesaplama yapmasıdır (M. Liu and Zhu 2018). Bunun sonucunda da mobil cihazlar ve daha düşük donanıma sahip bilgisayarlar üzerinde de çalışmaya uygun bir mimaridir (Khasoggi, Ermatita, and Sahmin 2019; X. Liu et al. 2019; Wang, Hu, et al. 2020).

MobileNet mimarısında kullanılan iki farklı parametre ile görüntüdeki nesnelerin ayrıntıları ve gecikme süresini kontrol edebilmek mümkündür. Eğer bu parametrelere girilen değerler düşük tutulacak

olursa Palmprint Recognition gibi minimum sayıda özellikle aynı derecede etki gösterebilir. Bunun sonucunda mimariyi basitleştirebilmek yada kompleks hale getirebilmek mümkün olmaktadır (Michele, Colin, and Santika 2019). Bu durum MobileNetV2 mimarisinin temel avantajlarından birisi olarak sayılmaktadır (Bi et al. 2020).

Derin öğrenme algoritmalarında genel olarak bu tarz avantajlar sağlayan ana etmenler, mimari yapının sahip olduğu özel konvolüsyon yapılarıdır. MobileNetV2 mimarisi ve farklı derin öğrenme mimarilerinde kullanılan bu yapılardan bazıları;

- Derinlemesine konvolüsyon
- Derinlemesine ayrılabılır konvolüsyon (Depthwise seperable convolution)
- Noktasal konvolüsyon (Pointwise convolution) yapılarıdır.

2.3.1. Derinlemesine konvolüsyon (Depthwise convolution, DC)

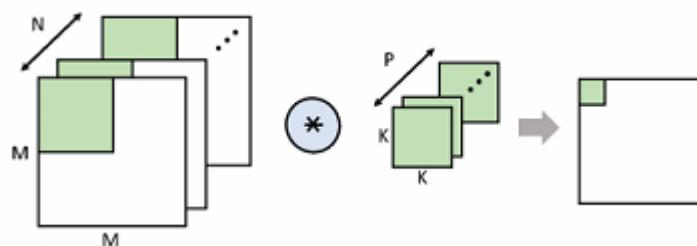
Literatürde birçok uygulamada konvolüsyon terimi yerine filtre kavramı da kullanılmaktadır. Bir sinir ağına, standart konvolüsyon katmanı uygulanırsa;

$Giriş_{genişliği} \times Giriş_{yüksekliği} \times Filtre_{genişliği} \times Filtre_{yüksekliği}$ kadar parametre içerir. Bu nedenle çok fazla parametre olduğunda sistemin öğrenememe durumu ya da öğrenmenin çok uzun sürdüğü durumlarla karşılaşılabilmektedir. Bundan kaçınmak için geliştirilen farklı konvolüsyon yöntemleri vardır. Derinlemesine konvolüsyon (Depth-

wise convolution) ve derinlemesine ayrılabilen konvolüsyon (depth-wise separable convolution) yöntemleri bunlardan sadece birkaçıdır.

Derinlemesine konvolüsyon yönteminde birden fazla kanaldan oluşan giriş görüntüsünde her bir kanala bir derinlik seviyesinde iki boyutlu bir filtre uygulanmaktadır.

Örneğin $M=8$ ve $N=3$ için 8×8 boyutundaki renkli bir görüntü, $8 \times 8 \times 3$ (giriş genişliği \times giriş yüksekliği \times giriş kanalı) boyutlarında bir giriş matrisi oluşturur. Burada giriş matrisi renkli bir görüntü olması nedeniyle $K=4$ ve $P=3$ için $4 \times 4 \times 3$ boyutlarında bir filte uygulanır. Bu durumdaki standart bir konvolüsyon işlemi Şekil 11'de gösterilmiştir.

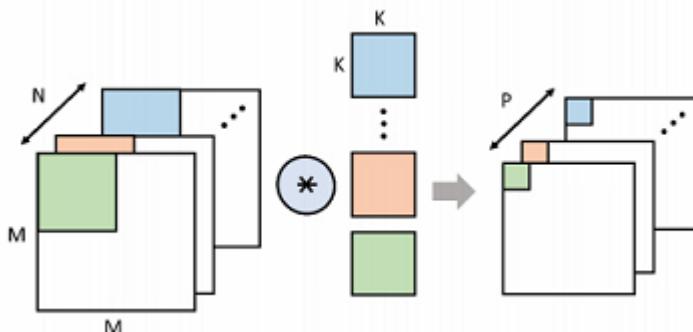


Şekil 11: Standart Konvolüsyon İşlemi (SC, (Bai, Zhao, and Huang 2018))

Buradan da görüleceği üzere konvolüsyon işlemi doğrudan uygulanmaktadır. Derin konvolüsyon yönteminde ise renkli görüntünün her bir giriş kanalına ayrı ayrı filtre uygulanarak çok kanallı bir çıkış matrisi elde edilmektedir. Bu durum Şekil 12'de detaylı olarak gösterilmiştir.

Genel olarak derin konvolüsyon yöntemini özetlersek aşağıdaki üç adımdan oluşmaktadır.

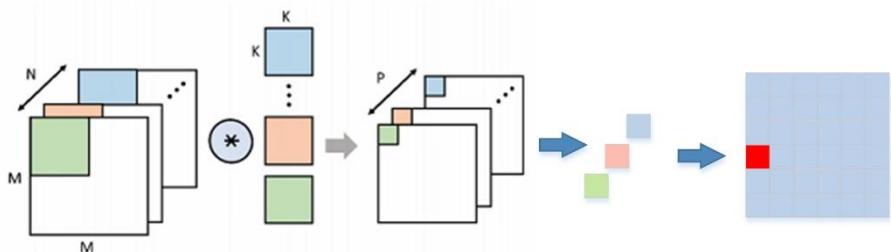
1. Girişi görüntüsünü kanallara ayırma.
2. Her bir kanala filtreleri uygulama ve her bir kanal için iki boyutlu çıkış matrisi elde etmek (giriş görüntüsündeki kanal sayısı ile kullanılan filtre arasındaki kanal sayısı eşit olmalıdır).
3. Elde edilen çıkış matrislerini toplama.



Şekil 12: Derinlemesine Konvolüsyon (Depth-wise convolution, (Bai, Zhao, and Huang 2018))

2.3.2. Derinlemesine ayrılabilir konvolüsyon (Depthwise separable convolution, DWC)

Derinlemesine konvolüsyon yöntemindeki tüm adımlar bu yöntemde de gerçekleştirilmektedir. Derinlemesine konvolüsyon yöntemine ek olarak kanallar arasında 1×1 boyutunda bir filtre işlemi daha gerçekleştirilir. Bu işlem standart konvolüsyon işlemine benzer şekilde gerçekleştirilmektedir ve Şekil 13'de gösterilmektedir. Bu adım, farklı çıkış kanalları için birçok kez tekrarlanabilir.

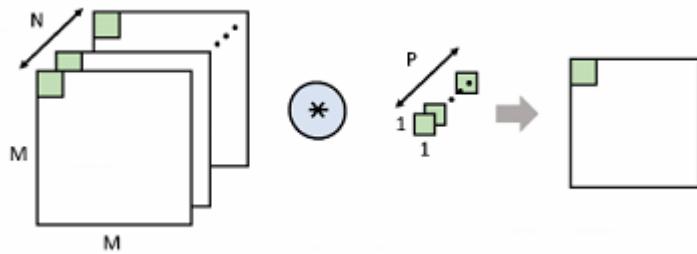


Şekil 13: Derinlemesine Ayrılabilir Konvolüsyon (Depthwise Separable Convolution)

2.3.3. Noktasal konvolüsyon (Pointwise convolution, PWC)

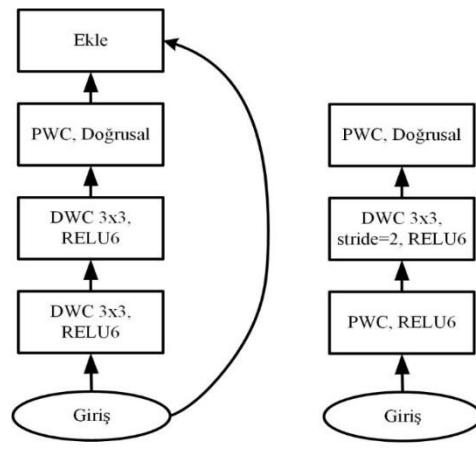
Üç veya daha fazla boyutlu verileri temsil etmek için genel olarak hacimsel gösterim kullanmaktadır. Örneğin, VoxNet (Maturana and Scherer 2015) her nesneyi $64 \times 64 \times 64$ çözünürlüğe kadar bir hacimle temsil etmektedir.

Görüntü uygulamalarında kullanılan ağ mimarilerinde de genelde bu gösterim kullanılmaktadır. Fakat bu kullanımın önemli bir dezavantajı bulunmaktadır. Hacimsel gösterim büyük miktarda bellek gerektirmektedir (Riegler, Osman Ulusoy, and Geiger 2017). Bu nedenle geliştirilen noktasal konvolüsyon yöntemi, veri matrislerinin her noktasına uygulanan bir konvolüsyon operatörü vasıtası ile gerçekleştirilmektedir. Bu durum Şekil 14'de gösterilmiştir.



Şekil 14: Noktasal Konvolüsyon (Pointwise Convolution)

MobileNetV2 mimarisi, derinlemesine ayrılabilir konvolüsyon yönteminin tercih edildiği mimarilerin başında gelmektedir. Bu mimari kendi içerisinde de ilk versiyonu ile kıyaslanırsa çıkış kanallarının daraltılarak ağırlık sayısının daha da azaltıldığı görülmektedir. Buna ek olarak derinlemesine ayrılabilir konvolüsyondan önce bir noktasal konvolüsyon katmanı daha içe aktarılarak performansı daha da arttırmıştır.



(a) Stride = 1

(b) Stride = 2

Şekil 15: Darboğaz (bottleneck)

Yapılan bu yeniliğe darboğaz (bottleneck) denir ve bu yapı Şekil 15'de gösterilmektedir. MobileNetV2 mimarisinde kullanılan katmanlar da kısaca Tablo 4'de sunulmuştur.

Tablo 4: MobileNetV2

| Giriş | Operatör | <i>t</i> | <i>c</i> | <i>n</i> | <i>s</i> |
|----------------|----------------------------|----------|----------|----------|----------|
| 224 x 224 x 3 | Konvolüsyon 2D | - | 32 | 1 | 2 |
| 112 x 112 x 32 | bottleneck | 1 | 16 | 1 | 1 |
| 112 x 112 x 16 | bottleneck | 6 | 24 | 2 | 2 |
| 56 x 56 x 24 | bottleneck | 6 | 32 | 3 | 2 |
| 28 x 28 x 32 | bottleneck | 6 | 64 | 4 | 2 |
| 14 x 14 x 64 | bottleneck | 6 | 96 | 3 | 1 |
| 14 x 14 x 96 | bottleneck | 6 | 160 | 3 | 2 |
| 7 x 7 x 160 | bottleneck | 6 | 320 | 1 | 1 |
| 7 x 7 x 320 | Konvolüsyon 1 2D 1 x 1 | - | 1280 | 1 | 1 |
| 7 x 7 x 1280 | Havuzlama (Ortalama) 7 x 7 | - | - | 1 | - |
| 1 x 1 x 1280 | Konvolüsyon 2D 1 x 1 | - | k | - | - |

Tablo 4'de kullanılan *t* giriş boyutlarına uygulanan genişleme değerini, *n* tekrar sayısını, *c* çıkış kanal sayısını ve *s* ise atlama (stride) değerini ifade etmektedir.

2.3.4. MobileNetV2 hesaplama maliyeti

Derin öğrenme mimarilerinin başarısı kadar işlem maliyetide önemlidir. Özellikle mobil ortamlarda ya da düşük donanım seviyesine sahip bilgisayarlarda çalıştırılacak derin öğrenme uygulamaları

doğrudan donanım bağımlıdır. Bu nedenle literatürde MobileNetV2 mimarisinin hesaplama maliyetine yönelik yapılan önemli çalışmalarda bulunmaktadır.

Howard ve Sandler'e göre bir L_i giriş matrisine $h_i * w_i * d_i$ boyutlarında K çekirdek konvolüsyon matrisi uygulanırsa ve matrisin değerleri $R \in k \times k \times d^i \times d^j$ ise olacak L_j çıkış matrisinin boyutları $h_i * w_i * d_j$ 'dır.

Hesaplama maliyeti ise $h_i * w_i * d_i * d_j * k * k$ 'dır. Standart konvolüsyonlu katmanların yerini derinlemesine ayrılabilir konvolüsyonlar alındığında standart konvolüsyon katmanlar kadar iyi sonuç vermektedir. Sadece $h_i * w_i * d_i (k^2 + d_j)$ kadar maliyeti vardır. Eğer burada derinlik için $k=1$ olursa, 1×1 matris demek olacağı için neredeyse standart konvolüsyonların toplamı kadar bir hesaplama maliyeti olmaktadır. k^2 ifadesi standart konvolüsyon katman ile karşılaşıldığında derinlemesine ayrılabilir konvolüsyon katmanın hesaplama maliyetini etkili bir şekilde azaltır.

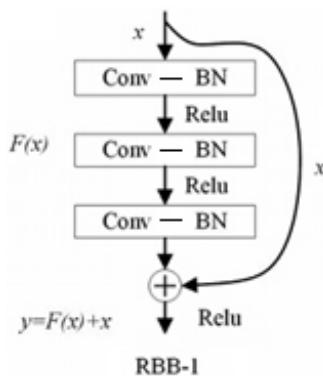
Örneğin $k=3$ olarak seçilirse ki bu 3×3 bir matris dolaştırmak demektir. Böyle bir durumda da hesaplama maliyeti yaklaşık olarak $1/8-1/9$ oranında azalacaktır (Howard et al. 2017; Sandler et al. 2018).

2.4. RESNET50 Mimarisi

Günümüzde yaygın olarak kullanılan derin öğrenme mimarilerinde biriside RESNET mimarisidir. RESNET 50 mimarisinde genel olarak toplam 177 katman bulunmaktadır (Matlab 2021). Bu mimarinin blok

ve katman yapısına göre RESNET18/50/101 ve 152 (Narin, Kaya, and Pamuk 2021; Rajpal et al. 2021) olmak üzere farklı yapılarla bulunmaktadır. Özellikle bu mimari yapıda eğitim aşamasında, klasik konvolüsyon yöntemlerine göre çok daha yüksek verim elde edilmektedir (T. Liu et al. 2019). Mimari içerisinde kullanılan yapılardan biriside Residual Building Block (RBB) yapısıdır. RBB içerisinde katmanları atlayan kısayol bağlantıları (shortcut connections) bulunmaktadır (T. Liu et al. 2019). Bu bağlantılar sayesinde esnek hesaplama yöntemlerinde genel bir sorun olan yerel optimum noktasına takılma sorununda çözülmlesi amaçlanmıştır. Bu nedenle RBB’ler kısayol bağlantıları kullanarak konvolüsyonel katmanların blok olarak atlama fikrini temel almaktadır.

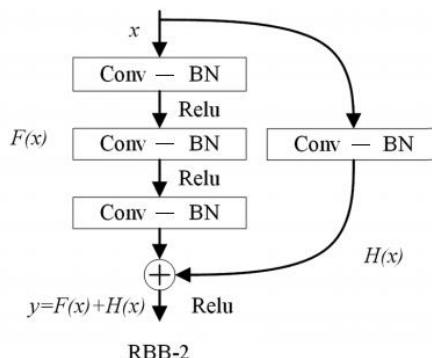
RBB’ler konvolüsyon (Conv), toplu normalleştirme (Batch Normalization-BN), aktivasyon (ReLU) katmanlarına ek olarak bir kısayoldan oluşur. Genel olarak RBB’ler Şekil 16 ve 17’de gösterildiği gibi RBB-1 ve RBB-2 olmak üzere iki farklı yapısı bulunmaktadır. RBB1 yapısında kullanılan kısayollar Şekil 16’da x ile gösterilmiştir. Bu yapıda konvolüsyon katmanı için doğrusal olmayan bir fonksiyon kullanılmış ve F ile gösterilmiştir. y ile ifade edilen RBB-1’in çıkışı ise Denklem (4)’de sunulmuştur.



Şekil 16: RBB-1 Yapısı

$$y = F(x) + x \quad (4)$$

RBB-2 yapısında kullanılan kısayollar ve genel yapısı ise Şekil 17' de gösterilmiştir. Bu yapıda da konvolüsyon katmanı için doğrusal olmayan bir fonksiyon kullanılmıştır.

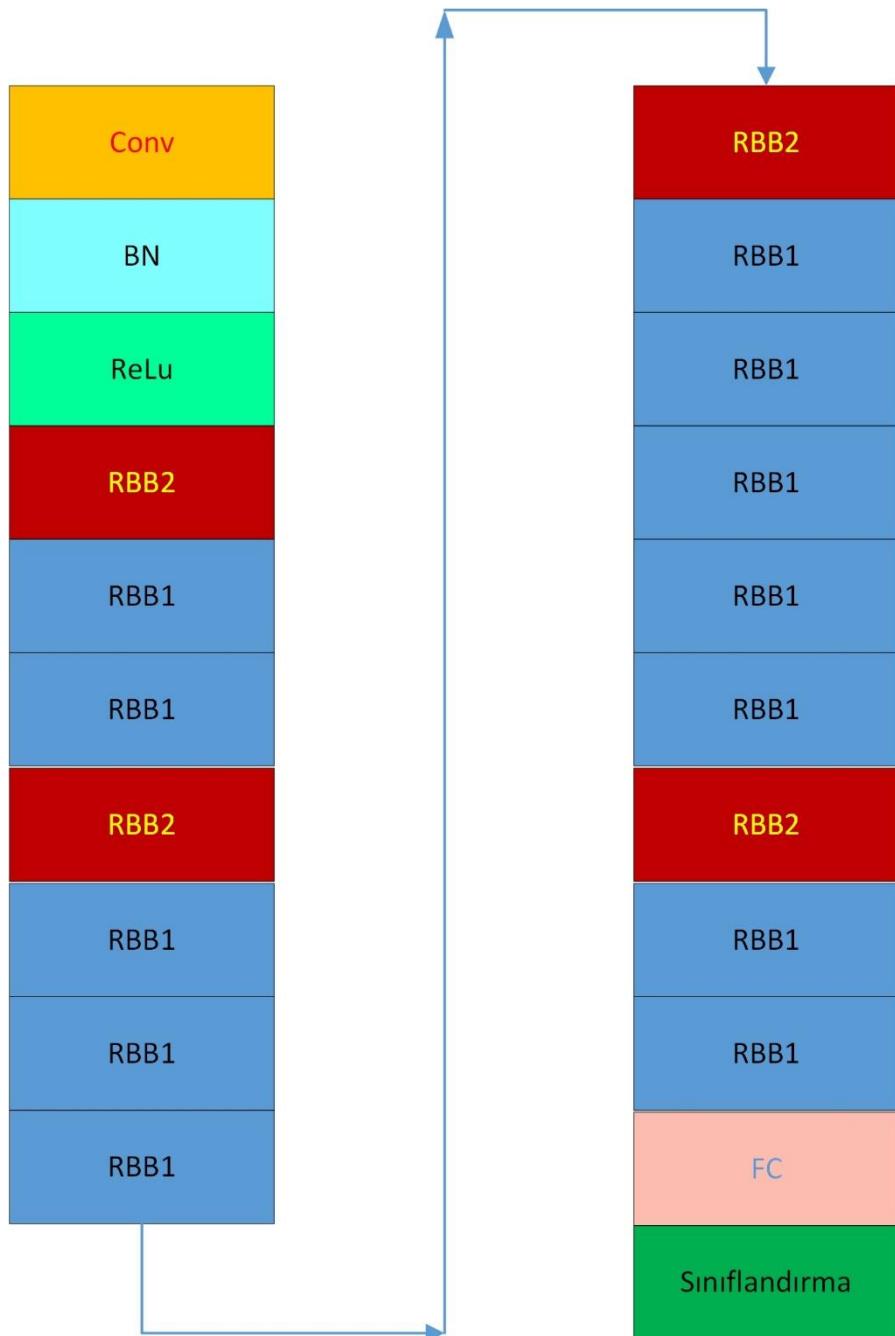


Şekil 17: RBB2 Yapısı

$$y = F(x) + H(x) \quad (5)$$

Şekil 17'den anlaşılacağı üzere RBB-2 yapısında bir adet Conv-BN katmanı daha eklenmiştir. RBB-2'nin yapısında bulunan kısayol çizgisi Şekil 17'de gösterildiği üzere H ile ifade edilmektedir. RBB-2'nin genel çıkış ise y ile gösterilirse bu durum Denklem (5) ile formülüze edilir.

RBB 1 ve RBB 2 ile oluşturulan RESNET 50 mimarisinin genel yapısı Şekil 18'de detaylı olarak gösterilmiştir.



Şekil 18: RESNET 50 Mimarisi

BÖLÜM 3

MATLAB UYGULAMALARI

Bu bölümde, derin öğrenme mimarileri ile ilgili örnek uygulamalar sunulmuştur. Derin öğrenme mimarileri her ne kadar sınıflandırma işlemleri için kullanılsada regresyon için kullanılan derin öğrenme mimari modelleri de literatürde bulunmaktadır (Zhao et al. 2017).

3.1. SQUEEZE.NET

Uygulama geliştirilebilmesi için ilk olarak elimizde derin öğrenme mimarilerinin uygulanabileceği bir veri setinin bulunması gerekmektedir. Bu nedenle MATLAB'in kendi içerisinde bulunan örnek veri setlerinden birisi tercih edilmiştir. Bu veri setinin MATLAB ortamına yüklenmesi komut satırından veya ilgili menülerden yapılabilmektedir.

Bu amaç doğrultusunda uygulamalarda MerchData veri seti kullanılacaktır. Bu veri seti içerisinde sınıflandırmaya uygun olarak kullanılabilecek görüntüler bulunmaktadır.

Uygulmada ilk olarak verilerin komut satırı ile yüklenmesi gösterilecektir. Bunun içinde aşağıdaki adımlar sırası ile takip edilmelidir.

Adım 1: Sınıflandırma işleminde kullanılacak görüntülerin MATLAB ortamında workspace'e yüklenmesi için aşağıdaki kod satırı çalıştırılır.

```
unzip('MerchData');
```

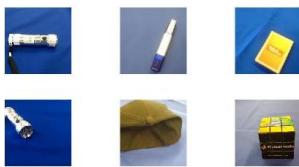
Sonrasında klasör isimleri ve etiketleri aşağıda gösterildiği üzere bir değişkene atanır.

```
imds =
imageDatastore('../MerchData','IncludeSubfolders',true,'LabelSource','foldernames');
```

Bir sonraki adımda ise yapay zekâ algoritmaları için çok önemli olan eğitim ve test verilerinin ayırma işlemi gerçekleştirilmiştir. Çalışmada kullanılan verilerin %80'i eğitim, geri kalanı ise test verisi olarak iki sınıfa ayrılmıştır.

```
[imdsTrain,imdsTest] =
splitEachLabel(imds,0.8,'randomized');
```

Kullanılan veri setindeki 75 adet görüntünün 60 tanesi eğitim, 15 tanesi test olarak iki ayrı sınıfa rastgele bir şekilde ayrılmıştır. Literatürde bundan farklı olarak eğitim ve test verilerini ayırmak için kullanılan algoritmalar da bulunmaktadır. Şekil 19'da eğitim setinden alınan 6 adet görüntü ve bu görüntülerini almak için gerekli MATLAB komutları Şekil 20'de gösterilmiştir.



Şekil 19: Örnek Eğitim Görüntüleri

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
fx >> numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,9);
figure
for i = 1:9
    subplot(3,3,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```

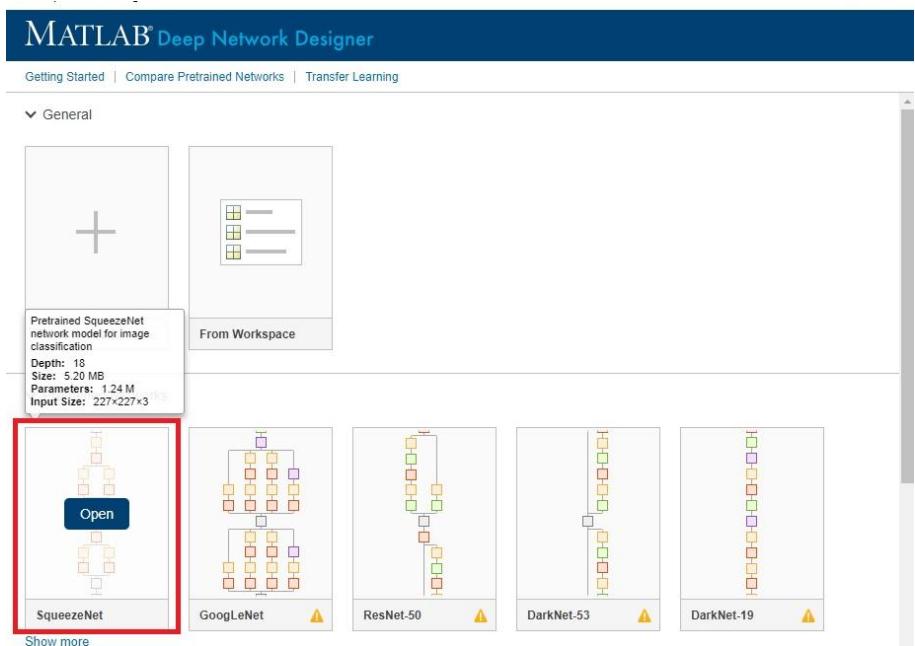
Şekil 20: Komutlar

Buraya kadar yapılan işlemlerde veri seti komut satırı ile MATLAB çalışma ortamına yüklenmiştir.

Adım 2: Çalışmada kullanılacak derin öğrenme mimarisi seçmek için deepNetworkDesigner komutu aşağıdaki şekilde çalıştırılmalıdır.

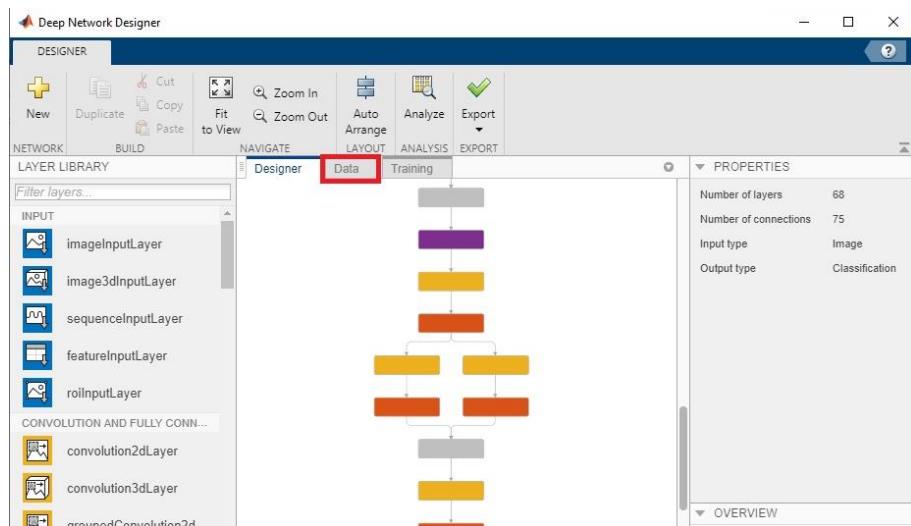
```
deepNetworkDesigner
```

Bu komut sonrasında Şekil 21'de gösterilen menüdeki mimarilerden bir tanesi seçilir. Çalışmada tercih edilen mimari ise yine Şekil 21'de kırmızı kutu içerisinde gösterilen SqueezeNet'dir.



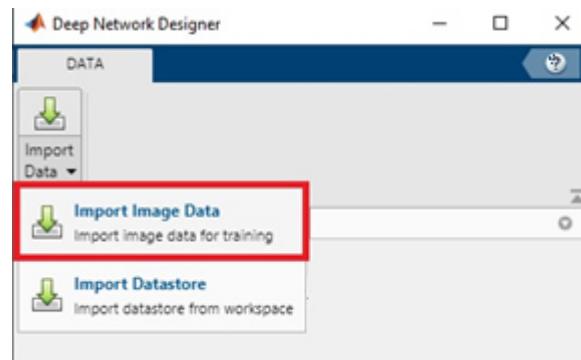
Şekil 21: DeepNetworkDesigner Ekran Görüntüsü

Mimari seçme işlemi gerçekleştirildikten sonra Şekil 22'de gösterildiği üzere **Deep Network Designer** menüsü karşımıza gelmektedir.



Şekil 22: Mimari Seçimi

Bu menüde bulunan ve kırmızı kutu içerisinde gösterilen Data sekmesi seçilerek mimaride kullanılacak veriler girdi olarak seçilecektir.

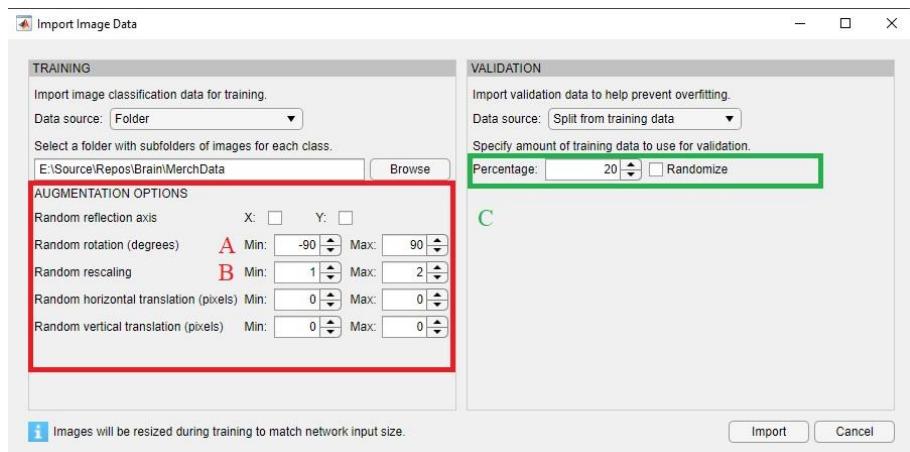


Şekil 23: Import Image Data

Bu işlem için ise Şekil 23'de gösterildiği üzere kırmızı kutu içerisindeki **Import Image Data** seçeneği ile veri seti çalışmaya dahil edilmektedir. Bu işlem için kod satırı ile MATLAB çalışma ortamına yüklenen veriler

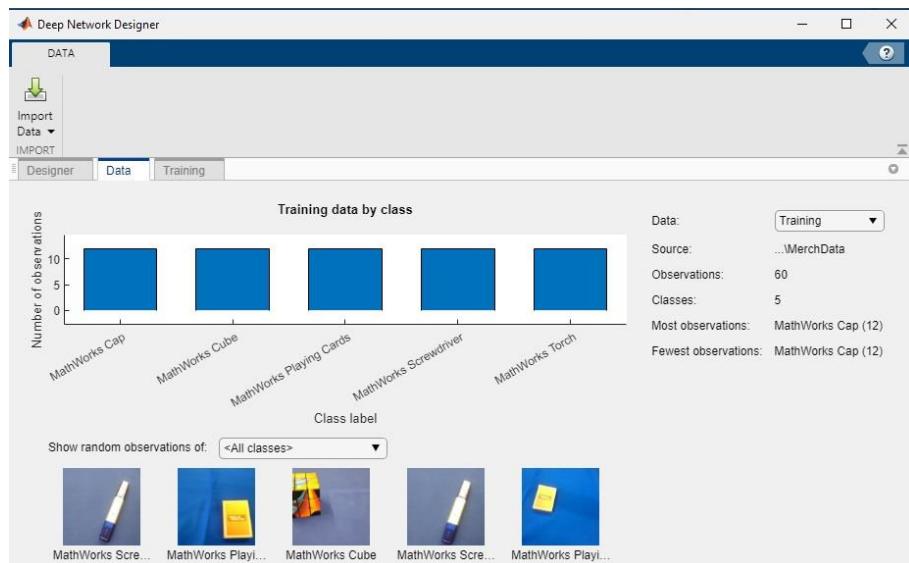
seçilebileceği gibi farklı klasörler içerisinde de veriler seçilebilmektedir.

Çalışmaya dâhil edilen veri miktarı bu çalışmada da olduğu gibi az olduğunda veri miktarının arttırılması gerekebilmektedir. Bu işlem içinde dataaugment yöntemi kullanılmaktadır. Çalışmada Şekil 24’de gösterildiği üzere dataaugment işlemi gerçekleştirılmıştır. Şekil 24’de A ile gösterilen işlem ile görüntüler -90 derece ile 90 derece arasında döndürülmüş ve B seçeneğinde gösterilen ölçekleme işlemi ile de büyütülp küçültülmüştür. Adım 1’de komut satırı ile gösterilen veri setini ayırma işlemi, [Import Image Data](#) menüsünde C ile gösterilen yeşil kutucuk içerisindeki ComboBox ile de gerçekleştirilebilmektedir.



Şekil 24: Dataaugment

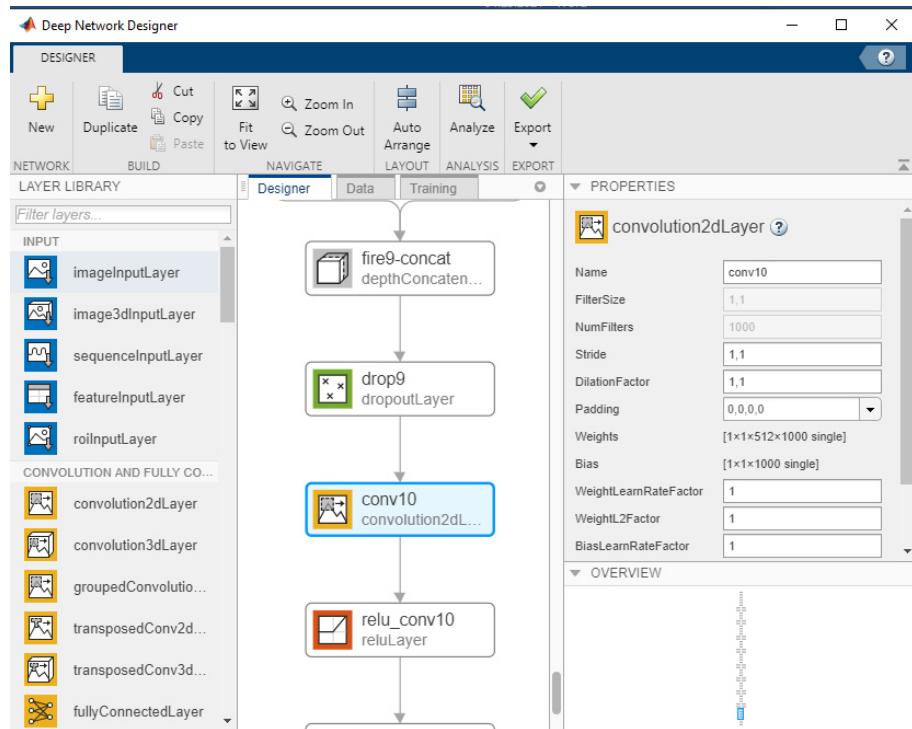
Şekil 24’de kırmızı ve yeşil kutularda gösterilen alanlar girilen değerler ile değiştirildikten sonra veriler import edilmektedir. Bu aşamadan sonra eğitim verileri ve sınıfları Şekil 25’de gösterilmiştir.



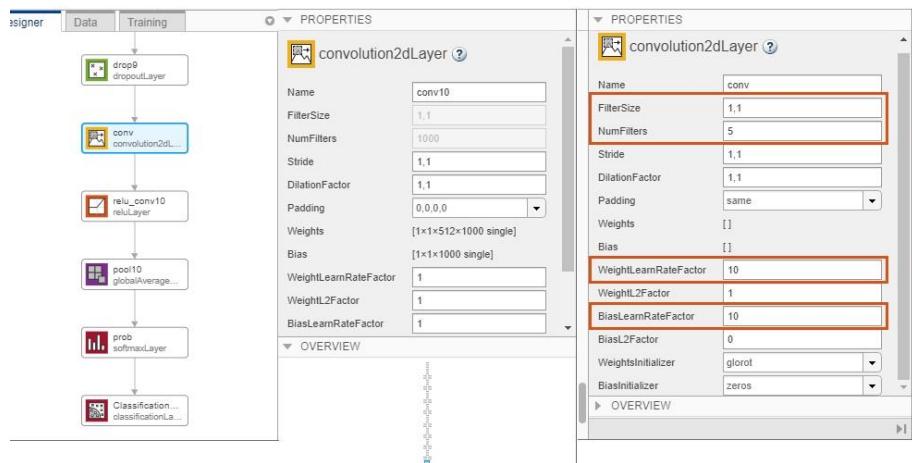
Şekil 25: Import Edilen Eğitim Verileri

Şekil 25’de gösterilen data menüsünden Şekil 26’da gösterilen Designer menüsüne tekrar geri geçildiğinde, ilk olarak mavi ile işaretlenen conv10 katmanı kaldırılarak yerine yeni bir Konvolizasyon katmanı ([convolution2dLayer](#)) eklenmektedir. Şekil 26 ve 27(a)’da gösterilen varsayılan ayarlar önceden eğitilmiş (PreTrained) olan mimariye ait değerlerdir.

Bu nedenle varsayılan ayarlar çalışmaya özgü olarak Şekil 27(b)’de gösterilen kırmızı kutucuk içerisindeki parametreler ile değiştirilmelidir. Kırmızı kutucuk içerisindeki [NumFilters](#) sayısı, bu çalışmada kullanılan veri setinin sınıf sayısını temsil etmektedir.



Şekil 26: Mimariye Ait Varsayılan Ayarlar

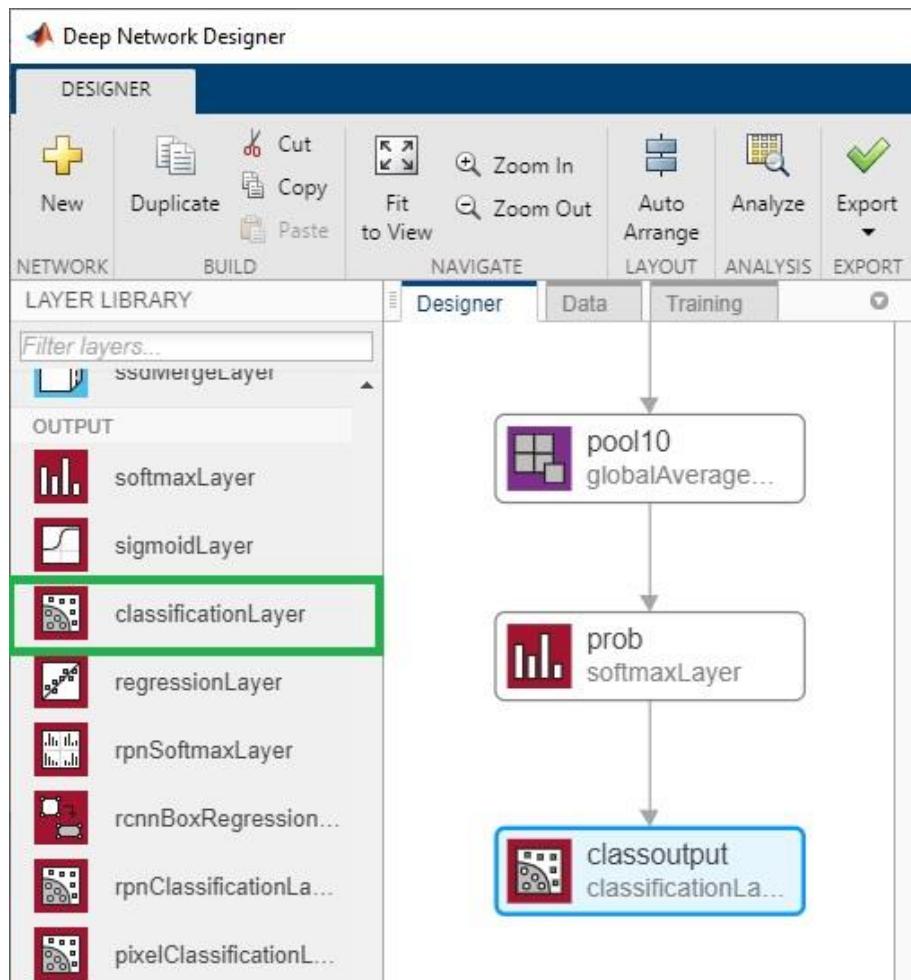


(a)

(b)

Şekil 27: Konvolüsyon Katmanı Değişikliği

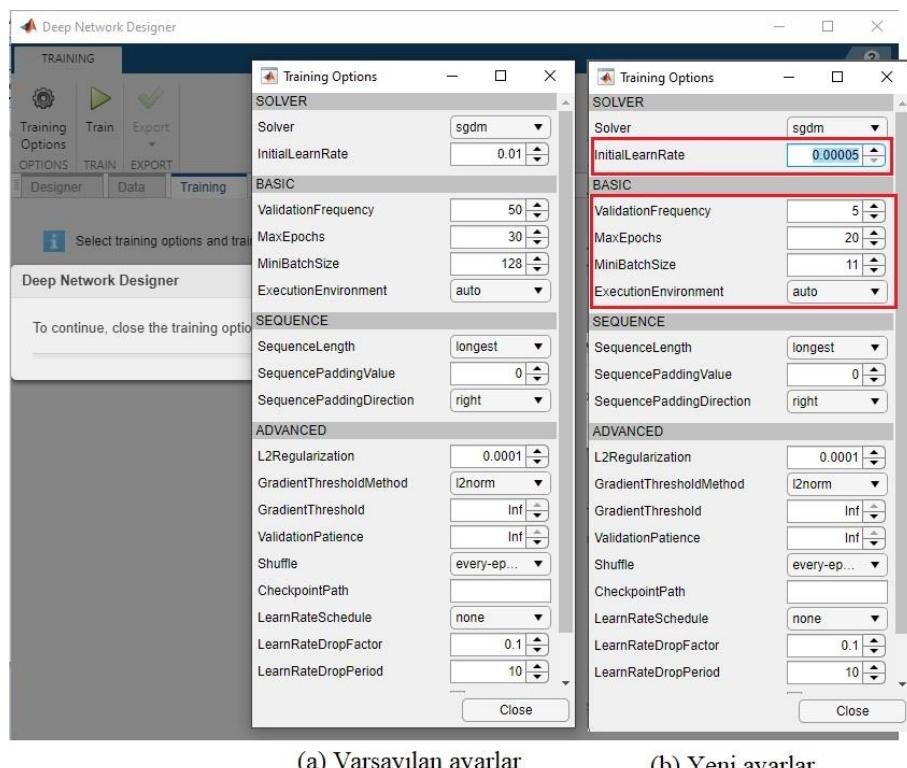
Şekil 28'de varsayılan olarak gelen Classification katmanı kaldırılarak yerine [Layer Library](#) bölümünden yeşil bir kutucuk içerisinde gösterilen Classification katmanı eklenmelidir.



Şekil 28: Layer Library Classificationlayer

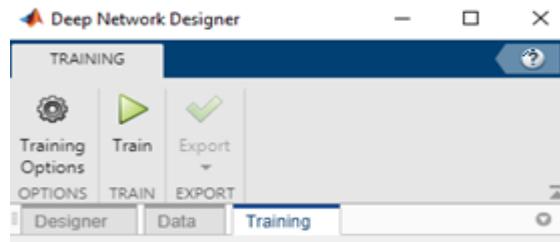
Şekil 27 ve 28'de gösterilen iki ana değişiklik ile birlikte varsayılan mimari bu çalışmada kullanılan veri setine uyarlanmıştır. Bundan

sonraki aşamada eğitim ayarlamaları için Deep Designer Network menüsünde bulunan Training sekmesine geçilerek Training options menüsü üzerinden Training options paneline geçilmektedir. Bu panel ilk açılış esnasında default olarak Şekil 29(a)'da ki ayarlar ile gelmektedir. Bu default ayarlar çalışmaya uygun olarak Şekil 29(b)'de kırmızı kutucuk ile gösterilen değerler ile değiştirilmektedir. Transfer katmanında öğrenmeyi yavaşlatmak için öğrenme oranı küçük bir değer olarak seçilmiştir. ValidationFrequency değeri ise her epoch işleminde hesaplanan test verisindeki doğruluğu göstermektedir.



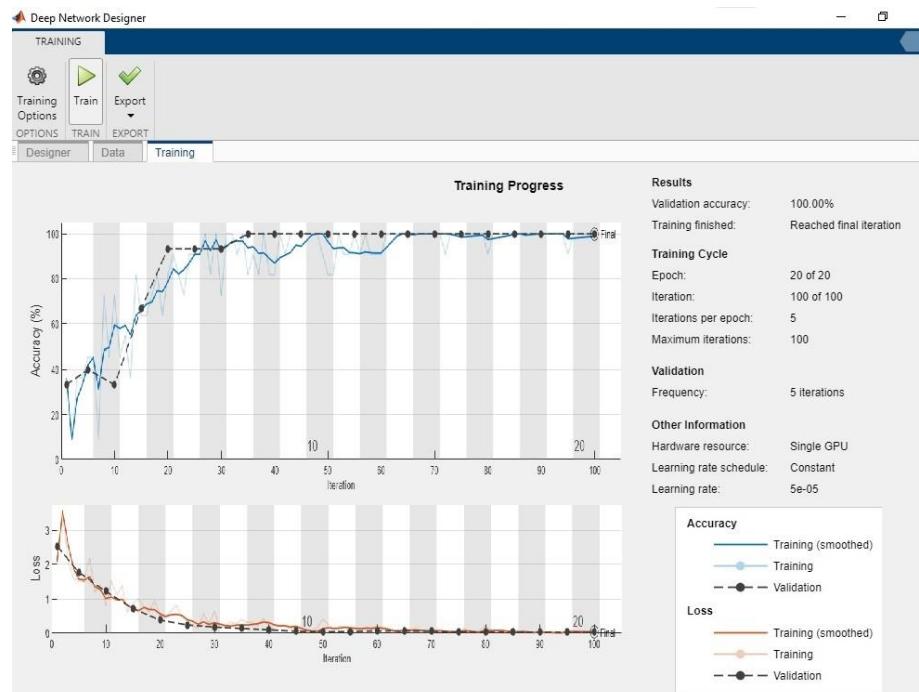
Şekil 29: Eğitim Ayarları

Eğitim işleminin son aşamasında ise Şekil 30'da gösterilen DeepNetwork Designer menüsünde bulunan Train butonuna basılarak sistemin eğitilmesine başlanmaktadır.



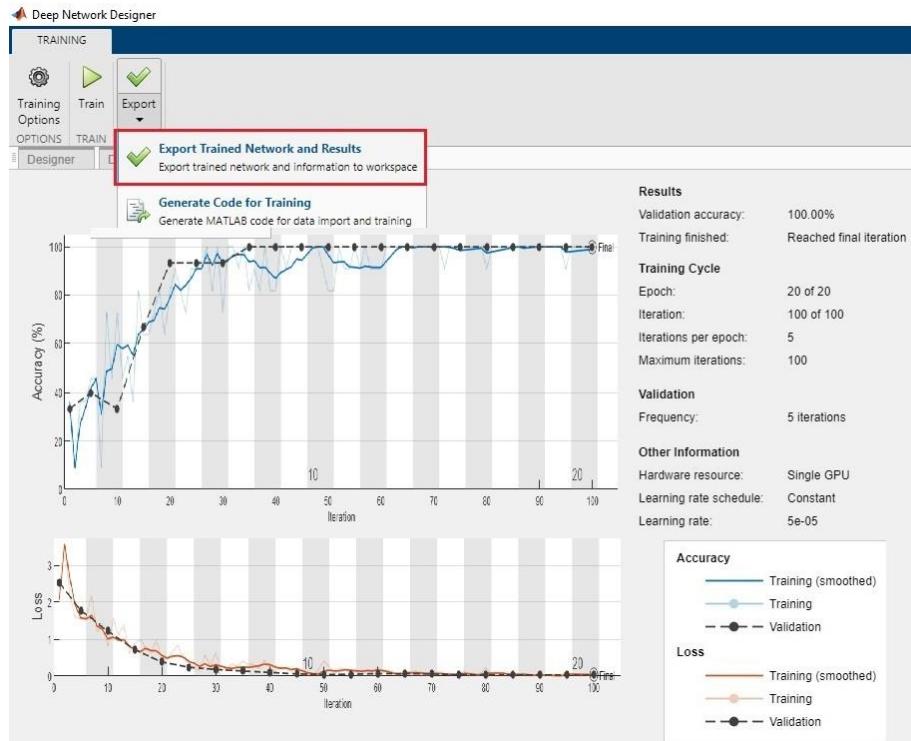
Şekil 30: Train Butonu

Eğitim işlemi sonucunda elde edilen doğruluk oranı ve eğitim süreci detaylı olarak Şekil 31'de sunulmuştur.



Şekil 31: Eğitim Sonucu

Adım 3: Tasarımı yapılan ve eğitimi tamamlanan mimari Şekil 32’de gösterildiği üzere farklı test işlemlerinde de kullanılmak üzere kaydedilmiştir.



Şekil 32: Eğitilmiş Ağın Kayıt İşlemi

Eğitimi tamamlanan ağıın kodları istenilirse [Generate Code for Training](#) butonu ile otomatik olarak üretilmektektir. Bu işlemler sonucunda workspace'in ve Command Window'un son hali sırasıyla Şekil 33(a) ve Şekil 33(b)'de gösterilmiştir.

| Workspace | |
|-------------------|--------------------|
| Name | Value |
| imds | 1x1 ImageDatastore |
| imdsTest | 1x1 ImageDatastore |
| imdsTrain | 1x1 ImageDatastore |
| trainedNetwork_1 | 1x1 DAGNetwork |
| trainInfoStruct_1 | 1x1 struct |

(a) Workspace

```
Command Window
New to MATLAB? See resources for Getting Started.
>> unzip("MerchData.zip");
>> imds = imageDatastore('MerchData','IncludeSubfolders',true,'LabelSource','foldernames');
>> [imdsTrain,imdsTest] = splitEachLabel(imds,0.8,'randomized');
fx >>
```

(b) Command Window

Şekil 33: İşlem Sonucu

3.1.1. Sistemin komut satırı ile test edilmesi

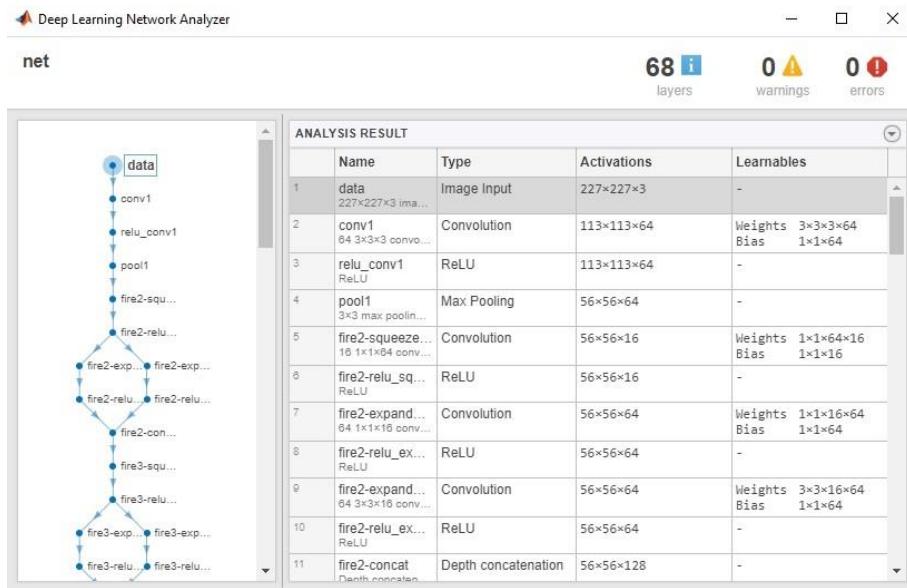
Tasarımı yapılan mimarinin komut satırı ile test edilmesi ise aşağıdaki şekilde gerçekleştirilmektedir. Bunun için çalışmada kullanılan veriler rastgele aşağıda gösterildiği üzere tekrar %80'e %20 olarak ikiye ayrılmıştır.

```
imds =
imageDatastore('../MerchData','IncludeSubfolders',true,
'LabelSource','foldernames');
[imdsTrain,imdsTest] =
splitEachLabel(imds,0.8,'randomized');
```

Test verileri ayarlandıktan sonra aşağıda ki adımlar takip edilerek eğitilen mimari üzerinden test işlemi gerçekleştirilmiştir.

```
net=trainedNetwork_1;
analyzeNetwork(net);
```

Buna ek olarak analyzeNetwork komutu ile oluşturulan mimarinin genel yapısı da Şekil 34'de gösterilmektedir.



Şekil 34: Mimari Kontrol Sonucu

Aşağıdaki komut satırları yardımcı ile de eğitilen ve test edilen sistemin doğruluğu hesaplanabilmektedir.

```
% % sinir ağı test işlemi
predLabels = classify(net,imdsTest);
testLabels = imdsTest.Labels;
% % sinir ağıının doğruluğunun hesaplanması
accuracy = sum(predLabels ==
testLabels)/numel(testLabels);
fprintf('Accuracy is %8.2f%%\n',accuracy*100)
```

3.1.2. Ön eğitilmiş ağdan öznitelik elde etme

Çalışmada oluşturulan mimari kullanın veriler ile eğitilmiş ve test verileri ile de doğruluğu ölçülmüştür. Kullanılan sınıflandırma katmanı doğrudan mimariye ait olan sınıflandırma katmanıdır. Eğer çalışmada

farklı sınıflandırma algoritmaları test edilmek istenilirse, mimariden elde edilen öznitelikler kullanılarak bu işlem gerçekleştirilmektedir.

Bu işlem için tasarlanan mimariden öznitelik elde etmek işlemi aşağıdaki komut satırları ile gerçekleştirilmektedir.

```
load squazeNet;

imds =
 imageDatastore('..../MerchData','IncludeSubfolders',true,'
LabelSource',
 'foldernames');

[imdsTrain,imdsTest] =
 splitEachLabel(imds,0.8,'randomized');

net=trainedNetwork_1;
analyzeNetwork(net);

inputSize = net.Layers(1).InputSize;
augimdsTrain =
 augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest =
 augmentedImageDatastore(inputSize(1:2),imdsTest);
```

```
layer = 'pool10';
featuresTrain =
 activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest =
 activations(net,augimdsTest,layer,'OutputAs','rows');
```

```
whos featuresTrain

Ytrain = imdsTrain.Labels;
YTest = imdsTest.Labels;

%Öznitelikler ve bunlara bağlı çıkış değerleri
%mimaridenden farklı olarak SVM (fitcecoc) kullanılarak
sınıflandırma işleminin gerçekleştirilmesi

classifier = fitcecoc(featuresTrain,YTrain);
```

```

YPred = predict(classifier,featuresTest);

%Doğruluk oranının elde edilmesi
accuracy = mean(YPred == YTest)
sonuc=[YPred YTest]

```

Oluşturulan mimarideki katmanlarda bulunan değerleri, Şekil 32'de gösterilen butona ile kaydedip sonrasında da alabilmek mümkündür. Çalışmada kullanılan ve Şekil 34'de sunulan mimarinin son katmanlarından olan Pool10 katmanından alınan veriler SVM işlemine tabi tutularak sınıflandırma işlemi gerçekleştirılmıştır. İstenilirse oluşturulan mimarinin farklı katmanlarından da veriler alınarak daha farklı işlemlere tabi tutulabilmektedir.

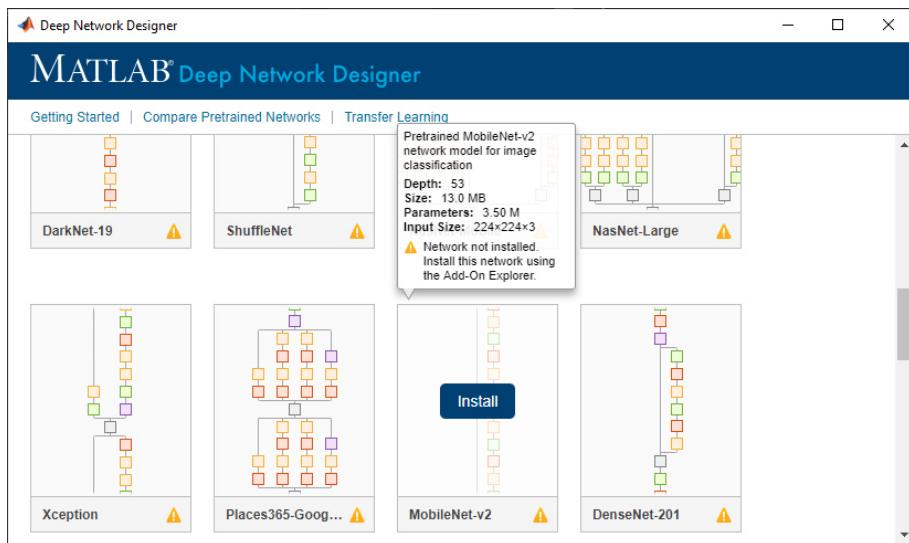
3.2. MOBILENET V2

MobileNetV2'nin genel mimarisi Bölüm 2, Tablo 4'de detaylı olarak sunulmuştur. Bu mimari MATLAB 2020b'de kurulu gelmemektedir. Bu nedenle MobileNetV2 mimarisini MATLAB ortamına kurmak için aşağıdaki adımlar sırası ile takip edilmelidir.

Adım 1: Komut satırına deepNetworkDesigner komutu yazılarak, ilgili menü ekrana çağrılmalıdır.

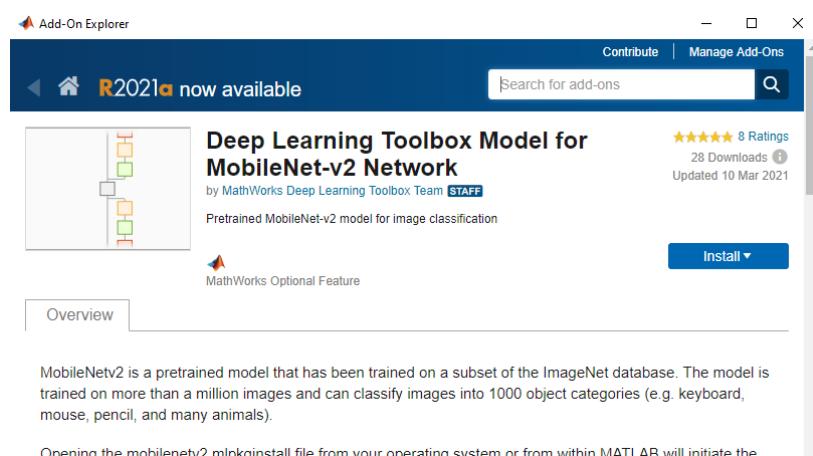
```
deepNetworkDesigner
```

Adım 2: “deepNetworkDesigner” komutu yazıldıktan sonra ekrana gelen ve Şekil 35'de gösterilen menü incelendiğinde de MobileNetV2'nin MATLAB ortamında yüklü olmadığı görülmektedir. Bu nedenle Şekil 36'da gösterildiği üzere MobileNetV2'nin üzerindeki Install butonuna basılmalıdır.



Şekil 35: Kurulum Ekranı 1

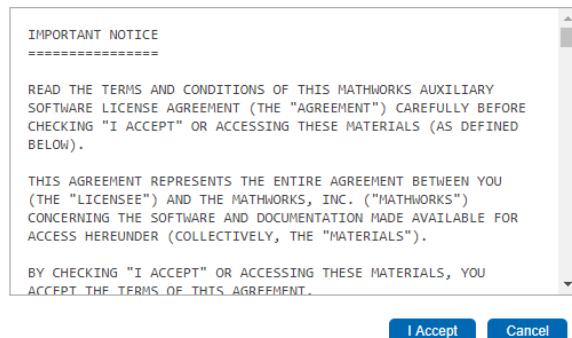
Kurulum ekranında ilgili butona tıklandıktan sonra Şekil 36'da ki menü ekrana gelmektedir. Bu menüden de Install butonuna tıklanarak kurulum işlemine başlanmaktadır.



Şekil 36: Kurulum Ekranı 2

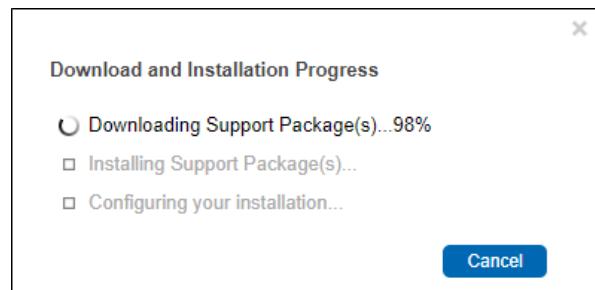
Kurulum işlemine başlanıldığında Şekil 37'deki lisans sözleşmesi karşımıza gelmektedir. Bu lisans sözleşmesi kabul edilerek bir sonraki adıma geçilmektedir.

MathWorks Auxiliary Software License Agreement



Şekil 37: Kurulum Ekranı 3

Şekil 38'de gösterilen kurulum ekranı sonlandığında mimarının MATLAB ortamına yüklenmesi tamamlanmaktadır.

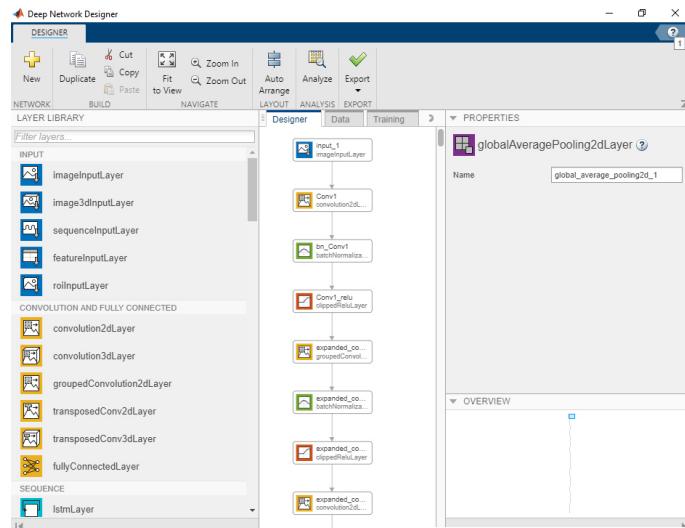


Şekil 38: Kurulum Ekranı 4

Yüklenen mimariyi çağrırmak için ise aşağıda gösterilen komut satırı kullanılmaktadır.

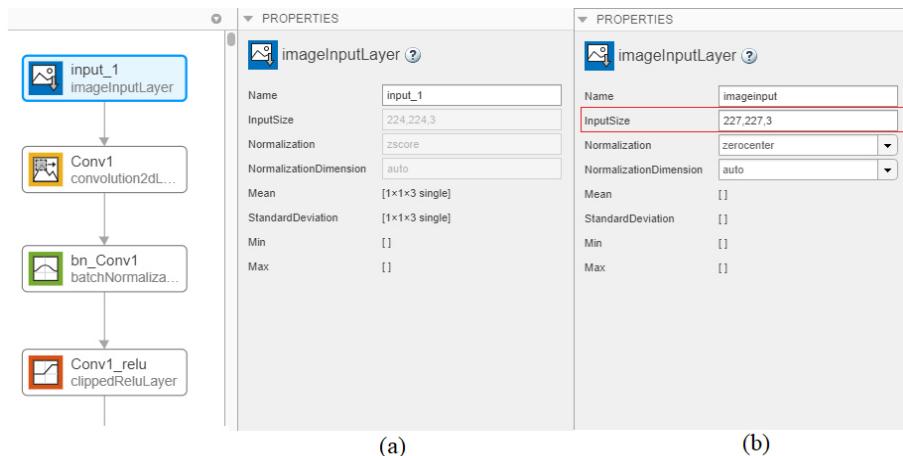
```
deepNetworkDesigner(mobilenetv2)
```

Bu komut satırından sonra karşımıza gelen tasarım ekranı Şekil 39'da gösterilmektedir.



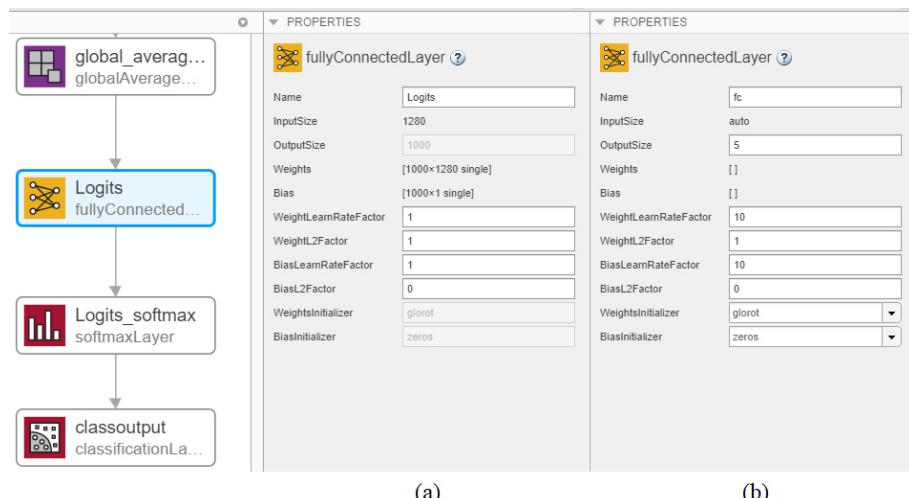
Şekil 39: MobileNetV2 Tasarım Ekranı

Şekil 40'da gösterilen input_1 katmanı silinerek Layer Library menüsünden yeni bir imageInputLayer eklenmektedir. Eklenen bu katman çalışmada kullanılacak verilere göre yeniden düzenlenmelidir. MobileNetV2 uygulamasında bir önceki çalışmada kullanılan MerchData veri seti kullanılacaktır. Bu nedenle imageInputLayer katmanı bu veri setine göre Şekil 40(b)'de gösterildiği gibi yeniden düzenlenmiştir.



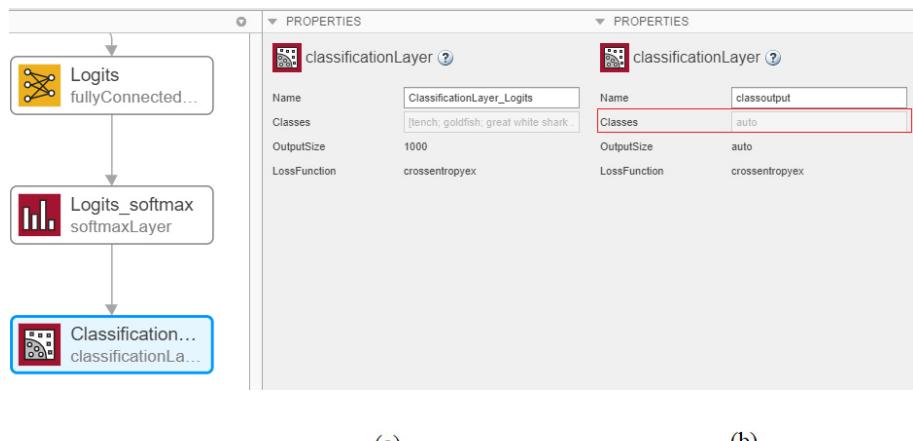
Şekil 40: ImageInputLayer ‘in Düzenlenmesi

Bir sonraki adımda FullyConnectedLayer katmanındaki Şekil 41 (a)’da default olarak gelen parametreler Şekil 41(b)’de gösterilen değerler ile değiştirilmelidir.



Şekil 41: Fullyconnectedlayer Katmanı Düzenlenmesi

Son adımda ise Classification Layer silinerek Layer Library menüsünden yeni bir Classification Layer eklenmekte ve kullanılacak verilere göre Şekil 42(b)' de gösterildiği üzere revize edilmiştir.

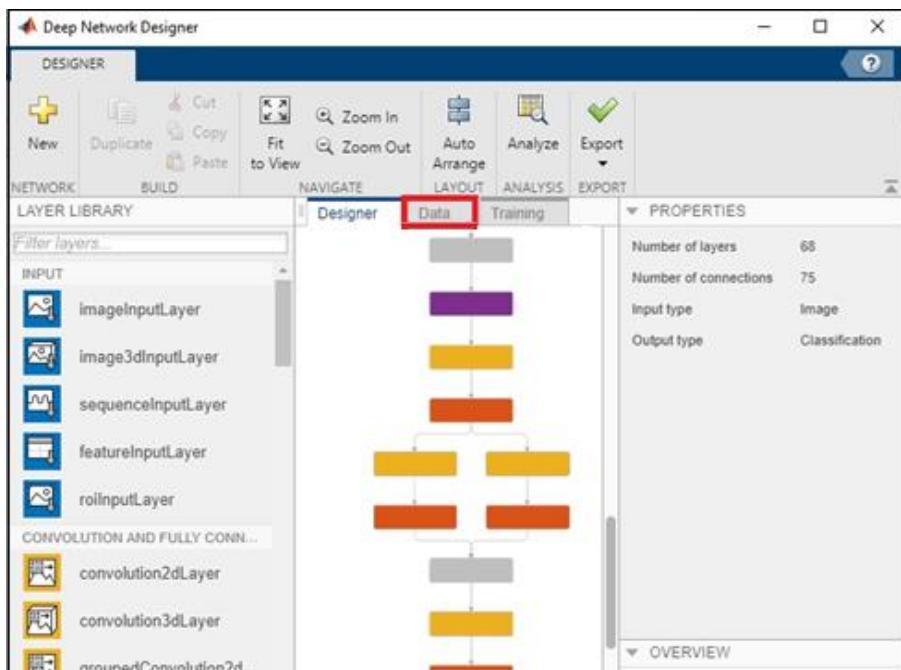


Şekil 42: Classificationlayer'in Yeniden Düzenlenmesi

Mimarının genel olarak yapısı düzenlenerek sonra workspace'a MerchData verileri aşağıdaki komutlar ile eklenmiştir.

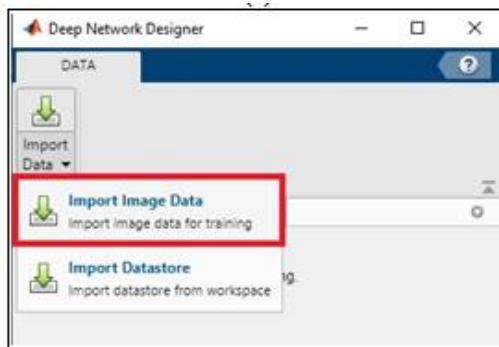
```
unzip('MerchData');
imds =
imageDatastore('../MerchData', 'IncludeSubfolders', true, ...
    'LabelSource',
    'foldernames');
```

Workspace'e eklenen bu veriler Şekil 43'de gösterilen data sekmesinden seçilerek Şekil 44'deki import image data seçeneği ile çalışmaya dahil edilmiştir.



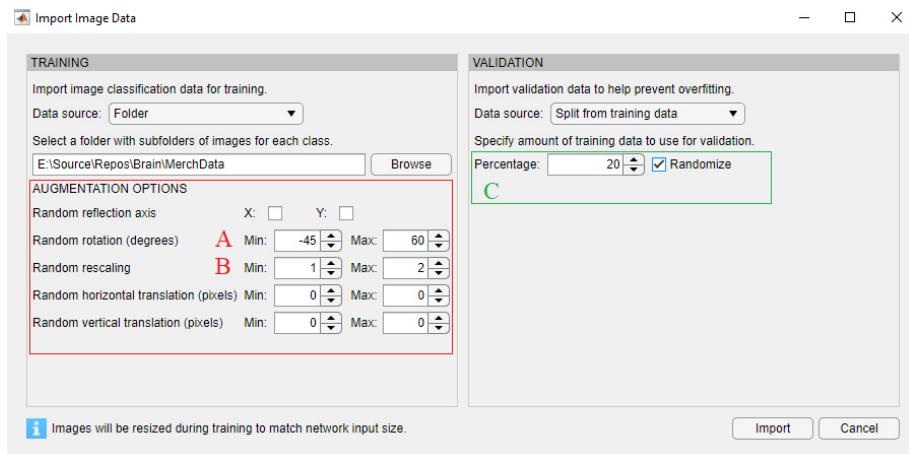
Sekil 43: Veri Seti Ekleme

Çalışmaya dahil edilen veri miktarı az olduğu için veri miktarının arttırılması gerekmektedir. Bu işlem içinde dataaugment yöntemi kullanılmaktadır.

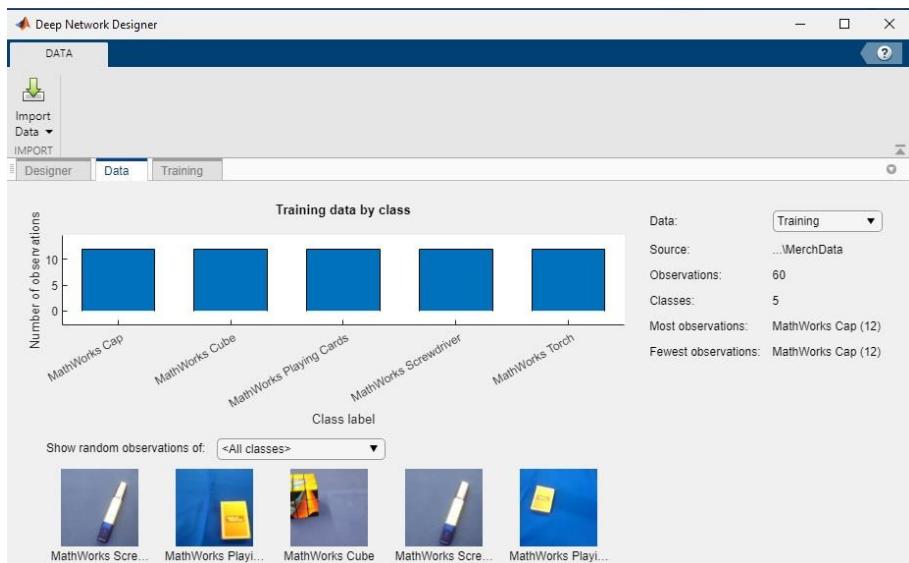


Şekil 44: Veri Seti Ekleme

Çalışmada Şekil 45'de gösterildiği üzere dataaugment işlemi gerçekleştirılmıştır. Şekil 45'de A ile gösterilen işlem ile görüntüler - 45 derece ile 60 derece arasında döndürülmüştür. Bu döndürme işleminde tercih edilen döndürme açısı kullanıcının ihtiyacına göre değiştirilebilmektedir. Aynı Şekil 45 (b)'de gösterilen ölçeklendirme işlemi içinde geçerlidir. Şekil 45 (c)'de gösterilen seçenek ile de eğitim ve test verilerinin oranı belirlenmektedir. Aynı şekilde Randomize seçeneği ile de bu veriler rastgele olarak karıştırılabilmektedir. Bu aşamadan sonra eğitim verileri ve sınıfları Şekil 46'da gösterilmiştir. Mimaride kullanılmak üzere veriler eğitim ve test seti olarak ayırdıktan sonra eğitim aşamasına geçilmektedir.

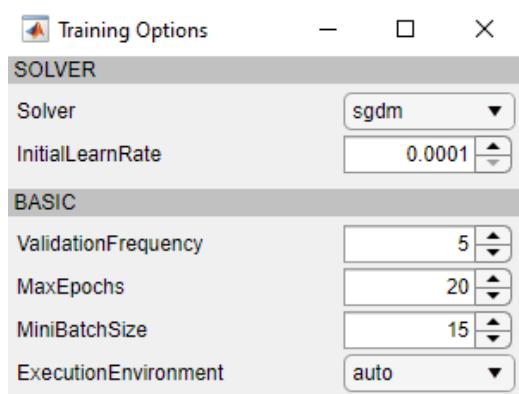


Şekil 45: Dataaugment

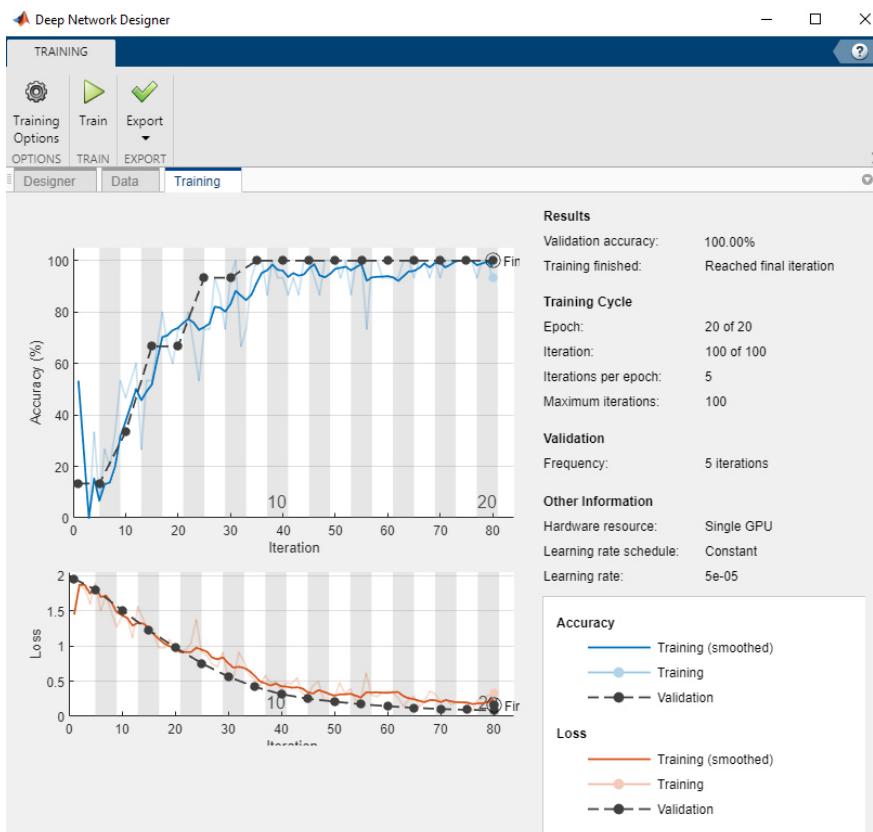


Şekil 46: Import Edilen Eğitim Verileri

Bu aşamada Şekil 46'da gösterilen training sekmesi seçildikten sonra training options penceresinden Şekil 47'deki ayarlamalar yapılmaktadır.



Şekil 47: Training Options Ayarları



Şekil 48: Simülasyon Sonucu

Bu ayarlamaların sonrasında da Şekil 48'deki Training butonuna basılarak sistem eğitim ve test için çalıştırılmaktadır. Eğitilmiş MobileNetV2 mimarisi daha sonraki bir zamanda kullanılmak üzere istenilirse SqueezeNet mimarisinde anlatıldığı gibi kaydedilebilmektedir.

3.2.1. Sistemin komut satırı ile test edilmesi

Eğitim işlemi tamamlanan MobileNetV2 mimarisi komut satırı ile aşağıdaki şekilde test edilmiştir.

```
% %% MobileNet V2 mimarisinin test edilmesi
predLabels = classify(net,imdsTest);
testLabels = imdsTest.Labels;

accuracy = sum(predLabels ==
testLabels)/numel(testLabels);
fprintf('Accuracy is %8.2f%%\n',accuracy*100)
```

3.2.2. Ön eğitilmiş ağdan öznitelik elde etme

Çalışmada oluşturulan mimari eğitilmiş ve test verileri ile doğruluğu hem kod satırından hem de simülasyon ile ölçülmüştür. Kullanılan sınıflandırma katmanı doğrudan mimariye ait olan sınıflandırma katmanıdır. Eğer çalışmada farklı sınıflandırma algoritmaları test edilmek istenilirse, mimariden elde edilen öznitelikler kullanılarak bu işlem gerçekleştirilebilir.

Tasarlanan mimariden öznitelik elde etmek için sırasıyla aşağıdaki komut satırları çalıştırılır.

```
load MobileNetV2;

net=trainedNetwork_1;
analyzeNetwork(net);

inputSize = net.Layers(1).InputSize;
imds =
 imageDatastore('..../MerchData','IncludeSubfolders',true,'
LabelSource','foldernames');
[imdsTrain,imdsTest] =
 splitEachLabel(imds,0.8,'randomized');
augimdsTrain =
 augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest =
 augmentedImageDatastore(inputSize(1:2),imdsTest);

layer = 'fc';
featuresTrain =
 activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest =
 activations(net,augimdsTest,layer,'OutputAs','rows');

whos featuresTrain

YTrain = imdsTrain.Labels;
YTest = imdsTest.Labels;

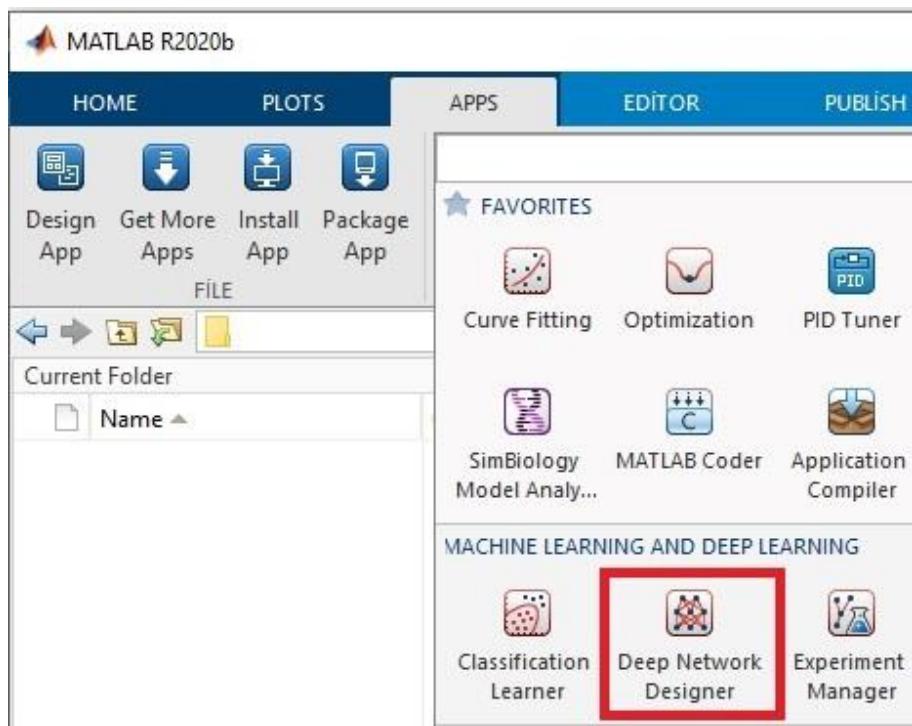
%Öznitelikler ve bunlara bağlı çıkış değerleri mimariden
farklı olarak SVM (fitcecoc) kullanılarak sınıflandırma
işleminin gerçekleştirilmesi

classifier = fitcecoc(featuresTrain,YTrain);
YPred = predict(classifier,featuresTest);

%Doğruluk oranının elde edilmesi
accuracy = sum(YPred == YTest)/numel(YTest);
fprintf('Accuracy is %8.2f%%\n',accuracy*100)
```

3.3. RESNET

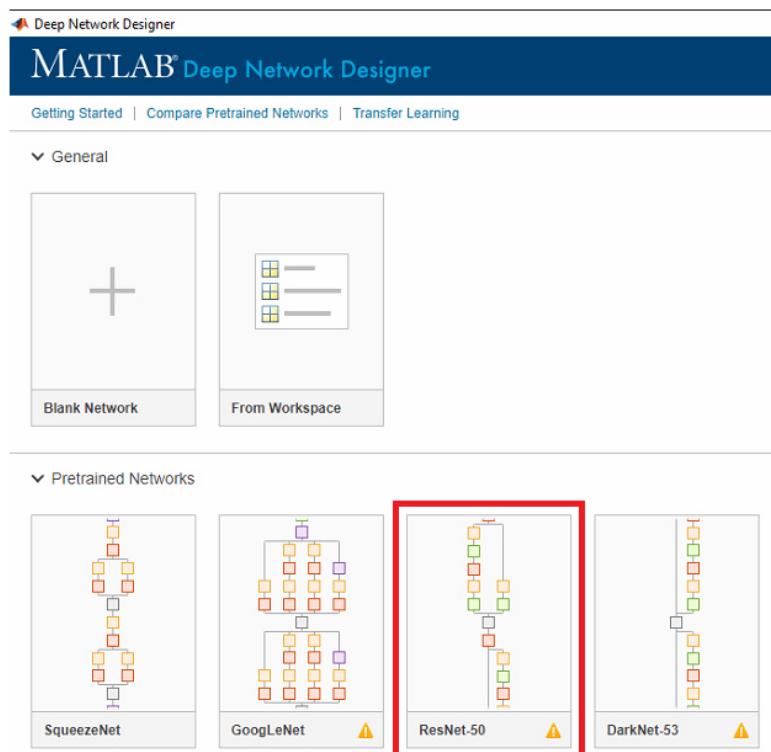
RESNET 50'nin genel mimarisi Bölüm 2 Şekil 18'de detaylı olarak sunulmuştur. SqueezeNet ve MobileNetV2 uygulamalarından farklı olarak, bu mimari MATLAB ortamında araç kutuları kullanılarak çağrılacaktır. Bu işlem için ilk olarak Şekil 49'daki araç kutuları menüsü açılmaktadır.



Şekil 49: MATLAB Araç Kutusu

Şekil 49'de açılan menüden kırmızı kutu içerisinde gösterilen DeepNetwork Designer seçilmelidir. Böylece daha önce komut satırı ile çağrılan ekran, RESNET 50 mimarisinde araç kutusu ile ekrana getirilmiştir. MobileNetV2 mimarisinde olduğu gibi bu mimari de

MATLAB ortamında kurulu değildir. Bu nedenle diğer uygulamalarda ve Şekil 50-51'de gösterildiği gibi RESNET 50 mimarisi bilgisayar ortamına yüklenmelidir.



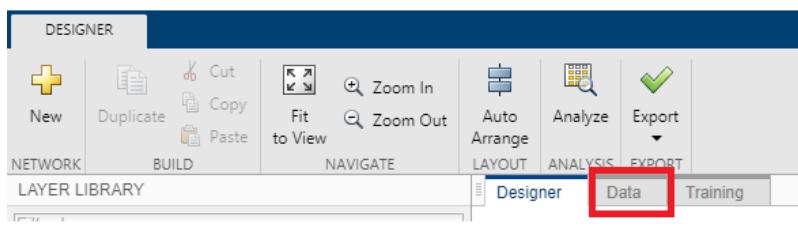
Şekil 50: RESNET50 Kurulum Ekranı

RESNET 50 mimarisi MATLAB ortamına yüklenikten sonra çalışmada kullanılacak olan verileri mimariye dahil edilmelidir.

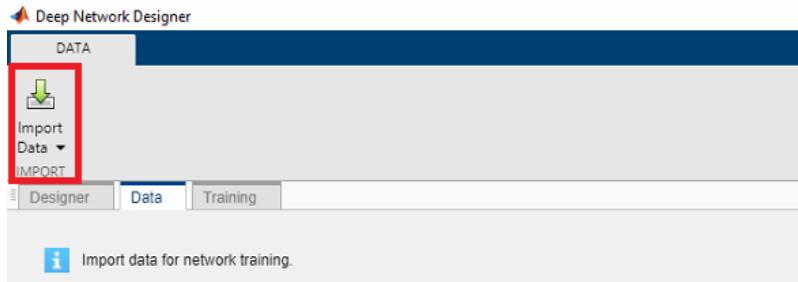


Şekil 51: Kurulum Ekranı

Bunun için Şekil 52(a) ve (b)' de gösterilen Data seçeneği kullanılarak veriler yüklenmelidir.



(a)

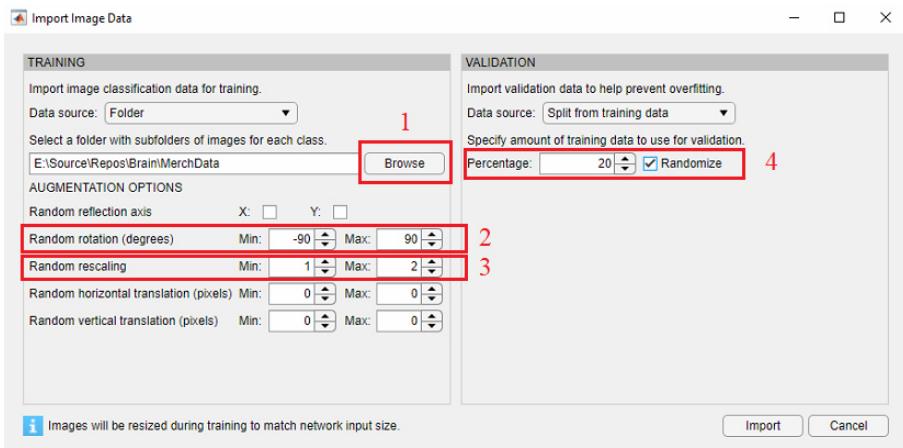


(b)

Şekil 52: Çalışmada Kullanılacak Verilerin Yüklenmesi

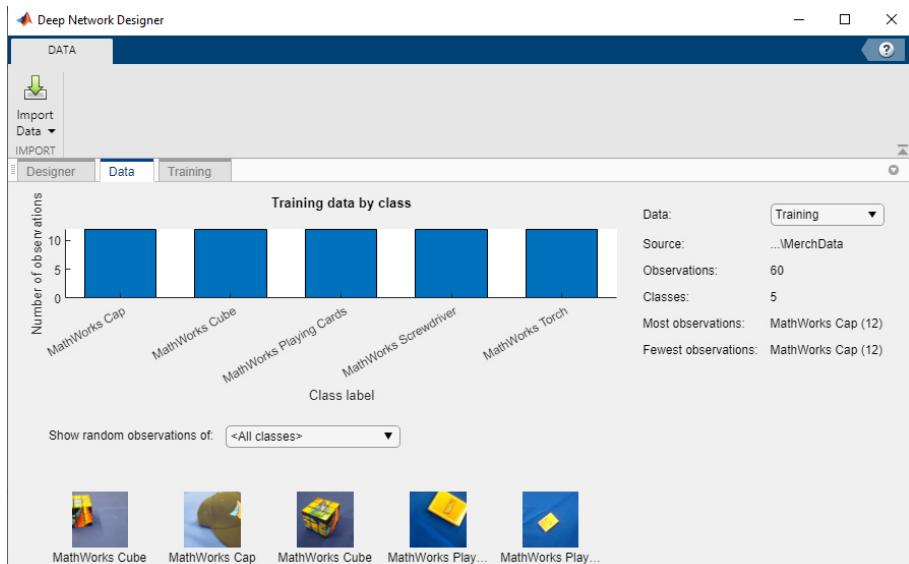
Çalışmada kullanılacak verilerin bir dosyadan seçimi, seçilen veriler üzerinde `data_augment` işlemi ve verilerin rastgele bir şekilde

belirlenen oranda eğitim ve test verileri olarak ayrılma işlemi Şekil 53'de gösterilen adımlar sırası ile takip edilerek gerçekleştirilmektedir.



Şekil 53: Örnek Veri Setindeki Görüntülerini Yükleme

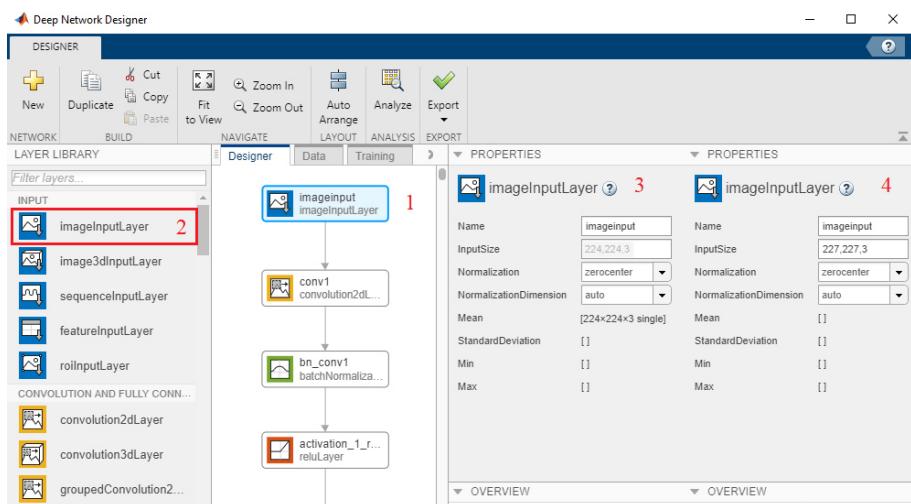
Sisteme toplam 75 adet görüntü yüklenmiştir. Şekil 53'de görüldüğü üzere yüklenen görüntülerin %80'i eğitim, %20'i test verisi olarak rastgele bir şekilde ikiye ayrılmıştır.



Şekil 54: Yüklenen Veriler ve Özellikleri

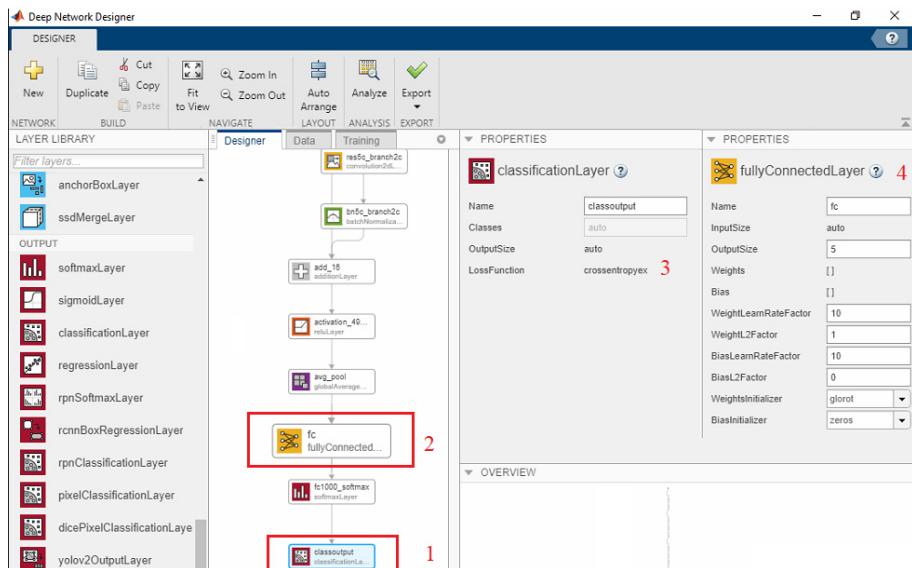
Yüklenen veriler ve sınıfları Şekil 54'de detaylı olarak gösterilmektedir.

Bir sonraki adımda ise kullanılan mimariyi, yüklenilen verilere göre tekrar revize etmek gerekmektedir. Bu işlem için ilk olarak Şekil 55'de 1. adımda gösterilen imageinput katmanı silinerek yerine 2. adımda Layer Library bölümünde gösterilen yeni bir imageinput katmanı mimariye eklenmelidir. Şekil 55'de 3. Adımda default olarak bulunan imageinput katmanın özelliklerini de 4. adımda gösterilen parametreler ile değiştirmelidir.



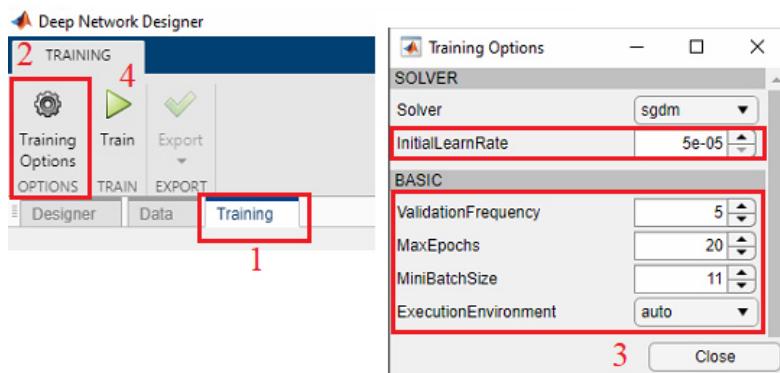
Şekil 55: Imageinput Katmanını Revize Etmek

Sonraki adımda ise Şekil 56'da 1. adımda gösterilen classoutput katmanı silinerek, Layer Library menüsünden yeni bir classoutput katmanı eklenmektedir. Buna ek olarak 2. adımda gösterilen fullyConnected katmanın veri çıktı sayısının aynı olmadığı için benzer şekilde silinerek değiştirilmelidir. Bu işlem gerçekleştirildiğinde 3. adımda gösterildiği gibi OutputSize'ın Auto ya dönüşmiş olması gerekmektedir. Değiştirilen fullyConnected katmanı 4. adım da gösterilen parametrelerle göre revize edilmiştir.



Şekil 56: Mimarının Revize Edilmesi

Kullanıcı istediği şekilde diğer katmanları da kendi isteği doğrultusunda revize edebilmektedir. Uygulamada ise sadece sınıflandırma ve fullyConnected katmanları revize edilmiştir. Revize işlemi bitirdikten sonra eğitim işlemi için Şekil 57'de gösterilen adımlar sırası ile uygulanmıştır.



Şekil 57: Eğitim Ayarlamaları

Tasarlanan mimariyi kullanılan verilere göre eğitmek için Şekil 57'deki adımlar tek tek takip edilmiştir. 3. adımdaki Training options seçenekindeki parametreler de revize edilmiştir. 4. adımda gösterilen Train butonuna basıldıktan sonra eğitim işlemi başlamakta ve bu işlem sonucunda Şekil 58'de gösterilen sonuç ekranı elde edilmektedir.



Şekil 58: Sonuç Ekranı

Şekil 57'de gösterilen ayarlara göre eğitilen ağın workspace alanına kaydedilmesi için Şekil 59'da gösterilen buton kullanılmalıdır.



Şekil 59: Eğitilmiş Ağın Çıkarılması

3.3.1. Sistemin komut satırı ile test edilmesi

Eğitim işlemi tamamlanan RESNET50 mimarisi kayıt edildikten sonra komut satırı ile aşağıdaki şekilde de test edilebilmektedir.

```
% %% RESNET50 mimarisinin test edilmesi
load RESNET50;

net=trainedNetwork_1;
predLabels = classify(net,imdsTest);
testLabels = imdsTest.Labels;

accuracy = sum(predLabels ==
testLabels)/numel(testLabels);
fprintf('Accuracy is %8.2f%%\n',accuracy*100)
```

3.3.2. Ön eğitilmiş ağdan öznitelik elde etme

Kullanılan verilere göre revize edilen RESNET50 mimarisi eğitilmiş ve test verilerine göre hem menüler ile hem de komut satırı vasıtası ile çalıştırılmıştır. Revize edilen mimaride sınıflandırma katmanında default olarak Softmax sınıflandırıcı kullanılmaktadır. Eğer çalışmada farklı sınıflandırma algoritmaları test edilmek istenilirse, mimariden elde edilen öznitelikler kullanılarak bu işlem gerçekleştirilebilir.

Tasarlanan mimariden öznitelik elde etmek için sırasıyla aşağıdaki işlemler uygulanmış ve sonrasında SVM ile sınıflandırma gerçekleştirilmiştir.

```
% Kaydedilen mimari ağ aşağıda yüklendi ve net
% değişkenine atandı.
load RESNET50;

net=trainedNetwork_1;
% mimari ağ analiz edildi.
analyzeNetwork(net);

inputSize = net.Layers(1).InputSize;
imds =
 imageDatastore('..../MerchData','IncludeSubfolders',true,
 'LabelSource','foldernames');

% Eğitim ve test verileri rastgele olarak %80'e %20
% olarak ayarlandı.

[imdsTrain,imdsTest] =
 splitEachLabel(imds,0.8,'randomized');

augimdsTrain =
 augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsTest =
 augmentedImageDatastore(inputSize(1:2),imdsTest);
```

```
layer = 'fc';
featuresTrain =
activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest =
activations(net,augimdsTest,layer,'OutputAs','rows');

whos featuresTrain
YTrain = imdsTrain.Labels;
YTest = imdsTest.Labels;

%Öznitelikler ve bunlara bağlı çıkış değerleri mimariden
%farklı olarak SVM (fitcecoc) kullanılarak sınıflandırma
%işleminin gerçekleştirilmesi

classifier = fitcecoc(featuresTrain,YTrain);
YPred = predict(classifier,featuresTest);

%Doğruluk oranının elde edilmesi
accuracy = sum(YPred == YTest)/numel(YTest);
fprintf('Accuracy is %8.2f%%\n',accuracy*100)
```

KAYNAKÇA

- Anandharajan, T R V, G Abhishek Hariharan, K K Vignajeth, and R Jijendiran. 2016. “Weather Monitoring Using Artificial Intelligence.” In *2016 2nd International Conference on Computational Intelligence and Networks (CINE)*, IEEE, 106–11.
- Arora, Raman, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. 2016. “Understanding Deep Neural Networks with Rectified Linear Units.” *arXiv preprint arXiv:1611.01491*.
- Bai, Lin, Yiming Zhao, and Xinming Huang. 2018. “A CNN Accelerator on FPGA Using Depthwise Separable Convolution.” *IEEE Transactions on Circuits and Systems II: Express Briefs* 65(10): 1415–19.
- Bekkerman, Ron, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. 2003. “Distributional Word Clusters vs. Words for Text Categorization.” *Journal of Machine Learning Research* 3(Mar): 1183–1208.
- Bi, Chongke et al. 2020. “Mobilenet Based Apple Leaf Diseases Identification.” *Mobile Networks and Applications*: 1–9.
- Buchanan, Bruce G. 2005. “A (Very) Brief History of Artificial Intelligence.” *AI Magazine* 26(4): 53.
- Chollet, Francois. 2018. *361 Deep Learning with Python*. Manning New York.
- Hajkowicz, Stefan et al. 2019. “Artificial Intelligence: Solving Problems, Growing the Economy and Improving Our Quality of Life.”
- Hinton, Geoffrey E. 2012. “A Practical Guide to Training Restricted Boltzmann Machines.” In *Neural Networks: Tricks of the Trade*, Springer, 599–619.
- Howard, Andrew G et al. 2017. “Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” *arXiv preprint arXiv:1704.04861*.
- Huang, Ming-Hui, Roland Rust, and Vojislav Maksimovic. 2019. “The Feeling Economy: Managing in the next Generation of Artificial Intelligence (AI).” *California Management Review* 61(4): 43–65.
- Iandola, Forrest N et al. 2016. “SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters And< 0.5 MB Model Size.” *arXiv preprint arXiv:1602.07360*.

- Jiang, Fei et al. 2017. "Artificial Intelligence in Healthcare: Past, Present and Future." *Stroke and vascular neurology* 2(4).
- Jones, Nicola. 2014. "Computer Science: The Learning Machines." *Nature News* 50(7482): 146.
- Khasoggi, B, E Ermatita, and S Sahmin. 2019. "Efficient Mobilenet Architecture as Image Recognition on Mobile and Embedded Devices." *Indonesian J. Electr. Eng. Comput. Sci* 16: 389–94.
- Kosovic, Branko et al. 2020. "A Comprehensive Wind Power Forecasting System Integrating Artificial Intelligence and Numerical Weather Prediction." *Energies* 13(6): 1372.
- Kumar, N, N Kharkwal, R Kohli, and S Choudhary. 2016. "Ethical Aspects and Future of Artificial Intelligence." In *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, , 111–14.
- Langford, John, and Robert Schapire. 2005. "Tutorial on Practical Prediction Theory for Classification." *Journal of machine learning research* 6(3).
- Liu, Mason, and Menglong Zhu. 2018. "Mobile Video Object Detection with Temporally-Aware Feature Maps." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, , 5686–95.
- Liu, Tianyi et al. 2019. "Towards Understanding the Importance of Shortcut Connections in Residual Networks." *arXiv preprint arXiv:1909.04653*.
- Liu, Xuefeng et al. 2019. "Real-Time Marine Animal Images Classification by Embedded System Based on Mobilenet and Transfer Learning." In *OCEANS 2019-Marseille*, IEEE, 1–5.
- Maas, Matthijs M. 2019. "How Viable Is International Arms Control for Military Artificial Intelligence? Three Lessons from Nuclear Weapons." *Contemporary Security Policy* 40(3): 285–311.
- Macrae, Carl. 2019. "Governing the Safety of Artificial Intelligence in Healthcare." *BMJ quality & safety* 28(6): 495–98.
- Matlab. 2021. "Matlab RESNET50 Documentation." https://www.mathworks.com/help/deeplearning/ref/resnet50.html?searchHighlight=resnet&s_tid=srchtitle.
- Maturana, Daniel, and Sebastian Scherer. 2015. "Voxnet: A 3d Convolutional Neural

- Network for Real-Time Object Recognition.” In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 922–28.
- McGovern, Amy et al. 2017. “Using Artificial Intelligence to Improve Real-Time Decision-Making for High-Impact Weather.” *Bulletin of the American Meteorological Society* 98(10): 2073–90.
- Metlek, Sedat, and Kiyas Kayaalp. 2020. “Derin Öğrenme ve Destek Vektör Makineleri İle Görüntüden Cinsiyet Tahmini.” *Düzce Üniversitesi Bilim ve Teknoloji Dergisi* 8(3): 2208–28.
- Michele, Aurelia, Vincent Colin, and Diaz D Santika. 2019. “Mobilenet Convolutional Neural Networks and Support Vector Machines for Palmprint Recognition.” *Procedia Computer Science* 157: 110–17.
- Morgan, Forrest E et al. 2020. *Military Applications of Artificial Intelligence: Ethical Concerns in an Uncertain World*. RAND PROJECT AIR FORCE SANTA MONICA CA SANTA MONICA United States.
- Najafabadi, Maryam M et al. 2015. “Deep Learning Applications and Challenges in Big Data Analytics.” *Journal of big data* 2(1): 1–21.
- Narin, Ali, Ceren Kaya, and Ziynet Pamuk. 2021. “Automatic Detection of Coronavirus Disease (Covid-19) Using x-Ray Images and Deep Convolutional Neural Networks.” *Pattern Analysis and Applications*: 1–14.
- Ongsulee, Pariwat. 2017. “Artificial Intelligence, Machine Learning and Deep Learning.” In *2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE)*, IEEE, 1–6.
- Pang, Yanwei, Manli Sun, Xiaoheng Jiang, and Xuelong Li. 2017. “Convolution in Convolution for Network in Network.” *IEEE transactions on neural networks and learning systems* 29(5): 1587–97.
- Qiu, Junfei et al. 2016. “A Survey of Machine Learning for Big Data Processing.” *EURASIP Journal on Advances in Signal Processing* 2016(1): 1–16.
- Rajpal, Sheetal et al. 2021. “Using Handpicked Features in Conjunction with ResNet-50 for Improved Detection of COVID-19 from Chest X-Ray Images.” *Chaos, Solitons & Fractals* 145: 110749.
- Riegler, Gernot, Ali Osman Ulusoy, and Andreas Geiger. 2017. “Octnet: Learning

- Deep 3d Representations at High Resolutions.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, , 3577–86.
- Rong, Guoguang et al. 2020. “Artificial Intelligence in Healthcare: Review and Prediction Case Studies.” *Engineering* 6(3): 291–301.
- Sandler, Mark et al. 2018. “Mobilenetv2: Inverted Residuals and Linear Bottlenecks.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, , 4510–20.
- Sion, Grafiela. 2018. “How Artificial Intelligence Is Transforming the Economy. Will Cognitively Enhanced Machines Decrease and Eliminate Tasks from Human Workers through Automation?” *Journal of Self-Governance and Management Economics* 6(4): 31–36.
- Stone, Peter et al. 2016. “Artificial Intelligence and Life in 2030: The One Hundred Year Study on Artificial Intelligence.”
- Wang, Wei, Yiyang Hu, et al. 2020. “A New Image Classification Approach via Improved MobileNet Models with Local Receptive Field Expansion in Shallow Layers.” *Computational Intelligence and Neuroscience* 2020.
- Wang, Wei, Hui Liu, et al. 2020. “Investigation on Works and Military Applications of Artificial Intelligence.” *IEEE Access* 8: 131614–25.
- Yu, Dong, and Li Deng. 2010. “Deep Learning and Its Applications to Signal and Information Processing [Exploratory Dsp].” *IEEE Signal Processing Magazine* 28(1): 145–54.
- Zhao, Zheng et al. 2017. “LSTM Network: A Deep Learning Approach for Short-Term Traffic Forecast.” *IET Intelligent Transport Systems* 11(2): 68–75.

ÖZGEÇMİŞLER

Dr. Öğr. Üyesi. Sedat METLEK

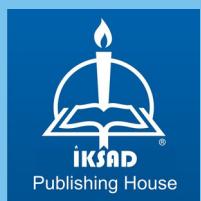


Dr. Sedat METLEK, Lisans eğitimini Marmara Üniversitesi Elektronik ve Bilgisayar eğitiminde, Yüksek Lisans eğitimini Süleyman Demirel Üniversitesi Elektronik Bilgisayar Eğitimi bölümünde ve Doktora eğitimini de Süleyman Demirel Üniversitesi Elektronik Haberleşme Mühendisliğinde tamamlamıştır. Halen Burdur Mehmet Akif Ersoy Üniversitesinde Dr. Öğretim Üyesi olarak çalışmaktadır. Yazarın Derin Öğrenme, Yapay Zekâ, Görüntü İşleme ve Makine Otomasyonu konularında birçok akademik ve ticari çalışması bulunmaktadır.

Öğr. Gör. Dr. Halit ÇETİNER



Dr. Halit ÇETİNER, Lisans eğitimini Selçuk Üniversitesi Bilgisayar Mühendisliği bölümünde, Yüksek Lisans eğitimini Süleyman Demirel Üniversitesi Bilgisayar Mühendisliği bölümünde ve Doktora eğitimini de Süleyman Demirel Üniversitesi Bilgisayar Mühendisliği bölümünde tamamlamıştır. Halen Isparta Uygulamalı Bilimler Üniversitesinde Öğr. Gör. Dr. olarak çalışmaktadır. Yazarın Derin Öğrenme, Yapay Zekâ, Görüntü İşleme ve Gömülü Sistem konularında birçok akademik ve ticari çalışması bulunmaktadır.



ISBN: 978-625-7562-08-9